Facultad de Biblioteconomía y Documentación
Universidad de Granada

_____

RECONOCIMIENTO Y CONTROL DE EXPRESIONES LINGÜÍSTICAS POR MEDIO DE TRANSDUCTORES DE ESTADO-FINITO

Carmen Gálvez

Tesis Doctoral

Director
Dr. Félix de Moya-Anegón

Departamento de Biblioteconomía y Documentación

Granada, 2003

Editor: Editorial de la Universidad de Granada
Autor: Carmen Galvez Martínez
D.L.: Gr. 2285- 2006
ISBN: 84-338-4153-x

Facultad de Biblioteconomía y Documentación
Universidad de Granada

RECONOCIMIENTO Y CONTROL DE EXPRESIONES LINGÜÍSTICAS POR MEDIO DE TRANSDUCTORES DE ESTADO-FINITO

Memoria de tesis doctoral presentada por *Carmen Gálvez*
y dirigida por Dr. *Félix de Moya-Anegón*
para la obtención del título de *Doctora en Documentación*

Departamento de Biblioteconomía y Documentación

Granada, 2003

People say the same things in different ways. This variation poses difficult problems for finding information in online text.

G. Grefenstette,
Xerox Research Centre Europe

Resumen

RECONOCIMIENTO Y CONTROL DE EXPRESIONES LINGÜÍSTICAS POR MEDIO DE TRANSDUCTORES DE ESTADO-FINITO

Carmen Gálvez Martínez
Departamento de Biblioteconomía y Documentación
Facultad de Biblioteconomía y Documentación
Universidad de Granada

Tesis Doctoral

Dirigida por: Prof. Dr. Félix de Moya-Anegón
Catedrático de Documentación de la Facultad de Biblioteconomía y Documentación de la Universidad de Granada

El objetivo principal de este trabajo es crear bases de información lingüísticas: *Diccionarios* y *Gramáticas electrónicas*, que se puedan utilizar por mecanismos automáticos de análisis para la identificación y agrupación de variantes léxicas y sintácticas en idioma español. El objetivo potencial de estas herramientas sería mejorar las entradas a los índices de los sistemas de RI, con técnicas procedentes del PLN. La metodología de investigación lingüística que hemos seguido es el modelo *hipotético-deductivo*, bajo este planteamiento hemos formulado *hipótesis explicativas* sobre las estructuras de las variantes lingüísticas, que posteriormente hemos comprobado y evaluado empíricamente en un *corpus de verificación*. Para la formulación de las *hipótesis* hemos empleado *Expresiones y Relaciones* Regulares, como mecanismo automático de control hemos utilizado una aplicación informática basada en *Transductores de Estado-Finito Gráficos*. Con este procedimiento se han obtenido los siguientes resultados: *a)* equiparar *Formas flexionadas* a *Formas controladas*, por medio de las bases de información léxicas, consistentes en *Diccionarios electrónicos con 61659 entradas en total*; *b)* equiparar *variantes estructurales de Sintagmas Nominales* con *estructuras controladas*, por medio de las bases de información sintácticas, consistentes en *137 herramientas de análisis sintáctico; c)* comprobar las *hipótesis explicativas*, mediante la aplicación de los analizadores, desarrollados con estas bases de información, sobre un *corpus de verificación*; y *d)* evaluar los resultados de esta aplicación, que se sintetizan como sigue: los analizadores léxicos son muy precisos, *96.6%*, y consiguen reducir las variantes en un *26.4%*, mientras que los analizadores sintácticos son también muy precisos, *95%*, pero tienen un índice de *exhaustividad* bajo, *51%*. Las conclusiones generales que hemos extraído son las siguientes: (*i*) los analizadores léxicos constituyen una *técnica de conflación adecuada*, aunque tienen un problema de *infraanálisis*; y (*ii*) los analizadores sintácticos constituyen una *técnica de conflación adecuada*, aunque tienen un problema de *sobreanálisis*, que se puede solucionar con la aplicación de *modelos probabilísticos*.

TABLA DE CONTENIDOS

LISTA DE TABLAS

LISTA DE FIGURAS

Introducción

En la mayoría de los sistemas de RI los documentos se indizan por *unitérminos*, pero éstos pueden ser ambiguos y no lo suficientemente específicos para la discriminación de información, por eso algunas sistemas utilizan *términos compuestos*, o *sintagmas*, obtenidos en muchos casos con métodos estadísticos. La aproximación tradicional de los sistemas de RI se basa en este tipo de técnicas de indización automática para representar el contenido de los documentos (Salton 1980) (Salton 1989), (Croft *et al.* 1991), (Frakes y Baeza-Yates 1992).

Sin embargo, los conceptos que representan estos términos se pueden manifestar de distintas formas, conocidas como *variantes lingüísticas*. Un estudio previo sobre el problema de las variantes lingüísticas en los sistemas de RI se encuentra en (Sparck Jones y Tait 1984). A su vez, en la categorización de variantes se pueden distinguir *variantes léxicas o morfológicas*, *variantes sintácticas* y *variantes semánticas* (Jacquemin, Klavans y Tzoukermann 1997). Con el objetivo de que no se pierdan documentos relevantes, los sistemas de RI reconocen y agrupan las variantes mediante *algoritmos de conflación*. El proceso de conflación se puede desarrollar con distintos métodos como *eliminación de afijos, segmentación de palabras, bigramas de letras, técnicas lingüísticas, búsqueda léxica a través de un tesauro*, o *normalización de sintagmas*. Las técnicas más usadas son la *eliminación de afijos* y la

búsqueda léxica a través de un tesauro, ésta última se relaciona con el reconocimiento de las variantes semánticas, y no se va a tratar en este trabajo.

La aplicación de *técnicas de conflación* a las unidades léxicas tiene como objetivo que las distintas variantes léxicas se puedan considerar unidades equivalentes en los sistemas de RI. Los algoritmos de *stemming* son una de las *técnicas de conflación* más extendidas por medio de las cuales las variantes flexionales y derivacionales se reducen a un forma común, o *forma canónica*. Dos de los algoritmos de *stemming* más conocidos son el *algoritmo de Lovins* (Lovins 1968), y el *algoritmo de Porter* (Porter 1980). Otras técnicas, que se enfrentan al problema de la variabilidad del lenguaje con métodos lingüísticos, consisten en la *fusión* de las variantes léxicas en *palabras base*, o *lemas*. El proceso de lematización, o análisis morfológico de las variantes y su reducción a formas controladas, se desarrolla a partir de bases de información léxicas, almacenadas en *diccionarios electrónicos*, o *lexicones computacionales*. Uno de los analizadores morfológicos, que emplea esta técnica, es el desarrollado por Karttunen (Karttunen 1983).

De la misma forma, la aplicación de *algoritmos de conflación* a las unidades sintácticas se basa en la agrupación de las distintas variantes sintácticas a través de técnicas de equiparación de *patrones sintácticos*, generados a través de *gramáticas de sintagmas nominales*, que se consideran básicamente *gramáticas de patrones*. Los sistemas que utilizan estas técnicas realizan un análisis *lingüístico superficial* de determinados segmentos, o fragmentos textuales. Además del análisis superficial y el análisis de fragmentos del *corpus*, muchos sistemas emplean procesos de *desambiguación* de etiquetas. Las variantes sintácticas obtenidas con estos métodos se pueden agrupar, por último, en *estructuras sintácticas canónicas*. Estudios previos sobre la identificación de variantes sintácticas se pueden encontrar en (Schwarz 1990), (Sheridan y Smeaton 1992) y (Strzalkowski 1996).

La identificación y agrupación de variantes léxicas y sintácticas se puede considerar un proceso de *normalización*, consistente en comprobar si un término es una forma normalizada y, de no ser así, reemplazarlo por una *forma canónica*. Con la hipótesis de que las *técnicas de*

conflación –cuando se aplican a las variantes léxicas y sintácticas– con métodos lingüísticos se podrían considerar *técnicas de normalización, o de control*, cuya función sería regularizar variantes lingüísticas en *formas controladas*. Las cuestiones que nos vamos a plantear en este trabajo son la selección del modelo de representación y del mecanismo automático de reconocimiento para realizar este proceso.

1. Planteamiento general

La identificación y agrupación de las unidades lingüísticas dentro de los sistemas de RI se considera un área de aplicación propia del *Procesamiento del Lenguaje Natural* (PLN). Los sistemas de PLN poseen la siguiente estructura general: *a)* una *base de conocimiento* integrada por un *lexicón* y una *gramática*; y *b)* un *parser*, o programa que contiene un conjunto de algoritmos o instrucciones para procesar los datos lingüísticos anteriores. En el proceso de análisis de cadenas, el *lexicón* tiene la función de aportar la fuente de categorización sintáctica, y la *gramática* tiene la función de reflejar las relaciones estructurales de los componentes analizados. En el proceso de reconocimiento de cadenas, el *parser* toma el lexicón y la gramática, y decide si esa cadena de entrada se puede derivar de la gramática; y si es así, producir como salida algún tipo de representación para ella, como puede ser una estructura analizada, o un *árbol de derivación*.

Las técnicas de análisis sintáctico se derivan de las que se utilizan en los lenguajes formales, estableciéndose un paralelismo entre el proceso de reconocimiento de variantes lingüísticas y el proceso de reconocimiento de patrones, *pattern matching*. A partir de la relación anterior, vamos a vincular, por un lado, el modelo que vamos a adoptar para representar la *base de conocimiento* y, por el otro, el mecanismo, o *parser*, que vamos a adoptar para su reconocimiento automático. Para ello, es preciso hacer un breve recorrido en el tiempo hasta situar el momento en el cual se establece la conexión entre las estructuras lingüísticas y los mecanismos computacionales que son capaces de reconocerlas.

Básicamente, el PLN se basa en la modelización *matemática* del lenguaje, y dentro de los modelos matemáticos se distinguen nítidamente dos concepciones: *modelos simbólicos* y *modelos probabilísticos, o estocásticos*. En el origen de estos modelos se encuentran la *Máquina de Turing*, la aportación de Kleene (Kleene 1956) sobre Autómatas Finitos y Expresiones Regulares, y el trabajo de Shannon (Shannon y Weaver 1949) sobre la aplicación de los *procesos probabilísticos de Markov* a los Autómatas Finitos. A partir de todos estos trabajos, Chomsky fue el primero en considerar los *autómatas* como mecanismos para caracterizar las estructuras del lenguaje a través de las *gramáticas* (Chomsky 1956), sentado con ello las bases de la *Teoría de los Lenguajes Formales*. De forma simplificada, todas estas aportaciones fueron el origen de los dos grandes modelos matemáticos en los que se dividen las investigaciones del PLN y que han evolucionado tradicionalmente de forma separada (Jurafsky y Martin 2000):

a. *Los modelos simbólicos* construidos a partir de la formulación de la *Teoría de los Lenguajes Formales* y de la utilización de los autómatas para reconocer gramáticas. Estos modelos se basan en el *álgebra* y la *teoría de conjuntos*, en ellos los lenguajes se definen como sistemas formales compuestos por secuencias de símbolos y reglas, que establecen las combinaciones entre los símbolos. El procedimiento de investigación lingüística relacionado con estos modelos es el método *hipotético-deductivo*.

b. *Los modelos probabilísticos* construidos a partir de la *Teoría de la Información* y de la aplicación de probabilidades de transición a los autómatas. Estos modelos se basan en técnicas *orientadas-a-los-datos*, obtenidos con *métodos estadísticos* a partir de grandes *corpus lingüísticos*. Dentro de este grupo, se incluirían los modelos basados en el procesamiento de las probabilidades de aparición de las unidades lingüísticas por medio de *Modelos de Markov*. El procedimiento de investigación lingüística con estos modelos es el método *científico-inductivo*.

En este trabajo, vamos a adoptar el planteamiento de los *modelos simbólicos*, en el que el lenguaje se concibe como un conjunto *infinito* de secuencias generadas por un sistema *finito* de símbolos y reglas, denominado *gramática*. En relación con esto, si un *autómata* es un mecanismo abstracto que reconoce símbolos y decide si una cadena de entrada es *aceptada, o no* por el autómata, se podría establecer una conexión entre *gramáticas* y *autómatas*. Y éste fue precisamente el planeamiento de Chomsky: *concebir los autómatas como mecanismos que generan todas y sólo aquellas secuencias gramaticales de un lenguaje*.

Basándonos en la relación anterior, vamos a adoptar los formalismos y los mecanismos menos poderosos para plantear el problema del reconocimiento y agrupación de patrones lingüísticos: de la *Teoría de los Lenguajes Formales*, vamos a adoptar los formalismos *menos expresivos*, como son las *Gramáticas Regulares*, y de la *Teoría de Autómatas*, vamos a adoptar los reconocedores *más débiles*, como son los *Autómatas de Estado-Finito*. A su vez, la vinculación entre las *Gramáticas Regulares* y los *Autómatas de Estado-Finito* la vamos a establecer sistemáticamente a través del *Teorema de Análisis* y del *Teorema de Síntesis* de Kleene (Kleene 1956).

Retomando nuevamente la idea de que la arquitectura de un sistema de PLN con modelos simbólicos tiene básicamente dos componentes –un *base de conocimiento* y un *parser*, que opera sobre la base de conocimiento para generar estructuras analizadas– y teniendo en cuenta que hemos establecido un paralelismo entre el reconocimiento de variantes léxico-sintácticas y el reconocimiento de *patrones léxico-sintácticos*, las cuestiones básicas que vamos a tratar, relacionadas con las técnicas del PLN, son:

a. La construcción de bases de información lingüísticas: *Diccionarios electrónicos* y *Gramáticas electrónicas*.
b. La selección del mecanismo computacional, o *parser*, que vamos a adoptar para procesar de forma automática esa base de conocimiento y obtener unidades *conflactadas.*

Para la construcción de las bases de conocimiento vamos a tratar las variantes lingüísticas como *patrones lingüísticos* representados en *Expresiones Regulares*, *Relaciones Regulares* y *Gramáticas Regulares*. Para la selección del mecanismo computacional, o *parser*, que identifique las estructuras anteriores, vamos a adoptar una aplicación informática desarrollada por Silberztein (Silberztein 1999).

2. Objetivos

Con el planteamiento anterior, en este trabajo vamos a perseguir dos objetivos centrales:

1. *Creación de bases de información lingüísticas*: *Diccionarios* y *Gramáticas electrónicas*, que se puedan utilizar por mecanismos automáticos de análisis para la identificación y agrupación de variantes léxicas y sintácticas en idioma español.

2. *Comprobación de las hipótesis explicativas*, que hemos propuesto para los datos lingüísticos, aplicando los *Diccionarios y Gramáticas electrónicas* a un *corpus de verificación*, y posterior *evaluación de dicha aplicación.*

La finalidad potencial del desarrollo y evaluación de estas herramientas será mejorar, por medio de técnicas lingüísticas, las entradas a los índices de los sistemas de RI. Siguiendo con esto, el *primer objetivo*, relacionado con la construcción de las bases de información lingüísticas se desglosa en los siguientes *objetivos específicos*:

- Creación de bases de información léxicas: *Diccionarios electrónicos*

    - Describir la morfología flexiva de las unidades léxicas por medio de *hipótesis explicativas* representadas en *Relaciones Regulares*, que vinculan *Formas superficiales* a *Formas léxicas*, o canónicas.

- Construir diccionarios de *Formas léxicas simples* y *compuestas*.
- Asignar etiquetas a las *Formas léxicas simples y compuestas*.
- Resolver el problema de las irregularidades entre *Formas superficiales* y *Formas léxicas*.
- Trasladar las *Relaciones Regulares* léxicas a *Transductores de Estado-Finito Gráficos.*

- Creación de bases de información sintácticas: *Gramáticas electrónicas*

  - Describir las estructuras de los Sintagmas Nominales por medio de *hipótesis explicativas* representadas en *Expresiones Regulares*.
  - Calcular las *derivadas* de las *Expresiones Regulares* para la obtención de las *Gramáticas Regulares*.
  - Resolver el problema de la *recursividad* de determinadas estructuras sintagmáticas y el problema de la *ambigüedad* en la asignación de etiquetas.
  - Trasladar las *Gramáticas Regulares* a *Autómatas de Estado-Finito* Gráficos, o trasladar las estructuras de los grupos nominales directamente a *Autómatas de Estado-Finito*.
  - Representar las estructuras sintagmáticas como *Relaciones Regulares*, que vinculan las *variantes de las estructuras sintácticas* a *estructuras sintácticas controladas*.
  - Trasladar las *Relaciones Regulares* sintácticas a *Transductores de Estado-Finito Gráficos.*

El *segundo objetivo* es demostrar empíricamente las *hipótesis* que nos hemos planteado para representar las variantes lingüísticas y, a continuación, evaluar los resultados obtenidos. La primera etapa de este proceso consistirá en comprobar en un *corpus de verificación* si las secuencias reconocidas y agrupadas pertenecen, o no, al lenguaje generado por los analizadores léxico-sintácticos, esta comprobación se establecerá como un proceso de *deducción de las hipótesis* planteadas. La segunda etapa de este proceso será evaluar los

resultados de esta aplicación. En relación con todas estas cuestiones, el segundo objetivo se desglosa en los siguientes *objetivos específicos*:

- Aplicar los analizadores a un *corpus de verificación*, para comprobar empíricamente las *hipótesis explicativas* de los datos lingüísticos que hemos propuesto.
- Evaluar los resultados de la aplicación de los analizadores léxicos, desarrollados con los *Diccionarios electrónicos*, para identificar y agrupar variantes léxicas en estructuras *léxicas canónicas.*
- Evaluar los resultados de la aplicación de los analizadores sintácticos, desarrollados con las *Gramáticas electrónicas*, para identificar y agrupar las variantes sintácticas en estructuras *sintácticas canónicas.*

Por último, aunque no se pueda considerar un objetivo central en este trabajo, se va a buscar la fusión de los *modelos simbólicos* y los *modelos probabilísticos*. Con este propósito, a lo largo de este estudio vamos a intentar contrastar las soluciones que aportan los modelos simbólicos y los modelos probabilísticos ante los problemas de ambigüedad, y apuntar, al final, un leve acercamiento de ambos.

3. Estructura del trabajo

El *primer objetivo*, relacionado con la construcción de las bases conocimiento, se alcanza a lo largo de este trabajo como sigue:

En el **Capítulo 1** se *justifica* la realización de esta investigación ante el problema que presentan las variantes lingüísticas en los sistemas de RI. Para ello, se definen y clasifican los distintos tipos de variantes. Se analizan las distintas técnicas de *fusión* de variantes a partir de dos distinciones generales: *variantes morfológicas* y *variantes sintácticas*. Además del tipo de variantes se van a describir los distintos procedimientos generales que se utilizan

comúnmente para su reducción, según si emplean *técnicas no-lingüísticas*, o *técnicas lingüísticas*. También se va a analizar la complejidad de los procesos implicados en cada una de estas técnicas y se van a exponer los problemas que presentan la elección de uno u otro procedimiento.

En el **Capítulo 2** se va a realizar una *revisión* de la *Teoría de Lenguajes y Gramáticas Formales*. Se va a exponer la importancia de las *Gramáticas* como mecanismos generadores de cadenas lingüísticas a través de un sistema de reglas, que proporcionan descripciones formalizadas de dichas cadenas de un modo explícito. Las *Gramáticas* se van a plantear, de este modo, como *hipótesis explicativas* sobre la formación de las secuencias de una lengua. Esas hipótesis se deben comprobar empíricamente para demostrar si proporcionan, o no, una explicación satisfactoria de los datos lingüísticos. Con este objetivo, se van a utilizar mecanismos automáticos, desarrollados a partir de la *Teoría de Autómatas*, como una vía para caracterizar las *Gramáticas*. Para ello, vamos a tener en cuenta que, en el modelo teórico de los lenguajes formales se usa el *álgebra* y la *teoría de conjuntos* para definir los lenguajes como secuencias de símbolos, y con el mismo sistema vamos a desarrollar las descripciones de las cadenas lingüísticas. Así, las estructuras lingüísticas se van a describir con los medios de la *Teoría de los Lenguajes Formales*. Además, en este capítulo se va a realizar una revisión de las primeras aplicaciones de los lenguajes formales al PLN. En este sentido, se va a considerar la gran relevancia que tuvo la primera consideración teórica de la representación de las reglas fonológicas en *reglas de dos-niveles* (Johnson 1972), lo que supuso su posible modelización con técnicas de estado-finito, que dio origen a la *morfología de dos-niveles* (Koskenniemi 1983) y a investigaciones posteriores sobre la posibilidad de tratar determinados fenómenos lingüísticos por medio de transductores.

El **Capítulo 3** está dedicado a realizar una *revisión* de las técnicas de estado-finito y a los mecanismos que son capaces de reconocer Expresiones Regulares, fundamentalmente *Autómatas* y *Transductores de Estado-Finito*. Los conceptos básicos que vamos a tratar son los de *Lenguajes Regulares* y *Expresiones Regulares*. Los Lenguajes Regulares se construyen a partir de las operaciones básicas de *concatenación*, *unión* y *clausura de Kleene*. Para

representar estas operaciones se van a utilizar Expresiones Regulares, que proporcionan un *patrón* para las cadenas de ese lenguaje. A su vez, los mecanismos computacionales que manipulan Lenguajes Regulares pueden ser *aceptadores*, *reconocedores* y *transductores de cadenas*, a los que se les puede añadir un componente probabilístico. En este capítulo se va a realizar una revisión de todos estos mecanismos, y de las técnicas de estado-finito, fundamentalmente *determinación* y *minimización*.

En el **Capítulo 4** se va a exponer la *metodología* que hemos empleado para la creación de las bases de información lingüísticas y el desarrollo de los analizadores léxicos y sintácticos. El modelo de investigación que hemos adoptado es el método *hipotético-deductivo*, en el que se han planteado *hipótesis explicativas* de los fenómenos lingüísticos, basadas en la *Teoría de los Lenguajes Formales* y en la *Teoría de Autómatas*. Para la representación de las *hipótesis* hemos utilizado Expresiones Regulares y hemos obtenido las Gramáticas Regulares, y los Autómatas de Estado-Finito, equivalentes por medio de *derivaciones* de las Expresiones Regulares. Las derivaciones se han realizado según el *Teorema de Síntesis de Kleene* (Kleene 1956). A continuación, hemos trasladado las Gramáticas Regulares a los reconocedores de estado-finito correspondientes. Para establecer las Relaciones Regulares entre las *variantes de las estructuras sintácticas* y las *estructuras sintácticas controladas* se van a utilizar *Transductores de Estado-Finito Gráficos*. En este capítulo, también se van a tratar dos problemas: *1)* las irregularidades entre *Formas Léxicas* y *Formas Superficiales*, y *2)* la *ambigüedad* en el etiquetado. Para resolver el primer problema se va a optar por representar las irregularidades por medio de *Relaciones Regulares*, que se van a trasladar directamente a *Transductores de Estado-Finito Gráficos*, sin la mediación de complejas reglas morfológicas. Para resolver el segundo problema, se va a recurrir a *modelos simbólicos* basados en heurísticas, o reglas, como son las *Gramáticas Locales*. En relación con estos dos problemas, en este capítulo se van a exponer las soluciones que aportan otros modelos, *fundamentalmente los que aplican reglas de dos-niveles para resolver las irregularidades, entre formas superficiales/formas léxicas*, y los que aportan técnicas *estadísticas* para resolver la ambigüedad en el etiquetado.

En el **Capítulo 5** se va a desarrollar la *metodología* para la creación de las bases de conocimiento léxicas: *Diccionarios electrónicos*. Esta base de conocimiento se va a construir a partir de las unidades léxicas de un *corpus* restringido al dominio de la documentación. Los datos léxicos se van representar en un *diccionario de lemas*, o de *formas canónicas*, seguidas de un código vinculado a un *Transductor de Estado-Finito Gráfico*, que va a contener la descripción de la morfología flexional. La proyección de los transductores sobre el diccionario de formas canónicas va a dar lugar a la generación automática del *diccionario expandido de formas flexionadas*. A su vez, en este capítulo vamos a tratar los diferentes aspectos formales de la *flexión nominal*, *adjetival* y *verbal*. Además, se va resolver el problema de las irregularidades en la flexión entre *Formas Léxicas* y *Formas Superficiales*. Por otra parte, se va a proponer, aunque no se va a desarrollar, otra forma de representar la base de conocimiento léxico por medio de *Transductores Léxicos Gráficos*, y se van a indicar sus posibilidades para reconocer y agrupar subsistemas lingüísticos como: *formas derivadas*, *variantes de nombres personales*, o *expresiones de dominio*.

En el **Capítulo 6** se va a desarrollar la *metodología* para la creación de las bases de conocimiento sintácticas: *Gramáticas electrónicas que representan las estructuras de los Sintagmas Nominales*. Con este objetivo, las estructuras de los *Sintagmas Nominales* se van a especificar por medio de *hipótesis lingüísticas* formuladas en términos de Expresiones Regulares. Seguidamente se van a obtener las Gramáticas Regulares, que generan dichas estructuras, a partir de las *derivaciones* de las Expresiones Regulares. A continuación, se van a establecer las equivalencias entre Gramáticas Regulares y Autómatas de Estado-Finito. De este modo, para la representación de los grupos nominales se van a utilizar *Gramáticas Regulares Parciales*, en forma *patrones sintácticos*, y para el proceso de reconocimiento se van a utilizar también *analizadores de estado-finito parciales*, o fragmentales (*chunkers*) con el objetivo de identificar sólo esos *patrones nominales*. Por último, las variantes de las estructuras de los grupos nominales se van a representar por medio de *Relaciones Regulares*, que se van a trasladar directamente a *Transductores de Estado-Finito Gráficos*. Los transductores se van a encargar de relacionar las variantes sintácticas con *formas controladas*. Por otra parte, para resolver el problema de la *iteración* de constituyentes sintagmáticos se va

a recurrir a los *operadores de Kleene*, y para resolver el problema de la *recursividad*, cuando no sea posible obtener Gramáticas Regulares equivalentes, se va a poner un *límite a los fenómenos recursivos*.

El *segundo objetivo* planteado en este trabajo, relacionado con la comprobación empírica de las *hipótesis* aplicando los analizadores a un *corpus de verificación* y con la evaluación de esta aplicación, se alcanza como sigue:

El **Capítulo 7** se va a dedicar a dos cuestiones: *a)* la comprobación de las *hipótesis explicativas* que hemos propuesto sobre los datos lingüísticos con los que hemos desarrollado las bases de información, y *b)* la evaluación de los resultados de la aplicación de los analizadores. Para comprobar dichas *hipótesis*, primero se va a constatar empíricamente el número de *equiparaciones* de los patrones sintácticos en el *corpus de verificación*, y después se va a evaluar si esas equiparaciones son *correctas*, o *incorrectas*. Los parámetros de evaluación que se van a utilizar son una adaptación de la métrica clásica empleada en los sistemas de RI, como son los índices de *precisión/exhaustividad*. Con ello se va a calcular la eficacia de los analizadores léxicos y sintácticos para reconocer y agrupar las variantes lingüísticas en *formas controladas*. Por último, para completar este capítulo, se va a discutir si los analizadores desarrollados con técnicas lingüísticas son adecuados como generadores potenciales de las entradas a los índices de los sistemas de RI.

Finalmente, en el **Capítulo 8** se van a sintetizar las principales conclusiones a las que hemos llegado, a partir de los dos objetivos que nos hemos planteado, y se van a exponer los desarrollos futuros que pretendemos realizar.

Capítulo 1

El PROBLEMA DE LAS VARIANTES LINGÜÍSTICAS EN LOS SISTEMAS DE RECUPERACIÓN DE INFORMACIÓN

El objetivo de los sistemas de Recuperación de Información (RI) es recuperar de entre una colección de documentos aquellos que respondan a una necesidad de información, y ordenar estos documentos de acuerdo a un factor de relevancia. Este proceso se realiza normalmente por medio de: *a) métodos estadísticos* que se encargan de seleccionar los términos que representan mejor el contenido de los documentos; y *b)* un *fichero de índice inverso* que proporciona acceso a los documentos que contienen esos términos (Salton y McGill 1983). A su vez, la relación entre consultas y documentos se establece por el número de *términos* que tienen en común, para ello tanto las consultas como los documentos se representan por un *conjuntos de características* o por *términos de índice, index terms,* derivados directamente del

texto de los documentos, o indirectamente a través de un tesauro, por un proceso de *indización manual, o automático.*

Aunque en el proceso de equiparación entre los términos de las consultas y de los documentos intervengan distintas técnicas avanzadas de recuperación de información, un problema sin resolver sigue siendo la *inadecuada representación de ambos* (Strzalkowiski et al. 1999). El origen de esta inadecuación está en que los términos tienen *variantes morfológicas*, *léxicas* y *sintácticas* que no se pueden reconocer por simples *algoritmos de equiparación-de-términos* sin algún tipo de *procesamiento del lenguaje natural* (Hull 1996). En relación con esto, hay una intuición generalmente aceptada de que las técnicas del *Procesamiento del Lenguaje Natural* (PLN) podrían mejorar el rendimiento de sistemas de RI, sin embargo todavía no se ha podido establecer con suficiente claridad cómo incorporar los avances del PLN a los sistemas de recuperación.

La aplicación de técnicas del PLN a los sistemas de RI se basa en el hecho de que el contenido del documento y de la consulta se representa por medio de estructuras lingüísticas. Las representaciones de estas estructuras se configuran como términos de índice que pueden ser: *palabras simples*, *stems*, *phrases*, o *conceptos* (Belkin y Croft 1987). A su vez, estos términos pueden tener asociada una *ponderación*, conocida como $tf \cdot idf$, cuyo valor depende básicamente de la frecuencia de un término en un documento dado, del número de documentos que tienen asignado el término, y del número total de términos asignados a los documentos (Salton y McGill 1983). Sin embargo, aunque la base fundamental de muchos sistemas de RI sea la aplicación de cálculos estadísticos basados en la distribución de los términos, y su valor para la *discriminación de información*, nuestro objetivo va a estar dirigido a una cuestión previa: *el análisis léxico y sintáctico de los términos de índice antes de que se conviertan en elementos de recuperación*.

En una primera aproximación se pueden distinguir *términos simples*, *single terms*, frente a *términos en contexto*, *terms in context*, donde indicadores de relación se encargan de conectar distintos identificadores, dando lugar a entradas compuestas o sintagmas, *phrases* (Salton y

McGill 1983). En el caso de los términos simples, los identificadores de contenido se conocen como términos de índice, *keywords* o descriptores, y se representan por *unitérminos* para expresar los conceptos incluidos en cada documento, esos términos son eventualmente combinados, o *coordinados*, cuando se formula la petición de búsqueda. Por otra parte, cuando los términos compuestos se utilizan para el propósito de la indización, el proceso consiste en el uso de *phrases* que pueden incluir nombres, adjetivos, preposiciones o distintos indicadores de relación dando lugar a un proceso que se conoce como *precoordinación* (Salton y McGill 1983). Además de los términos de índice, algunos sistemas disponen de una lista de *palabras vacías* que proporciona términos de indización *pobres* y, cuando son identificados, se eliminan como términos de índice candidatos.

Uno de los primeros sistemas RI que incorporó técnicas para reducir las variantes de los términos al radical, o *stem*, y técnicas para agrupar los términos en *phrases* fue el sistema SMART (Salton 1980) (Buckley et al. 1995). Aunque cada vez son más los sistemas que integran técnicas de reducción de variantes, con métodos lingüísticos y estadísticos, los intentos de construir índices por medio de la agrupación a formas canónicas que han tenido más éxito son los que emplean cálculos estadísticos. Por otra parte, se ha demostrado que la indización por el *stem*, eliminando los afijos de las palabras, no mejora sustancialmente la eficacia de la recuperación, al menos para el inglés (Harman 1991). Sin embargo experimentos en otras lenguas, con una morfología flexional más compleja que el inglés, demuestran justo lo contrario (Popovic y Willett 1992). En la misma situación se encuentra la aplicación de técnicas lingüísticas en la indización por términos compuestos o *phrases*, que según un experimento realizado por Fagan (Fagan 1989) se llega a la conclusión de que los sintagmas identificados con métodos estadísticos son más eficaces que los identificados con métodos lingüísticos.

A la controversia anterior, se suma que las técnicas lingüísticas son muy complejas y necesitan manejar gran cantidad de conocimiento sobre el propio lenguaje si se quieren aplicar a textos sin restricciones, como son los textos de las bases de datos. A esto se añade la falta de un modelo de recuperación generalmente aceptado que utilice términos compuestos

obtenidos con técnicas lingüísticas, lo cual constituye *uno de los mayores obstáculos para la evaluación del impacto y la viabilidad del PLN en la RI* (Strzalkowski et al. 1999). Sin embargo, y a pesar del fracaso de la aplicación del PLN en los sistemas de RI, nuevas investigaciones proponen combinar técnicas lingüísticas con técnicas estadísticas. Además, algo a favor de las técnicas de procesamiento del lenguaje es que están en continuo avance y han demostrado su eficacia en la obtención de relaciones de términos compuestos a través de la identificación de estructuras por medio de *formalismos gramaticales.*

Teniendo en cuenta todos estos inconvenientes, en este trabajo nos vamos a centrar en la primera etapa del proceso de indización automática en lenguaje natural, esto es, en el proceso de examinar algorítmicamente los términos de índice para generar y controlar las unidades que se incorporarán posteriormente como potenciales entradas al fichero de búsqueda. No obstante, aunque el análisis léxico y sintáctico se considere sólo una etapa de la indización automática su tratamiento es bastante costoso, en comparación con otras fases del proceso, porque requiere no sólo el análisis sino la agrupación de las variantes de los términos. En torno a esta cuestión, en este trabajo vamos a partir de la consideración de que las técnicas de análisis y selección de términos de índice no van a depender del método de indización utilizado, por eso aclaramos que los métodos de indización automática no son el objetivo de este trabajo –estudios detallados sobre este asunto se encuentran en (Salton y McGill 1983), (Sparck Jones 1984), (Fagan 1989) y (Salton 1989); por otra parte, la aplicación de nuevas técnicas de indización basadas en redes neuronales se propone en (Chen et al. 1998), (Moya Anegón et al. 1998), (Doszkocs et al. 1990), (Guerrero Bote et al. 2002)–.

1.1. Las variantes lingüísticas en los sistemas de RI

Uno de los mayores problemas de la primera etapa del proceso de indización automática en lenguaje natural es la gran cantidad de variantes lingüísticas que muestran los términos de índice. Una variante se define como *aparición textual, text occurrence*, que está

conceptualmente relacionada a un término, *original term*, y que se puede usar para buscar información en la bases de datos textuales (Jacquemin y Tzoukermann 1999). Las variantes se pueden clasificar con distintos criterios, de forma general Arampatzis (Arampatzis et al. 1998) establece la siguiente ordenación:

- *Variantes morfológicas* vinculadas a la estructura interna de las palabras, según las cuales el mismo término puede aparecer de distintas formas.
- *Variantes léxico-semánticas* vinculadas a la proximidad semántica de las palabras, según las cuales diferentes términos pueden representar el mismo significado y múltiples significados se pueden representar con el mismo término.
- *Variantes sintácticas* vinculadas a la estructura de los sintagmas, según las cuales estructuras sintácticas semánticamente equivalentes se presentan con estructuras sintácticas diferentes.

En la mayoría de los casos, las variantes lingüísticas se consideran unidades *similares semánticamente* que podrían tratarse como *equivalentes* en los sistemas de RI (Hull 1996). Para poder realizar esas equivalencias se utilizan procedimientos de reducción de variantes consistentes en agrupar los términos que se refieren a conceptos equivalentes mediante *algoritmos de conflación*, *conflation algorithms*. Dentro de estos algoritmos, los más utilizados son los métodos de reducción de variantes morfológicas por medio de algoritmos de *stemming* y los métodos de reducción de variantes léxico-semánticas por medio de búsqueda léxica, *lexical lookup*, o búsqueda a través de un tesauro (Jacquemin y Tzoukermann 1999). En este trabajo vamos a desarrollar procedimientos de reducción de variantes morfológicas y sintácticas, dejando de lado las variantes léxico-semánticas. Por otra parte, es preciso subrayar que muchas veces vamos a utilizar el concepto de '*variante léxica*' en un sentido amplio para referirnos a todas las alteraciones formales de los términos, como pueden ser variantes ortográficas o gráficas, y no sólo a las variantes morfológicas.

En relación con las variantes morfológicas, nos vamos a limitar a la proximidad superficial de las palabras –*word forms*– aunque con el objetivo de encontrar proximidad conceptual. Los

métodos de conflación de variantes morfológicas son muy diversos: *eliminación de afijos, truncamiento de cadena de caracteres, segmentación de palabra, n-gramas*, y *morfología lingüística* (Lennon et al. 1981). Otra ordenación de los métodos de reducción de variantes morfológicas parte de la distinción entre métodos *manuales y* métodos a*utomáticos*, a su vez dentro de estos últimos se incluirían *eliminación de afijos, variedad de sucesores, búsqueda en tabla, y n-gramas* (Frakes 1992). Según Frakes, los métodos de conflación automática se denominan comúnmente lematizadores, y de entre todos los procedimientos de *lematización* el que ha obtenido mejores resultados es el algoritmo de eliminación de afijos que consiste en reducir los términos a radicales o *stems* eliminando el sufijo más largo. Pero además, hay otros procedimientos como los *métodos de variedad de sucesores* de una cadena que consisten básicamente en calcular el número de caracteres diferentes que siguen a esa cadena y, una vez obtenido el cálculo de las variedades de sucesores para esa cadena, se utiliza esa información para segmentar el término; los *métodos de búsqueda en tabla* consistentes en almacenar en una tabla todos los términos de índice y sus correspondientes *stems*, y realizar las búsquedas a través de la tabla; y *los métodos n-gramas* que se basan en la agrupación de los términos según los diagramas compartidos (un *bigrama* es un par de letras consecutivas, un *trigrama* son tres letras consecutivas, y así hasta *n-gramas*).

De entre todos los procedimientos diseñados para la fusión o conflación de variantes morfológicas los que mejores resultados han obtenido son los algoritmos de *stemming*, o algoritmos de *eliminación de afijos*, y dentro de éstos los de *coincidencia más larga, longest match*. El empleo de estos algoritmos permite agrupar términos relacionados semánticamente, reduciéndolos a formas simples y evitando así que se pierdan documentos relevantes en el momento de la recuperación. Sin embargo, aunque las *técnicas de stemming* sean beneficiosas para la reducción del tamaño del fichero índice (Salton 1989), los *stems* obtenidos con este método muchas veces no son unidades lingüísticas que se puedan utilizar para otro tipo de procesamiento, como puede ser el *parsing* sintáctico porque, aún operando con aspectos de morfología flexional y derivacional, estas técnicas no realizan un auténtico análisis morfológico. Además, con los algoritmos de eliminación de afijos no se obtienen *auténticas palabras* que se puedan utilizar para otros propósitos de RI, *como técnicas*

interactivas que requieren que el usuario seleccione términos para una posible ampliación, o *expansión de la consulta* (Hull 1996).

La solución a este problema es usar técnicas de *stemming* que realicen un auténtico análisis morfológico y proporcionen unidades lingüísticamente correctas como son los *lemas*, definidos a grandes rasgos como un conjunto de términos con el mismo *stem* y, opcionalmente, con la misma categoría gramatical. Estas unidades se obtienen con métodos lingüísticos, o técnicas de *lematización*, que utilizan diccionarios o lexicones susceptibles de representarse con formalismos de estado-finito. Aunque la obtención de unidades lingüísticamente correctas sea aparentemente una cuestión irrelevante en los sistemas de recuperación, sí son de gran utilidad para el posterior reconocimiento de construcciones complejas como son los sintagmas. Todo esto nos lleva finalmente a considerar sólo dos de las técnicas más importantes para la reducción de variantes morfológicas como son:

1. Técnicas no-lingüísticas: *métodos de stemming basados en la eliminación de afijos*
2. Técnicas lingüísticas: *métodos de lematización basados en lexicones*, representados en Máquinas de Estado-Finito.

En relación con la reducción de las *variantes léxico-semánticas*, la mayoría de los sistemas emplean procedimientos de *búsqueda léxica, lexical lookup*, por medio de diccionarios o tesauros que se utilizan para agrupar dos palabras que son completamente diferentes en la forma (Paice 1996). Los métodos *stemming* se aplican cuando los términos se asemejan morfológicamente, pero cuanto la semejanza es semántica se usan métodos de *búsqueda léxica*. Ambos procedimientos son además complementarios porque el *stemming* emplea las *similitudes gráficas para inferir proximidad léxica*, mientras que la *búsqueda léxica* se basa en datos *terminográficos con enlaces a sinónimos* (Jacquemin y Tzoukermann 1999). Como ya se ha indicado, las *variantes léxico-semánticas* no se van a tratar en este trabajo, aunque se va a proponer, en el capítulo correspondiente, la posibilidad de utilización de formalismos de estado-finito para agrupar términos sinónimos vinculándolos a formas normalizadas.

En lo que respecta a las variantes sintácticas, tenemos que el inconveniente de las técnicas *basadas-en-términos-simples* es que parten de la suposición de que los términos son independientes, y esto es falso en muchos casos. La mayoría de los sistemas de RI usan estos modelos, en los que el contenido de cada documento se representa por una colección no estructurada de *unitérminos –stem*, o *lemas*– sin incluir ningún tipo de relación. La independencia de los términos da lugar a su independencia estadística, y esto provoca representaciones inexactas que reducen la eficacia de los sistemas de RI. Un procedimiento más adecuado para producir índices mejores consistiría en identificar *multitérminos* o sintagmas, especialmente si esos sintagmas son significativos, *meaningful phrases*, y representan *conceptos importantes en el dominio de la base de datos* (Strzalkowski et al. 1999).

Los sintagmas que representan conceptos se incluyen dentro de lo que se denomina *descriptores complejos*. Se trata de indicadores del contenido integrados por más de un término que presentan de forma general dos tipos de relaciones: *a) semánticas*, y *b) sintácticas*. Las relaciones semánticas dependen del significado inherente de los términos implicados y se representan en las clases de un tesauro, mientras que las relaciones sintácticas dependen de la estructura gramatical de esos mismos términos y se representan en sintagmas. El tratamiento de ambas relaciones en los sistemas de recuperación, según Fagan (Fagan 1989), es el siguiente:

- *Relaciones semánticas entre los términos de un tesauro*: si los términos `A` y `B` son miembros de la misma entrada o notación del tesauro, `C`; entonces si `A` aparece en el texto de un documento, se asigna tanto `A` como `B`, y alternativamente `C`:

  si (*A o B*), *asigna* (*A y B*) *o C*

- *Relaciones sintácticas entre los términos de un sintagma*: si los términos `A` y `B` aparecen junto en un documento y forman parte de una ***relación de modificación***, uno respecto del otro, entonces se asigna el sintagma `AB`:

```
si (A y B), asigna AB
```

Las relaciones semánticas son de tipo *paradigmático* permitiendo ampliar los términos que se utilizan en la representación de los documentos, su rendimiento en los sistemas de recuperación es aumentar la *exhausitividad*. Por el contrario, las relaciones sintácticas son de tipo *sintagmático* permitiendo reducir los términos que se utilizan en la representación de los documentos, su rendimiento en los sistemas de recuperación es aumentar la *precisión* (Salton y McGill 1983). En este trabajo se van a tratar las relaciones sintagmáticas de los constituyentes que forman los sintagmas, en las que lo relevante es la estructura gramatical de los términos implicados, y no el significado. En relación con esto, muchos experimentos presentados en *TREC–6* (Harman 1997) proponen la combinación de términos en sintagmas para mejorar la eficacia de los sistemas de recuperación, aunque no se hayan obtenido los resultados deseados.

Las estructuras de los sintagmas están formadas por dos o más unidades consecutivas, y las relaciones entre estas unidades se codifican bajo la consideración de que los sintagmas son *construcciones endocéntricas*, o *modifier-head-constructions*, en las que todo el grupo de términos constituyentes se puede sustituir por un término del sintagma que se denomina el *núcleo de la construcción*. El interés está en identificar el *núcleo* de estas construcciones y en distinguir cuáles son los elementos *satélites* que lo modifican. Estos dos componentes juntos forman sintagmas que hacen referencia a conceptos más específicos –como en los sintagmas nominales bimembres «*document clustering*», «*Internet browsing*», o «*digital libraries*» en los que el primer elemento modifica al segundo–. Por lo tanto, es fundamental identificar el tipo de relación que mantienen los términos dentro de estas construcciones, con ese objetivo se han desarrollado diversos procedimientos que de forma general se agrupan en:

1. Técnicas no-lingüísticas: *métodos de identificación de estructuras basados en la co-ocurrencia de términos*.

2. Técnicas lingüísticas: *métodos de identificación de estructuras basados en la construcción de gramáticas*, representadas en *Máquinas de Estado-Finito*.

Los métodos no-lingüísticos para generar identificadores de sintagmas se basan en la asociación estadística de términos, o *co-ocurrencia de términos*, que están limitadas para captar la relación de modificación; mientras que la aplicación de métodos lingüísticos se basa en la identificación de la estructura de tales construcciones, por medio de reglas sintácticas, o gramáticas, que sí pueden captar la relación de modificación. En este último caso, se tienen que adoptar metodologías propias del *Procesamiento del Lenguaje Natural* (PLN).

Al problema de la identificación de estructuras se suma el de la posible agrupación de las distintas variantes estructurales de los sintagma, que haría necesario el uso de *algoritmos de conflación.* Mediante estos algoritmos se conseguiría fusionar estructuras sintagmáticas semánticamente equivalentes pero con estructuras sintácticas diferentes –como en el caso de las estructuras «*document representation*» / «*representation of document*»–. Bajo la hipótesis de que los *algoritmos de conflación* se pueden considerar *algoritmos de normalización* cuando se aplican con técnicas lingüísticas, vamos admitir que el análisis morfológico, léxico y sintáctico para los sistemas de RI puede ser el mismo que se utiliza en otros sistemas de procesamiento de texto, como es el de los compiladores de lenguajes de programación.

Desde la consideración anterior vamos a construir analizadores léxicos y sintácticos mediante formalismos de estado-finito que se van a encargar de identificar expresiones morfológicas, léxicas y sintácticas, a su vez como procedimiento de conflación en los tres casos vamos a emplear *Transductores de Estado-Finito*, con el propósito de transformar dichas expresiones en *formas canónicas*. El desarrollo de estas herramientas nos va a permitir reconocer, generar y controlar términos candidatos, que potencialmente continuarán su proceso para eventualmente ser añadidos al fichero de búsqueda. Por último, a las unidades obtenidas con este método se les podrá asignar un valor de ponderación, aunque no sea objeto de este trabajo, dada la eficacia que han demostrado los métodos estadísticos en los sistemas de RI.

Con el planteamiento anterior, en los siguientes apartados se va a realizar una revisión de los procedimientos más relevantes de reducción de variantes que se aplican en los sistemas de RI, proponiendo un modelo que incorpora conocimiento lingüístico, y que posteriormente vamos a desarrollar.

1.2. Procedimientos para la reducción de las variantes léxicas

Los procedimientos de reducción, o *fusión*, de variantes léxicas tienen como objetivo agrupar palabras similares a un término único, que puede ser un *stem*, o un *lema*, por medio de *algoritmos de conflación*, que tienen en cuenta los finales comunes de las palabras que pueden ser conflactadas. Los programas que realizan esta función se denominan: *a) programas de stemmer*, cuando este proceso se realiza con técnicas no-lingüísticas y algoritmos de *stemming*; y *b) programas lematizadores*, cuando este proceso se realiza con técnicas lingüísticas y algoritmos de *lematización*.

Un *algoritmo de stemming* se define como un procedimiento para reducir todas las palabras con el mismo *stem*, normalmente eliminando de cada palabra los *afijos derivacionales y flexionales* (Lovins 1968). El desarrollo de la conflación con técnicas de *stemming* consiste básicamente en eliminar de un término los afijos más largos posibles de acuerdo a un conjunto de reglas repitiéndose el proceso hasta que no se puedan suprimir más caracteres. Por esta razón, también se les denominan algoritmos de *stemming*, o de coincidencia más larga, *longest match*. Básicamente, las objeciones a este tipo de procedimientos son las siguientes:

- Las unidades que se obtienen no son lingüísticamente correctas, y esto reduce el nivel de comprensión de los índices.
- Eliminación de menos sufijos de los debidos impidiendo la *fusión*, o conflación, de términos relacionados, produciendo errores de *understemming*.

- Eliminación de más sufijos de los debidos dando lugar a que se unan términos que no están relacionados, produciendo errores de *overstemming*.

Los errores derivados de estas deficiencias se podrían atenuar utilizando técnicas lingüísticas por medio de *algoritmos de lematización,* definidos como un procedimiento para reducir los términos con el mismo *stem*, la misma categoría léxico-sintáctica y el mismo significado a una forma única. El desarrollo de la conflación con esta técnica consiste en eliminar de un término los afijos de acuerdo con un lexicón o diccionario. El lexicón se configura como una base de datos léxica que se emplea para realizar un análisis léxico de los términos de entrada, encargándose además de relacionarlos con una forma canónica, o *lema*. A su vez, la base de datos léxica se podría construir utilizando formalismos de estado-finito. Sin embargo, los problemas de este procedimiento son los siguientes:

- La creación de índices por medio de análisis léxico es muy costosa.
- Las irregularidades de las flexiones léxicas impiden en muchos casos que se realice una equiparación exacta entre las formas superficiales y las formas léxicas almacenadas en el lexicón. Para solucionar este problema se tienen que realizar una serie de transformaciones cuya representación es bastante compleja, como veremos.

A pesar de lo anterior, la mayor aportación de esta técnica es que los *algoritmos de lematización* se pueden proyectar como *algoritmos de normalización, o de control*. En el proceso de normalización un grupo de términos se reduce a una única forma canónica después de aplicar el programa de lematización. Habitualmente, el lematizador realiza un análisis morfológico que se encarga de vincular cada término superficial a una única forma normalizada, según las unidades especificadas en el diccionario. Además, la ventaja de este método es que las unidades obtenidas son lingüísticamente correctas y que, dependiendo del programa *lematizador*, a tales unidades se les asignan categorías POS, *part-of-speech,* que son imprescindibles para un posible procesamiento sintáctico posterior. Las formas normalizadas del diccionario se podrían definir como:

- **NOMBRES** → **masculino/singular**
- **ADJETIVOS** → **masculino/singular**
- **VERBOS** → **infinitivo**
- **DETERMINANTES** → **masculino/singular**
- **DEMOSTRATIVOS** → **masculino/singular**
- **CUANTIFICADORES** → **masculino/singular**
- **POSESIVOS** → **masculino/singular**
- /.../

En torno a las cuestiones planteadas, los siguientes apartados se van a dedicar a describir más en profundidad los procedimientos mencionados, siempre teniendo en cuenta que los métodos de reducción de variantes léxicas más utilizados en los sistemas de RI son los dependientes de un lenguaje y los que están diseñados para manejar variantes morfológicas con técnicas no-lingüísticas.

1.2.1. Técnicas no-lingüísticas

Los algoritmos de conflación con técnicas no-lingüísticas se denominan comúnmente *algoritmos de stemming*, o algoritmos de *eliminación de afijos*. Los principales algoritmos de *stemming* se aplican a la lengua inglesa y son los siguientes: *Algoritmo de Lovins* (Lovins 1968), *Algoritmo de Dawson* (Dawson 1974), *Algoritmo de Porter* (Porter 1980), *Algoritmo de Paice/Husk* (Paice 1990). El algoritmo que actúa con una reducción más agresiva es el *algoritmo de Lovins* según un estudio comparativo realizado por Harman (Harman 1991) entre tres modelos: un sencillo algoritmo denominado el *algoritmo de stemming "S"*, el *algoritmo de Lovins* y el *algoritmo de Porter*. El funcionamiento de un sencillo algoritmo de *stemming "S"* para la eliminación de plurales de los términos en inglés, según el estudio de Harman (Harman 1991), se expresa de la forma siguiente:

- Un conjunto de reglas que sólo se aplican a los términos con una determinada longitud, tres o más caracteres:

> **SI** *una palabra acaba en "ies" pero no es "eies" ni "aies"*
>
> **ENTONCES** `ies` → `y`
>
> **SI** *una palabra acaba en "es" pero no es "aes" "ees" o "oes"*
>
> **ENTONCES** `es` → `e`
>
> **SI** *una palabra acaba en "s" pero no es "us" ni "ss"*
>
> **ENTONCES** `s` → `NULL`

- Las reglas se aplican a las palabras con una *longitud suficiente*, tres o más caracteres, y de una forma dependiente, *la primera regla aplicable se usa sólo una vez.*
- Cada regla consta de tres partes: una *especificación*, que delimita el final de la palabra, una *lista de excepciones* y una *acción*.

A pesar de este sencillo procedimiento el *stem* resultante puede ser incorrecto porque no se especifica claramente cuál es la regla que se debe aplicar. Para resolver este problema el *stemmer de Lovins* se basa en la *equiparación mas larga* tomada de una lista bastante extensa de sufijos, mientras que el *stemmer de Porter* se basa en un algoritmo con un número muy pequeño de sufijos y unas pocas reglas de *reescritura* que tienen en cuenta el contexto de aparición de sufijos para su eliminación. El *algoritmo de Porter* se basa en dos planteamientos básicos: que el término *reducido* mantenga una determinada longitud y que el sufijo eliminado sea siempre el más largo. Además, para realizar la conflación se tienen que dar una serie de condiciones –*condiciones del stem* y *condiciones del sufijo*– y un conjunto de reglas de reescritura que se dividen en cinco pasos –*1a, 1b, 1c, 2, 3, 5a, 5b*–. Las condiciones y las reglas se expresan del modo siguiente:

1. Condiciones
   a. Condiciones del *stem*
      - La medida de un *stem*, `m`, se basa en sus secuencias alternativas *vocal consonante* y se define como:

$[C](VC)^m[V]$

los corchetes indican el carácter opcional de las ocurrencias, donde

C *es una secuencia de consonantes*

V *es una secuencia de vocales*

m *es el número de secuencias* **VC**

Por ejemplo,

m = 0	tie, the, by
m = 1	greats, green, perm
m = 2	deduct, labor, other

- ***S** el *stem* acaba en s (o en otras letra similares).
- ***v*** el *stem* contiene una vocal.
- ***d** el *stem* acaba en doble consonante.
- ***o** el *stem* acaba en una secuencia *consonante-vocal-consonante*, donde la consonante final no es w, x ni y.
- Las condiciones pueden contener expresiones con **and**, **or** y **not**.

b. Condiciones del sufijo

- Indica cuál es el sufijo actual

2. Acciones de las reglas

a. Adoptan la forma de reglas de reescritura según las cuales si un término acaba en S_1 se reemplaza por S_2, si cumple unas determinadas condiciones:

$S_1 \rightarrow S_2$

donde

S_1 es el sufijo actual de un término

S_2 es el sufijo nuevo

b. Las reglas se dividen en pasos que definen el orden secuencial en que se aplican, y sólo puede aplicarse una de las reglas del paso:

- Reglas del paso 1a : *plurales*

Condiciones del *stem*	Condiciones del sufijo	Regla
Ninguna	sses	sses → ss
Ninguna	ies	ies → i
Ninguna	ss	ss → ss
Ninguna	Ninguna	s → NULL

- Reglas del paso 1b: *gerundios y participios*

Condiciones del *stem*	Condiciones del sufijo	Regla
$m > 0$	eed	eed → ee
$^{*}v^{*}$	ed	ed → NULL
$^{*}v^{*}$	ing	ing → NULL

Si las reglas anteriores se han ejecutado con éxito, se aplican las siguientes reglas con el objetivo de modificar los *stems* procedentes de gerundios y participios:

Condiciones de *stem*	Cond. del sufijo	Regla
Ninguna	at	at → ate
Ninguna	bl	bl → ble
Ninguna	iz	iz → ize
$(^{*}\text{d and not}(^{*}\text{L or } ^{*}\text{S or } ^{*}\text{Z}))$	Ninguna	→ letra única
$(m = 1 \text{ and } ^{*}o)$	Ninguna	→ e

- Reglas del paso 1c*: sustitución de* **y** *final por* **i**

Condiciones del *stem*	Condiciones del sufijo	Regla
$^*v^*$	y	$y \rightarrow i$

- Reglas de paso 2: *palabras derivadas*

Condiciones del *stem*	Condiciones del sufijo	Regla
$(m > 0)$	ational	$ational \rightarrow ate$
$(m > 0)$	tional	$tional \rightarrow tion$
\....\	\...\	\...\

- Reglas de paso 3: *palabras derivadas*

Condiciones del *stem*	Condiciones del sufijo	Regla
$(m > 0)$	icate	$icate \rightarrow ic$
$(m > 0)$	ative	$ative \rightarrow NULL$
\....\	\...\	\...\

- Reglas de paso 4: *palabras derivadas*

Condiciones del *stem*	Condiciones del sufijo	Regla
$(m > 1)$	al	$al \rightarrow NULL$
$(m > 1)$	ance	$ance \rightarrow NULL$
\....\	\...\	\...\

- Reglas de paso 5a: *eliminación de sufijo* **e**

Condiciones del *stem*	Condiciones del sufijo	Regla
$(m > 1)$	e	$e \rightarrow NULL$
$(m = 1 \text{ and not } ^*o)$	e	$e \rightarrow NULL$

- Reglas del paso 5b*: doble* **l** *final*

Condiciones del *stem*	Condiciones del sufijo	Regla

$(m > 1 \text{ and } {}^{*}d \text{ and } {}^{*}L)$	Ninguno	$\rightarrow$ letra única

Aunque el *stemmer* de Porter mejora la recuperación en términos de exhaustividad (Porter 1980), y se adapta con éxito a otras lenguas, tiene una serie de problemas que aparecen en mayor o menor grado en otros procedimientos de *stemming*: *a)* es difícil de comprender y *modificar*; *b)* incurre en errores, unas veces por una *excesiva conflación* y otras veces por *falta de conflación*; y *c)* produce *stems* que no son auténticas palabras, por lo que suelen ser difíciles de interpretar para un usuario final (Xu y Croft 1998).

Además de los problemas anteriores, los errores de los algoritmos de *stemming* suelen estar relacionados con el hecho de que realizan agrupaciones incorrectas. Estos problemas surgen porque la mayoría de los *stemmer operan sin un lexicón e ignoran el significado de los términos* (Krovetz 1993). Un error habitual del *stemmer de Porter* es que agrupa términos como «*general*», «*generous*», «*generation*», y «*generic*» bajo el mismo *stem*, mientras que términos relacionados como «*recognize*» y «*recognition*» no los agruparía jamás. Para solucionar este problema, el procedimiento de conflación según el *stemmer de Krovetz*, también denominado *KSTEM*, se basa en diccionarios automatizados y reglas bien definidas para la morfología flexional y derivacional. En el caso de los afijos flexivos, el algoritmo de Krovetz funciona de la siguiente forma:

- Convierte los plurales en singulares, según las reglas

  ies $\rightarrow$ y
  es $\rightarrow$ s
  s $\rightarrow$ NULL

- Convierte los participios a presente y elimina la forma del gerundio, según las reglas:

  ed $\rightarrow$ NULL
  ing $\rightarrow$ NULL

- El resultado se comprueba, finalmente, con las *entradas del diccionario*

Sin embargo, aunque el *stemmer de Krovetz* resuelve algunos errores de conflación no produce mejores resultados que el *stemmer de Porter* en términos de *exhaustividad/precisión*, una de las razones de su falta de eficacia es que depende excesivamente de las entradas del diccionario, lo que provoca que la conflación sea demasiado *conservativa* (Xu y Croft 1998). Todos estos problemas se pueden resolver con otros procedimientos de conflación basados en el análisis léxico, por medio de diccionarios o lexiconesen lugar de reglas, que describiremos en el apartado siguiente.

1.2.2. Técnicas lingüísticas

Las técnicas de *stemming* basadas en análisis morfológico se presenta en un analizador léxico desarrollado por un grupo de lingüistas computacionales de Xerox, *Multi-Lingual Theory and Technology Group* (MLTT). Una de las mayores aplicaciones de esta herramienta es el *parsing* morfológico, que se puede utilizar para reducir las variantes léxicas en los sistemas de RI. El análisis de las formas flexionadas de los términos se realiza por medio de un lexicón representado por medio de formalismos de estado-finito.

El analizador de Xerox se basa en el *modelo de análisis morfológico de dos-niveles* propuesto por Kimmo Koskenniemi (Koskenniemi 1983). Este modelo pretende demostrar que cualquier unidad léxica se puede representar como una correspondencia entre una *Forma Léxica*, o subyacente, y una *Forma Superficial*:

Forma léxica: m a t r i x λ + s
Forma superficial: m a t r i c e s

El modelo de Koskenniemi tuvo su implementación computacional en el analizador léxico desarrollado por Karttunen denominado PC-KIMMO (Karttunen 1983), que fue el germen del analizador morfológico de Xerox. La primera versión del *parser* KIMMO consigue

descomponer palabras por medio de la integración de dos módulos analíticos: por un lado, un componente basado en reglas de dos-niveles y, por otro, un componente léxico que incluye una lista de morfemas tanto *stems* como afijos (Fig. 1.1). Sin embargo, esta primera versión no proporciona las categorías léxicas *part-of-speech* (POS), de la palabras y, por esta razón, no constituye un analizador adecuado para su aplicación a un *parser* sintáctico posterior, que precisa como *input* el texto previamente etiquetado con las categorías léxicas correspondientes. Esta limitación se corrige en una segunda versión del PC-KIMMO que incorpora las categorías morfológicas como parte del lexicón.

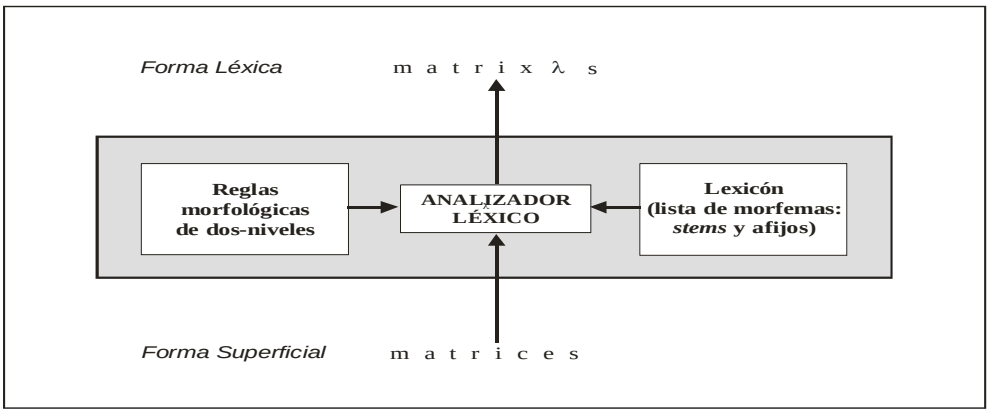


Fig. 1.1: Componentes del analizador léxico PC-KIMMO

En PC-KIMMO una palabra superficial de entrada se analiza, según el lexicón y las reglas, en estructuras de secuencias de morfemas. La estructura de secuencias de morfemas está compuesta por: *a)* una *forma léxica*, *b) comentarios* o *gloss*, *c) categorías léxicas* y, *d) características estructurales*. El resultado de este análisis de morfemas se transfiere una *gramática léxica*, que da lugar a un *árbol de análisis* en el que se señalan las característica *part-of-speech* (POS), de las palabras (Antworth 1995). Así, según un ejemplo tomado de Antworth, la palabra «*enlargements*» quedaría dividida en la siguiente estructura de morfemas y en el correspondiente *árbol de análisis* (Fig. 1.2):

```
Form:  en+            `large          +ment          +s
Gloss: VR1+           `large          +NR25          +PL
Cat:   PREFIX         AJ              SUFFIX         INFL
Feat:  [from_pos:AJ   [head: [pos:AJ]] [from_pos:V    [from_pos:N
        head: [pos:V]]                  head: [pos:N]]  head:  [number:PL

                                                          pos:   N ]]
```

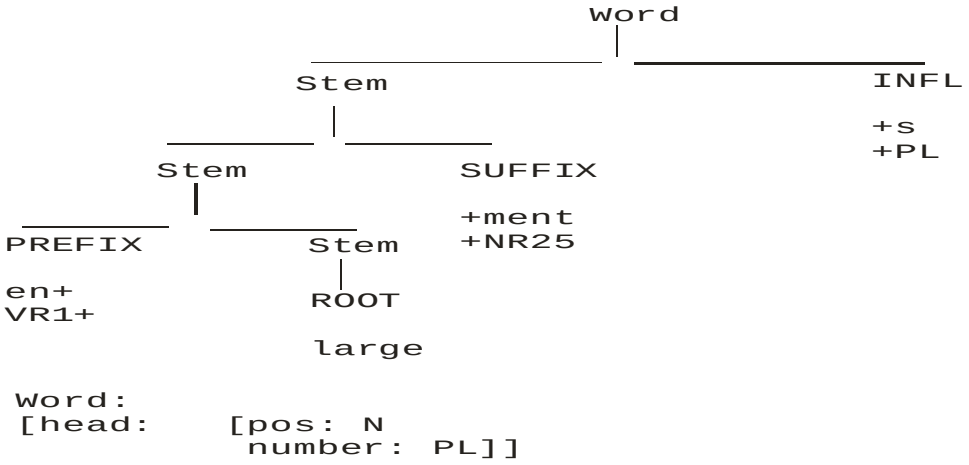


Fig. 1.2: Árbol de análisis léxico

De esta forma, el resultado del análisis léxico reside en la asignación de las distintas categorías léxico-gramaticales a las unidades léxicas, además de la incorporación de propiedades sintácticas de concordancia (como *género*, *número*, o *persona*) y de información morfológica (como puede ser el patrón de formación de la palabra). La característica estructural *POS* del caso anterior es *Nombre* y la característica de número es *Plural*, esta información es la que se transmitirá posteriormente al *parser* sintáctico.

Con el planteamiento anterior, la base del analizador morfológico de Xerox reside en que entre las formas superficiales de un lenguaje y sus correspondientes *lemas* se establece una *Relación Regular*, definida por una *Expresión Regular* susceptible de ser compilada en un *Transductor de Estado-Finito* (Karttunen 1994). Las formas léxicas se almacenan en un diccionario, representado en un transductor, en el que cada lema tiene asignado las

propiedades morfológicas correspondientes a esa forma. En el proceso de reconocimiento, el transductor sigue un *path*, o secuencia de estados y arcos, desde un estado inicial a un estado final, dando como resultado que una forma superficial se equipare a una forma léxica (Fig. 1.3):

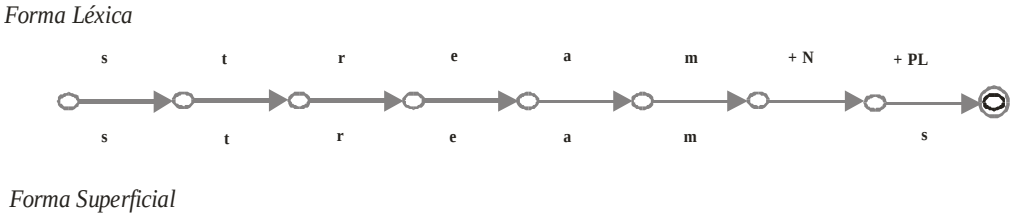


Fig. 1.3: Equiparación entre formas superficiales y formas léxicas

La reducción de variantes léxicas con esta técnica presenta el siguiente problema: la correspondencia entre formas superficiales y léxicas se complica cuando se producen alteraciones morfológicas, dependientes del contexto, que dificultan la equiparación entre ambas formas, como en `matrices→matrix+N+PL`. Para solucionar este problema el analizador léxico de Xerox construye dos componentes o módulos (Karttunen 1994):

1. Un *lexicón* representado en un formalismo de estado finito que define el conjunto de formas válidas de un lenguaje.
2. Un *conjunto de reglas* representadas en un formalismo de estado-finito que se encargan de asignar a las formas superficiales las formas léxicas correspondientes así como las categorías POS.

El formalismo en el que se representan los dos componentes es un transductor con miles de estados y un número infinito de *paths*. La especificación metodológica de este procedimiento se realizará en otro capítulo, cuando se haya expuesto el funcionamiento de las técnicas de

estado-finito. De cualquier forma, a lo largo de este trabajo se va a proponer y desarrollar un modelo lingüístico menos complejo que se adapta a las necesidades concretas de los sistemas de recuperación.

1.3. Procedimientos para la reducción de las variantes sintácticas

El reconocimiento de estructuras complejas representadas en sintagmas se considera un indicador del contenido de los documentos mejor que los términos usados aisladamente, por esta razón son muchos los métodos que se han desarrollado para su identificación. Básicamente, hay dos aproximaciones diferenciadas, métodos no-lingüísticos y métodos lingüísticos, que dan lugar a los denominados comúnmente *sintagmas estadísticos –statistical phrases–* y *sintagmas sintácticos –syntactic phrases–*.

La identificación de sintagmas con técnicas estadísticas se basa en la *co-ocurrencia* de los términos que componen estas construcciones, tales como el cálculo de la frecuencia de determinados pares de términos adyacentes. Para la producción de sintagmas estadísticos es preciso pre-procesar un texto con el objetivo de obtener un diccionario de sintagmas, o *lexicon phrasal*, definido como lista de *phrases* que aparecen con una determinada frecuencia (Fagan 1989). La indización subsiguiente de los documentos se basa en la identificación de los sintagmas a partir de ese *diccionario.* Sin embargo, los procedimientos estadísticos tienen los siguientes inconvenientes (Salton y McGill 1983):

- Los sintagmas seleccionados pueden ser *estadísticamente significativos* pero *sintácticamente incorrectos.*
- La falta de control en la selección de sintagmas puede producir inconsistencias que reducen el rendimiento de los sistemas de RI.

Con el objetivo de solucionar los problemas anteriores sería preciso aplicar métodos lingüísticos que fueran capaces de identificar las estructuras sintácticas de estas construcciones y establecer algún tipo de control en la selección de sintagmas. Con este objetivo, sería preciso emplear métodos pertenecientes al PLN. Sin embargo, para que la aplicación de las técnicas del PLN a la RI sea efectiva se deberían cumplir una serie de condiciones: *1) disponibilidad para procesar gran cantidad de textos*; *2) disponibilidad para procesar textos sin restricciones*, en los que pueden aparecer palabras desconocidas, nombres propios, o errores de trascripción; *3) representación parcial y superficial del contenido de los textos* (Evans y Zhai 1996).

Las condiciones anteriores contribuyen a una *relajación* de las técnicas del PLN cuando se aplican a los sistemas de RI, provocando que el PLN se vuelva relativamente sencillo, porque aunque los analizadores léxicos y sintácticos intervengan en los textos sin restricciones de las bases de datos, no es preciso un análisis completo y en profundidad del contenido de los documentos sino que es suficiente un *análisis superficial de determinados fragmentos*. Teniendo en cuenta los pre-requisitos anteriores, la aplicación del PLN a los textos se desarrolla en una secuencia de procesos cuya finalidad es reconocer y anotar las distintas unidades del nivel de análisis correspondiente:

1. *Etiquetado morfosintáctico.*
2. *Lematización*, o *normalización y control morfológico*, basada en un lexicón.
3. *Desambiguación* de etiquetas POS.
4. *Análisis sintáctico*, necesario para identificar la estructura de los sintagmas.
5. *Normalización y control de estructuras sintácticas.*

La complejidad de cada proceso hace que los lingüísticas computacionales implementen cada tarea en módulos separados, en lo que se denomina comúnmente *arquitectura* o *ingeniería lingüística* del sistema. De entre las herramientas disponibles de propósito general para realizar cada etapa del proceso de análisis se podrían utilizar las siguientes:

- Para el etiquetado morfosintáctico se podría utilizar el *etiquetador de categorías basado en reglas de Brill* (Brill 1992). De forma más específica, para el tratamiento morfosintáctico del idioma español se ha desarrollado el etiquetador *SPOST* (Farwell et al. 1995), el etiquetador *SMORPH* (Aït-Mokhar y Rodrigo Mateos 1995), y el etiquetador *MACO+* elaborado por la *Universidad Politécnica de Cataluña* (Márquez y Padró 1997).
- Para el proceso de lematización se puede utilizar el *analizador morfológico de Xerox* (Karttunen 1983) (Karttunen et al. 1992). De entre las herramientas específicas para la lematización de idioma español se encuentra el programa *MACO+* (Márquez y Padró 1997) y la aplicación desarrollada por Sánchez-León (Sánchez-León 1995).
- Para la desambiguación de etiquetas se pueden utilizar modelos simbólicos basados en reglas, o modelos estocásticos basados en métodos estadísticos como el *Modelo Oculto de Markov* (Cutting et al. 1992)
- Para el análisis sintáctico se puede utilizar el analizador de sintagmas *NPtool* (Voutilainen 1997), o el analizador *AZ Noun Phraser* elaborado en el *Laboratorio de Inteligencia Artificial de la Universidad de Arizona*. Una herramienta específica para el análisis sintáctico del idioma español es el analizador IFSP, *Incremental Finite-State Parsing*, elaborado por Gala (Gala 1999).

Dado que las herramientas anteriores proporcionan un análisis parcial sobre los distintos fenómenos lingüísticos, además de que en muchos casos se trata de analizadores de carácter general para todo tipo de aplicaciones lingüísticas, nuestro objetivo va a estar dirigido a desarrollar un modelo que integre todos los componentes y que sea especialmente relevante para las necesidades de los sistemas de RI, restringiendo el análisis sintáctico a los *Sintagmas Nominales*.

De cualquier forma, tanto si se utilizan recursos lingüísticos de propósito general, como si se desarrolla un modelo específico, sigue existiendo el problema del reconocimiento de las variantes sintácticas, en este caso SSNN estructuralmente distintos pero semánticamente equivalentes. La solución a este problema es construir programas capaces de reducir todas las variantes a formas canónicas, o *formas normalizadas* en las que cada sintagma tenga asignado

un *papel bien definido que refleje la complejidad de la estructura sintáctica* (Salton y McGill 1983).

Una aplicación en la cual se adoptan métodos lingüísticos para normalizar las estructuras de los *Sintagmas Nominales* la podemos encontrar en el sistema IRENA (*Information Retrieval Engine Based Natural Language Analysis*) (Arampatzis et al. 1997). En el proceso de normalización con este sistema cualquier sintagmas se equipara a una estructura, *phrase frame*, compuesta por un núcleo y distintos modificadores, *head-modifier*, `PF = [h, m]`.

Un proceso de control de estructuras parecido al anterior se propone en el sistema desarrollado por Strzalkowski (Strzalkowski et al. 1999), en el cual las estructuras de los sintagmas se reducen a la cadena normalizada `núcleo + modificador`, *head + modifier pairs stream*, El procedimiento propuesto por Stzalkowski se basa en un sistema de RI que utiliza técnicas estadísticas ampliadas con un modulo de PLN, cuya función es reducir los documentos a colecciones de *pares-de-palabras*, *núcleo-modificador*, por medio de un análisis sintáctico. El objetivo de la aplicación de técnicas lingüísticas es normalizar estructuras sintácticas capturando la uniformidad semántica a través de las variantes de las formas superficiales.

Otro procedimiento para la regularización de estructuras sintácticas se desarrolla en el sistema CLARIT (Evans et al. 1996). La técnica que utiliza este sistema consiste en generar un *lexicon phrasal*, de forma similar al que se usa en los métodos estadísticos, pero en este caso por medio de análisis lingüístico, y no basado en la co-ocurrencia de términos. Una vez que el texto de los documentos se pre-procesa para generar la lista de *phrases*, se iniciaría la indización de documentos por medio de análisis lingüístico para identificar posibles sintagmas candidatos. Cuando tales sintagmas se identifican, se busca en el *lexicon phrasal* y si existen, por medio de una equiparación exacta, entonces los sintagmas seleccionados se usan para indizar el documento (Evans y Zhai 1996). Si los sintagmas del documento no existen en el lexicón pero se encuentran algunos de sus constituyentes, entonces el documento se indiza por los sintagmas constituyentes. De esta forma, la clave del sistema CLARIT para

controlar las estructuras sintácticas consiste en indizar documentos sólo por los sintagmas existentes en el *lexicon phrasal*, y de esta forma se logra llevar a cabo el proceso de control.

En este trabajo vamos a proponer una alternativa a los procedimientos anteriores basada en técnicas lingüísticas que nos va a permitir equiparar *estructuras sintácticas superficiales* a *estructuras sintácticas controladas*. Además, con el desarrollo de esta aplicación se superaría una de las más antiguas y mayores dificultades entorno al reconocimiento de las variantes sintácticas como es: *la necesidad de almacenar y manipular las miles de variantes que pueden tener los sintagmas*, provocando que su identificación se vuelva *impracticable* (Salton 1989). La aplicación que vamos a desarrollar aquí se va a basar en realizar el proceso de control de estructuras sintácticas por medio de *transductores*.

1.3.1. Técnicas no-lingüísticas

La generación de sintagmas con técnicas no-lingüísticas consiste en la asociación estadística de los términos para crear identificadores específicos del contenido de los documentos que mejoren los índices de *precisión* de los sistemas de RI. Los sistemas que usan estas técnicas construyen SSNN por medio de la co-ocurrencia de dos o más términos constituyentes, *two-term phrases*. Para establecer las relaciones entre los dos términos se usan las propiedades de frecuencia de aparición de cada término particular con el objetivo de que la frecuencia de aparición del sintagma resultante sea más pequeña que las de sus componentes individuales y, por lo tanto, más específica.

Básicamente los métodos no-lingüísticos de *generación-de-sintagmas* utilizan la frecuencia de dos o más términos particulares, $TERM_k$ y $TERM_h$, y la sustituyen por un sintagma, $PHRASE_{kh}$. Sin embargo, como los sintagmas pueden incluir términos cuya frecuencia en la colección es mayor de lo esperado, debido a la frecuencia de los términos individuales, se aplica la siguiente fórmula donde la cohesión del *par-de-términos* de define como (Salton y McGill 1983):

$$COHESION_{kh} = SIZE \cdot FACTOR \cdot \frac{PAIR - FREQ_{kh}}{TOTFREQ_k \cdot TOTFREQ_h}$$

donde

- $SIZE \cdot FACTOR$ representa un factor relacionado con el tamaño del vocabulario de indización.
- $PAIR\text{-}FREQ_{kh}$ es la frecuencia total de $TERM_k$ y $TERM_h$ en la colección.
- $TOTFREQ_k$ y $TOTFREQ_h$ representa la frecuencia de los términos individuales en la colección, y se obtiene según:

    - La frecuencia del término k en el documento i, $FREQ_{ik}$.
    - La frecuencia total de cada término individual en la colección, obtenida a partir de la suma de las frecuencias de cada término a través de todos los n documentos, $TOTFREQ_k = \sum_{i=1}^{n} FREQ_{ik}$

La aplicación de la formula anterior permite seleccionar sintagmas con pares de términos con una cohesión suficientemente alta, sin embargo se deben establecer restricciones relacionadas con el contexto en el que los dos términos co-ocurren. Las restricciones del contexto permiten mejorar los resultados de la construcción de sintagmas, reduciendo el número de sintagmas generados con este método, lo que se traduce en un aumento de la precisión. Estas restricciones pueden imponer que el contexto de los dos términos sea: *1)* el *mismo documento* y *las mismas frases* de esos documentos particulares, *2) las mismas frases* pero con al *menos* k *palabras entre los componentes del sintagma; 3) las mismas frases* y en *posiciones de palabras adyacentes*; y *4) las mismas frases* y en *posiciones de palabras adyacentes en el orden de palabra correcto* (Salton y McGill 1983).

El proceso de formación de sintagmas estadísticos no se puede desvincular del proceso de indización. Como ya se ha mencionado de forma simplificada, la indización por medio de

sintagmas consiste en pre-procesar un corpus para construir un diccionario de sintagmas, o *lexicon phrasal*, definido como una lista de sintagmas que aparecen con una determinada frecuencia. Posteriormente, la identificación de los sintagmas potenciales de los documentos consistiría en buscar tales sintagmas en el *lexicon phrasal* y comprobar si aparecen con una determinada frecuencia y si es así, asignar a los documentos tales sintagmas.

En el proceso de formación de sintagmas se utilizan dos medidas: *a)* la *frecuencia inversa del documento*; y *b)* los *valores de discriminación del término*, o *los términos*, en una colección de documentos. Con estos dos parámetros se podrían identificar tres clases de términos (Salton y McGill 1983):

- *Términos de frecuencia media*, con valor de discriminación positivo, que se usan como términos de índice sin transformación.
- *Términos de frecuencia alta*, con valor de discriminación negativo, que se eliminan o se incorporan al proceso de formación de sintagmas.
- *Términos de frecuencia baja*, con valor de discriminación cero, que se incorporan a un tesauro.

Usando estas frecuencias, el componente principal del sintagma, *núcleo de sintagma* o *phrase head*, podría ser un término cuya frecuencia en el documento tuviera un valor de discriminación negativo. Los otros componentes del sintagma podrían ser términos con una frecuencia media o baja, definidos a partir de una relación de co-ocurrencia con el núcleo de esta construcción, los componentes satélites podrían co-ocurrir en distintos contextos como podrían ser de *forma adyacente*, o dentro de *la misma sentencia* (Salton 1989). Sin embargo, el proceso de formación de sintagmas controlado sólo por la *co-ocurrencia de términos no genera sintagmas de calidad*, por eso es necesario un mayor control de la formación de sintagmas por medio de la utilización de *criterios sintácticos* (Salton 1989). Todo esto conduce a la necesidad de desarrollar métodos pertenecientes al PLN.

Una combinación de métodos estadísticos y lingüísticos se presenta en el sistema *XTRACT* (Smadja 1993). Esta aplicación se basa en localizar en los textos la combinación recurrente de determinadas palabras, como «*free text*», «*user interface*», o «*source code*». Este tipo de combinaciones constituyen lo que se denomina *colocaciones*, o *collocations*. Las propiedades de estas construcciones es que son: *a) arbitrarias*, *b) dependientes de un dominio, c) recurrentes*, y *d) constituyen clusters cohesivos de palabras*, es decir, la presencia de una palabra implica el resto (Smadja y McKeown 1990). Estas propiedades hacen que se puedan utilizar estadísticas de co-aparición para su localización. La entrada al sistema *Xtract* es una palabra simple para la cual se quieren encontrar *colocaciones,* con este objetivo se desarrollan tres etapas generales:

1. La primera etapa consiste en extraer pares de palabras, o *bigramas*. Cada uno de los *bi-gramas* se forma por la búsqueda de una palabra de entrada y unas cinco palabras que aparecen junto a ella, en esta etapa se utilizan *métodos estadísticos,* que se encargan de medir la distancia más probable entre las palabras.
2. La segunda etapa consiste en utilizar los *bigramas*, identificados en la primera etapa, para encontrar *colocaciones* de más de dos palabras, o *n-gramas,* y se retienen la palabras que ocupan posiciones con una probabilidad mayor según un determinado umbral.
3. La tercera etapa consiste en añadir *información sintáctica* a las *colocaciones,* para ello todas las sentencias del *corpus* que contienen las dos palabras en esa posición se etiquetan con categorías *part-of-speech* (POS), por medio de un etiquetador estocástico. A continuación un *parser* sintáctico intenta identificar determinadas estructuras sintácticas y producir *relaciones sintácticas binarias* tales como «*Verbo-Objeto*» (V O), «*Nombre-Adjetivo*» (N A), o «*Nombre-Nombre*» (N N). Por último, se cuenta la frecuencia de cada relación binaria para el *bi-grama* y se realiza un análisis estadístico de esta distribución, al final de esta etapa una *colocación* se aceptaría si las dos palabras buscadas se usan con *la misma relación sintáctica*.

El sistema *Xtract* utiliza un procedimiento híbrido entre técnicas estadísticas y *parsing*, pero las técnicas lingüísticas que aplica se limitan a determinar la relación sintáctica de dos

palabras, (V O) (N A) (N N), y asignar esa identificación a las sentencias. Por lo tanto, el análisis que realiza este sistema se basa fundamentalmente en la distribución estadística de los términos y no en un auténtico análisis lingüístico por medio de gramáticas, como sucede en los sistemas que se describen en el siguiente apartado.

1.3.2. Técnicas lingüísticas

El reconocimiento sintagmas con técnicas lingüísticas requiere el desarrollo de gramáticas formales que especifiquen de forma explícita la estructura sintáctica de estas construcciones. La clasificación de los formalismos gramaticales más extendida es la establecida en la *jerarquía de Chomsky* (Chomsky 1957), en la que las gramáticas se definen básicamente según su poder para generar las distintas construcciones lingüísticas de una lengua. De forma simplificada, el poder de las gramáticas se sitúa entre las menos expresivas, como son las *Gramáticas Regulares*, a las más expresivas, como son las *Gramáticas Sintagmáticas*. Los formalismos gramaticales son necesarios para el análisis de las construcciones sintagmáticas, porque los programas de análisis sintáctico tienen como función aceptar como entrada cadenas de términos con del objetivo de producir como salida una representación estructurada de tales construcciones, este proceso lo desarrollan a partir de lexicones y gramáticas electrónicas..

La restricción del objeto de análisis de los sistemas de RI a los *Sintagmas Nominales* hace que sean suficientes formalismos débiles y poco expresivos, desarrollados con herramientas de análisis simples. Este tipo de sintagmas se generan con Gramáticas Regulares y se reconocen con *parser* superficiales, basados en mecanismos de estado-finito, como describiremos de forma más extensa en otro capítulo.

Las técnicas de estado-finito aplicadas al análisis sintáctico han dado lugar a las *Redes de Transición* (RT) y a las *Redes de Transición Recursivas* (RTN). Una Red de Transición se

configura como una red de nodos y arcos etiquetados con símbolos terminales: palabras o categorías léxicas. Cada transición, o *path*, entre los nodos se produce si la cadena de entrada pertenece a la categoría con la que está etiquetado el arco. Muchas veces una palabra puede tener más de una categoría, con lo cual de un nodo partiría más de un arco, en estos casos se habla de *no-determinismo*, este problema se soluciona con algoritmos de análisis que incluyen mecanismos de *backtracking*, de *vuelta a atrás*, provocando que se retorne al estado anterior, o transformado el autómata no-determinista en otro determinista, como se describirá en otro capítulo. Este fenómeno se produce porque el análisis sintáctico con esta técnica constituye un proceso de búsqueda en el que se explotan todas las posibles interpretaciones hasta que se cumplan las restricciones que impone la gramática. De esta forma, los mayores inconvenientes que presentan todos los formalismos basados en técnicas de estado-finito es la redundancia en la ejecución de determinadas operaciones, y el consiguiente gasto de tiempo en almacenar la descripción de los estados en cada punto de la elección, debido a que con frecuencia el procesador sintáctico analiza el mismo constituyente varias veces y si detecta en algún momento del recorrido que la elección es incorrecta utiliza un *algoritmo de backtracking o retrotrazado* (Grishman 1986) que deshace todo lo que se ha hecho después de la elección incorrecta.

Un analizador sintáctico basado en las citadas Redes de Transición se simbolizqa en una red conectada de *nodos*, representados por estados en los que al menos uno se implanta como estado inicial y otro como estado final, y *arcos*, representados por transiciones (Fig. 1.4). Las entradas a las RT son las etiquetas obtenidas en un análisis previo por medio del lexicón. En el proceso de análisis los arcos están etiquetados con nombres de categorías léxicas y una cadena de caracteres se acepta si las transiciones tiene una categoría igual a la etiqueta del arco. Con este procedimiento, el reconocimiento de cadenas comienza a partir del estado inicial y la red es capaz de reconocer una cadena si al transitar por todos los estados se llega a un estado final. Por el contrario, si en el recorrido aparece alguna transición que no sea posible, entonces la cadena no se acepta, o se considera incorrecta.

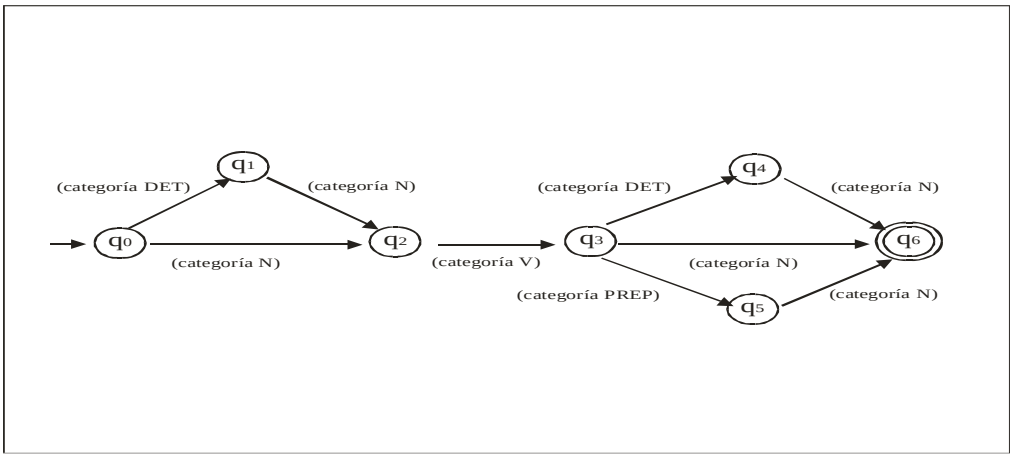


Fig. 1.4: Analizador sintáctico basado en una Red de Transición (RT)

Otro formalismo que tiene también su origen en los Autómatas Finitos son las denominadas *Redes de Transición Aumentadas*, *Augmented Transition Network* (*ATN*), (Woods 1970). Este modelo fue desarrollado como una interfaz para el acceso en lenguaje natural a una base de datos y cuyo mecanismo se ha usado en muchos sistemas de compresión del lenguaje. Las ATN constituyen un sistema de computación que incorpora varias clases de conocimiento añadiendo distintas operaciones en los arcos como: *condiciones* que filtran la transición entre estados, *llamadas* a otras redes que reconozcan los componentes de una oración o llamadas a distintos procedimientos que construyen estructuras que formarán parte del análisis final. Se trata por tanto de un formalismo de análisis sintáctico que procede de la Inteligencia Artificial diseñado para simular tareas cognitivas complejas, de ahí que, por un lado *reproduzcan las estrategias y expectativas de los usuarios lingüísticos* y, por otro, expresen las reglas gramaticales a modo de *procedimientos de utilización* de esas reglas (Rumelhart 1977).

A pesar de ciertas semejanzas entre este formalismo y los Autómatas Finitos, las ATN no son intrínsecamente un analizador sintáctico sino un sistema basado en reglas que intenta realizar simulaciones del procesamiento lingüístico mediante ordenador, además las reglas que representan pertenece un formalismo gramatical más complejo que las Gramáticas

Regulares, como es la *Gramática Independiente del Contexto*, por esta razón no se van a tratar en este trabajo.

Siguiendo con las RT, se puede decir que este formalismo equivale a un Autómata de Estado-Finito, que es el formalismo que vamos a utilizar para reconocer las estructuras de los sintagmas. Para ello las estructuras de estas construcciones de describen por medio de patrones sintácticos, o *Expresiones Regulares* definidas como un metalenguaje para la identificación de las estructuras que generan las Gramáticas Regulares, que a su vez reconocen los Autómatas de Estado-Finito. Según esto, la identificación de las estructuras de un grupo de sintagmas se podría determinar a partir de la especificación de Expresiones Regulares como:

```
N
N N
A N
DET N
DEM N
CUANT DET ORD N
CUANT DET CARD N
...
```

La RT que reconocería estas estructuras se puede representar en un diagrama de transiciones (Fig. 1.5), o en una tabla de transiciones (Fig. 1.6):

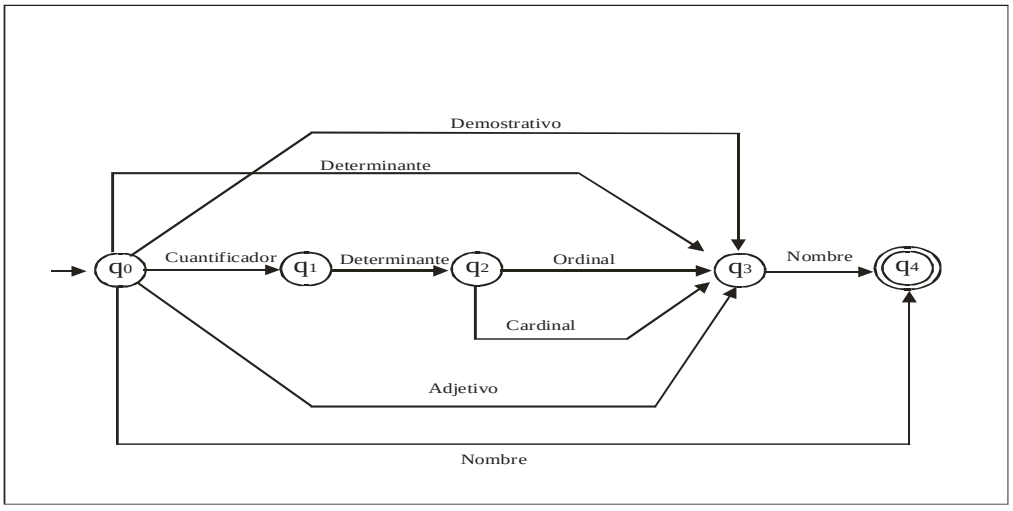


Fig. 1.5: Red de Transición representada en un diagrama de transiciones

	DET	DEM	CUANT	ORD	CARD	A	N
0	3	3	1	0	0	3	4
1	2	0	0	0	0	0	0
2	0	0	0	3	3	0	0
3	0	0	0	0	0	0	4
4	0	0	0	0	0	0	0

Fig. 1.6: Red de Transición representada en una tabla de transiciones

No obstante, y a pesar de la simplicidad de los formalismos de estado-finito, los métodos lingüísticos para el reconocimiento de estructuras sintagmáticas plantean al menos los siguientes problemas:

- La *ambigüedad estructural*, según la cual una misma construcción se puede analizar con distintos patrones sintácticos, todos ellos correctos según la gramática que se utilice, provocando un fenómeno de *sobreanálisis*.
- La *falta de cobertura*, o *infraanálisis*, según la cual los formalismos gramaticales sólo pueden reconocer las combinaciones que están especificadas en las reglas de la gramática.
- La *determinación del tipo de relación* que mantienen los constituyentes de los sintagmas nominales que no sea la simple yuxtaposición de componentes.

Todo esto vendría a confirmar que los métodos lingüísticos constituyen modelos insuficientes para el reconocimiento de los sintagmas de indización en los sistemas de RI. Sin embargo los procedimientos lingüísticos pueden ser efectivos si se restringe el análisis a *representaciones canónicas consistentes en formas normalizadas* (Salton y McGill 1983). En otras palabras, la aplicación del PLN a la RI es eficaz si se reduce el análisis a: *1) áreas temáticas limitadas*; *2) vocabulario limitado*; y *3) patrones sintácticos limitados*. En consecuencia, la solución de los problemas de la indización por medio de sintagmas sintácticos se encontraría en la capacidad

para desarrollar modelos lingüísticos que permitan generar y reconocer *formas canónicas*, en dominios limitados.

Un método lingüístico para la representación de sintagmas adecuado a los sistemas de IR por el que se obtienen formas normalizadas reduciendo las variantes sintácticas se propone en el sistema IRENA. En este sistema cualquier sintagma se equipara a una estructura, *phrase frame*, compuesta por un núcleo y distintos modificadores, *head-modifier*, (Arampatzis et al. 1998):

$$\texttt{PF} = [\texttt{h}, \texttt{m}]$$

Utilizando la estructura `PF`, los sintagmas nominales desde el punto de vista de la RI se normalizarían de la forma siguiente:

$$\texttt{NP} = \texttt{det}^* \ \texttt{pre}^* \ \texttt{head} \ \texttt{post}^* \rightarrow [\ \texttt{head}, \texttt{pre}^* \ \texttt{post}^*\ ]$$

donde

- `det` está compuesto por *determinantes, cuantificadores, etc.*
- `pre` está compuesto por *adjetivos, nombres, u otros sintagmas.*
- `head` está compuesto por *un nombre*
- `post` está compuesto por *adjetivos, sintagmas preposicionales, oraciones de relativo, etc.*

y donde

- el asterisco denota *iteración* de constituyentes.
- y los pre- y post-modificadores pueden incluir *recursivamente* otros sintagmas nominales.

Los sintagmas se derivarían de los documentos por medio de técnicas lingüísticas, utilizando etiquetado de categorías, desambiguación morfológica y análisis sintáctico. Después de la

aplicación de este proceso en el que además algunos constituyentes –como determinantes o cuantificadores se eliminan– la normalización de un sencillo sintagma nominal, como: «*digital library on medical science*», con la estructura `PF = [h, m]` daría lugar al siguiente resultado:

`digital library on medical science → [library, digital; on medical science]`

Otro sistema que utiliza métodos lingüísticos para la representación normalizada de sintagmas se propone en un modelo de recuperación basado en una cadena de módulos, *streams*, consistente en: *eliminación de palabras vacías*, *stemming morfológico, extracción de sintagmas, normalización de sintagmas y extracción de nombres propios* (Strzalkowski et al. 1999). La normalización de sintagmas se lograría reduciendo las formas superficiales a pares de palabras controladas, `head + modifier pairs stream`, tal y como se muestra a continuación:

- Las variantes de un sintagma

  `information retrieval`
  `retrieval of information`
  `retrieve more information`
  `information that is retrieved`
  `/.../`

  se reducirían al par normalizado `head + modifier`

  `retrieve + information`

  donde

  `retrieve` es el `head`

  `information` es el `modifier`

La técnica lingüística para la reducción de variantes sintácticas que vamos a adoptar en este trabajo se basa en el establecimiento de un paralelismo entre los lenguajes naturales y los artificiales, según el cual ambos lenguajes se definen por un conjunto de expresiones

simbólicas, o formalizaciones matemáticas, denominadas *Expresiones Regulares*. Ese conjunto de expresiones se define por reglas que especifican qué expresiones están bien formadas, a su vez, el conjunto de reglas constituiría una gramática electrónica. En consecuencia, la primera fase para la identificación de sintagmas es construir las gramáticas que reflejen las estructuras correctas de tales sintagmas y, posteriormente, trasladar esas estructuras a un mecanismo que no sólo sea capaz de reconocerlas sino que agrupe todas las variantes sintácticas de esas estructuras.

Las especificaciones metodológicas de este modelo se expondrán más adelante, por ahora nos vamos a limitar a indicar que la reducción de variantes sintácticas se va realizar por medio de *Transductores de Estado-Finito*, *Finite-State Transducer* (FST), definidos de forma simplificada como una red de nodos y arcos etiquetados con categorías POS que se encargarán de reconocer determinadas secuencias en el *input*, y de proporcionar algún tipo de información lingüística en el *output*. Una cadena se reconocería y transformaría si se produce un *path* desde un nodo, considerado el estado inicial, hasta llegar a otro nodo, considerado el estado final.

Utilizando este formalismo, una vez que se han construido las gramáticas electrónicas que generan las estructuras sintácticas correspondientes, se trasladarán a un autómata o transductor. De forma general, un transductor es un aceptador de construcciones sintácticas compuesto de alguna forma por *dos autómatas* que funcionan en *paralelo*: un autómata se encargaría de identificar las cadenas de caracteres superficiales y otro autómata se encargaría de relacionar dichas cadenas con las estructuras canónicas correspondientes (Fig. 1.7). Sin embargo, en este trabajo –y con el objetivo de utilizar este formalismo en los sistemas de RI como un mecanismos de control de construcciones sintagmáticas– vamos a transformar las formas sintácticas canónicas en *identificadores de sintagmas enumerados* que se van a implantar, en este caso, como agrupadores de estructuras (Fig. 1.8).

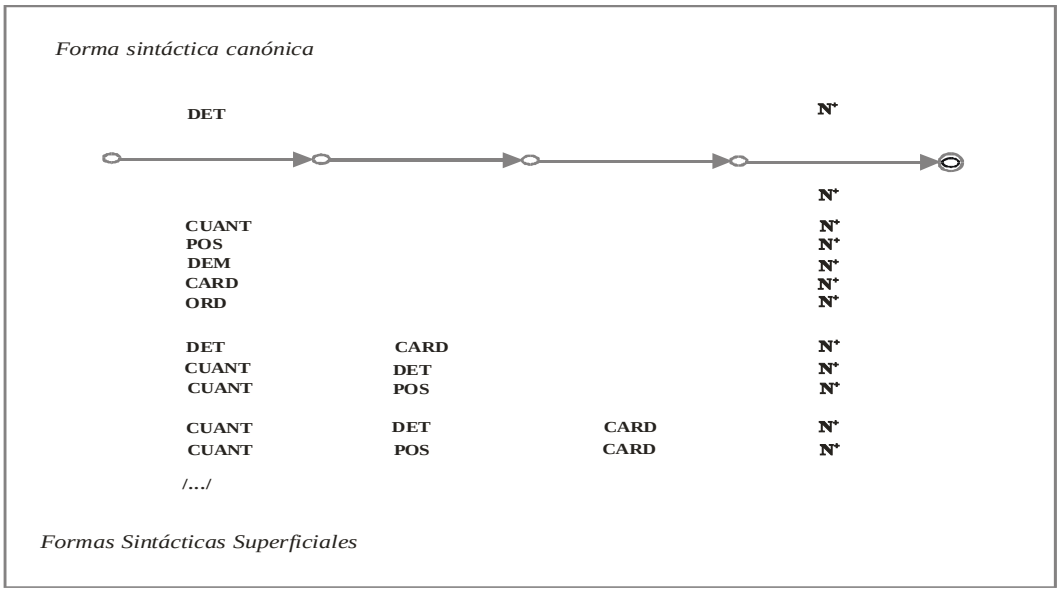


Fig. 1.7: Equiparación entre *formas sintácticas superficiales* y *una sola forma sintáctica canónica*

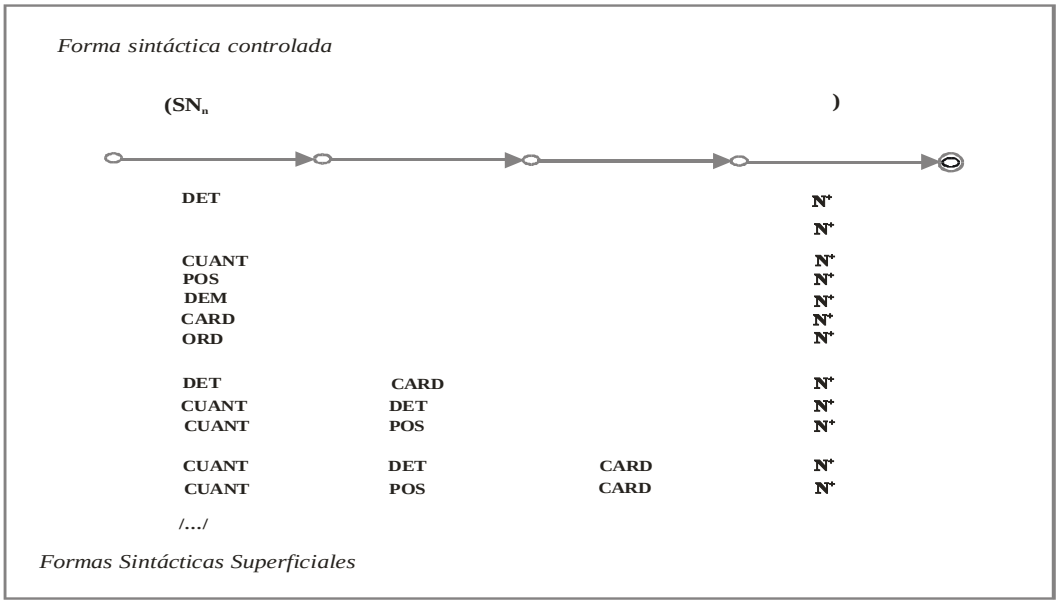


Fig. 1.8: Equiparación entre *formas sintácticas superficiales* y un *identificador de sintagma enumerado*

Por otra parte, las entradas a los FST serían cadenas etiquetadas, por eso el principal problema de este método es el de la ambigüedad en el etiquetado. Es muy difícil sintetizar aquí cómo se va resolver este problema: a grandes rasgos, con el objetivo de eliminar la

ambigüedad de las etiquetas de entrada, el texto se va convertir en transductores gráficos, y sobre éste se va a aplicar el mecanismo que se encargará de reconocer y agrupar las Expresiones Regulares generadas por las gramáticas electrónicas. Este proceso se describirá en otro capítulo, en el que se comprobará, además, que la aportación esencial de los formalismos de estado-finito es su capacidad para identificar y agrupar miles de variantes sintácticas a través de miles de transiciones entre los arcos.

Capítulo 2

MODELOS DE ESTADO-FINITO EN LA REPRESENTACIÓN LINGÜÍSTICA

La Lingüística Computacional se ocupa de la simulación del lenguaje natural, por medio de procesos automáticos, con el objetivo de realizar aplicaciones concretas como *Traducción Automática*, *Recuperación Información*, o *Extracción de Información*. Las aplicaciones en las que interviene el procesamiento automático del lenguaje natural requieren el uso de una gran cantidad de conocimiento sobre el propio lenguaje. Ese conocimiento tradicionalmente se puede dividir en categorías diferenciadas como: *fonología*, *morfología*, *sintaxis, semántica*, *pragmática* y categoría *discursiva* o *textual*. La primera cuestión que se plantea es que, aunque estas categorías se estudien de forma aislada con la finalidad de ser modeladas computacionalmente, actúan de forma conjunta. El segundo problema, relacionado con el planteamiento anterior, es el de la ambigüedad léxica que provoca que una palabra pueda poseer más de un categoría sintáctica y la ambigüedad sintáctica que hace que se pueda producir más de un análisis correcto para algunos de los componentes de la oración. La

indeterminación léxica y sintáctica, a la que se puede sumar la ambigüedad semántica, tiene como consecuencia que algunas secuencias puedan tener diferentes interpretaciones y, una vez más, se tiene que asumir la interconexión entre las distintas clases de conocimiento.

Para los problemas anteriores hay diversas soluciones dependiendo básicamente del *formalismo* que se use para representar el conocimiento lingüístico, y de la complejidad del *algoritmo* que se use para manejar esa representación (Jurafsky y Martin 2000). La mayoría de las aplicaciones del *Procesamiento del Lenguaje Natural* (PLN) poseen distintos mecanismos tanto para representar como para gestionar los distintos tipos de ambigüedad. Hay sistemas que tienen en cuenta el contexto para la asignación de categorías a las palabras, o que asignan probabilidades a las categorizaciones y a las reglas sintácticas de forma que se pueda obtener alguna medida de probabilidad relativa con el objetivo de seleccionar cuál ha de ser el análisis correcto. Todas estas cuestiones se han planteado tradicionalmente en la investigación lingüística por esta razón es necesario revisar los trabajos más importantes que se han realizado sobre estos problemas, centrándonos fundamentalmente en las importantes aportaciones de Chomsky.

En la arquitectura habitual de un sistema de reconocimiento de *patrones léxicos* y *sintácticos* se distinguen distintos niveles de representación, básicamente nivel de palabra y nivel de sintagma. Ambos se pueden dividir de forma generalizada en distintos componentes, que pueden variar según las características del sistema específico:

- El nivel léxico, *word level*, subdividido a su vez en dos componentes: *1) Tokenizador* Textual, *Text Tokenizer*; y *2)* Procesador Léxico, *Lexical Processor*.

- El nivel de procesamiento de sintagma y sentencias, *phrase level*, subdividido en: *1)* Reconocedor de Entidades, *Named Entity Finder*; *2)* Reconocedor de Sintagmas, *Phrase Recognizer*; y *3)* Reconocedor de Sentencias, *Clause Recognizer* .

Aunque para los objetivos de este trabajo no es de interés el desarrollo de una teoría rigurosa de la estratificación de los niveles de descripción lingüística, sí es interesante plantear una metodología que nos permita representar el conocimiento en un dominio a partir de un determinado nivel de análisis, y que además sea capaz de generarlo y reconocerlo alguna clase de mecanismo. Una descripción de todas las estructuras y componentes de un dominio de conocimiento sería una tarea muy extensa, además de inabordable en la práctica. Por tanto, es necesario desarrollar medios y recursos que reconozcan ese conocimiento a partir de mecanismos finitos. A su vez, la concepción metodológica de los niveles en los que actúan esos recursos es útil en los sistemas que manipulan algún tipo de PLN, porque de alguna forma sirve de modelo para dicha aplicación.

A raíz de lo anterior, una cuestión previa que habitualmente se plantea en las aplicaciones que manejan PLN es cómo se describen y reconocen las distintas clases de conocimiento necesario para realizar una tarea específica, en este caso reconocimiento de *patrones léxicos* y *sintácticos*. A pesar de la variedad y complejidad del conocimiento lingüístico que se precisa, hay un número relativamente pequeño de modelos y algoritmos dentro de los paradigmas computacionales bajo los que se puede plantear este problema, según una clasificación general realizada por Jurafsky y Martin (Jurafsky y Martin 2000) contamos con los siguientes:

a. *Modelos o formalismos* que se usan para capturar, o representar el conocimiento lingüístico. La clasificación a grandes rasgos de estos formalismos es la siguiente:

   - *Máquinas de Estado* (*State machines*). Dentro de este formalismo estarían: *Autómatas Finitos Deterministas* y *No-Deterministas, Transductores de Estado-Finito, Redes de Transición Aumentadas, Redes de Transición Recursiva*, o los modelos que tienen un componente probabilístico como los *Autómatas Probabilísticos* , los *Modelos de Markov* y los *Modelos Ocultos de Markov*.

- Sistemas basados en *Reglas Formales (Formal Rule Systems)*. Dentro de los cuales estarían: *Gramáticas Regulares*, *Relaciones Regulares*, *Gramáticas Libres de Contexto*, y todas las variantes de las gramáticas anteriores que incluyan un componente probabilístico.
- Formalismos basados en la *Lógica (Logic-based formalism)*. El modelo más usado es el denominado *Cálculo de Predicados de Primer Orden*, otros modelos dentro de este formalismo lo constituyen las *Redes Semánticas* o los *Grafos de Dependencia Conceptual*.
- *Modelos de incertidumbre.* Se trata de modelos aproximados, incompletos e inciertos para representar el conocimiento como es la *Teoría Probabilística de Bayes* (*Basic Bayesian Probability Theory*).

b. *Algoritmos* o conjunto de reglas que se usan para manipular los modelos o *representaciones anteriores* y producir el comportamiento que se desee. Entre los más importantes:

- *Algoritmos de Búsqueda de Espacio de Estado* (*State space search algorithms*)
- *Algoritmos de Programación Dinámica* (*Dynamic programming algorithms*)

Las *Máquinas de Estado* y los sistemas basados en *Reglas Formales* constituyen los modelos más adecuados para representar el conocimiento de la *fonología*, *morfología* y *sintaxis*. Los formalismos basados en la *Lógica* –así como las redes semánticas, *frames* o *scripts*– son los que se utilizan de forma generalizada para representar el *conocimiento semántico*, *pragmático* y *textual*, aunque también se pueden aplicar en representaciones sintácticas. A su vez, cada uno de estos *modelos se puede ampliar con procedimientos que usan técnicas probabilísticas para capturar datos lingüísticos*, con el objetivo de resolver los problemas de ambigüedad (Jurafsky y Martin 2000). Por otra parte, los algoritmos que se aplican en el PLN casi todos entrarían dentro del grupo de analizadores o *parsers* que aceptan un *input* y construyen algún tipo de estructura según para él. Los algoritmos de búsqueda de estado están asociados tanto a la *Máquinas de Estado* como a los *sistemas basados en Reglas* y se basan en una *búsqueda a*

través de un espacio de estados que representan hipótesis sobre el input (Jurafsky y Martin 2000) –los estados representarían un emparejamiento de estructuras con las distintas piezas del *input* y su objetivo sería lograr la estructura correcta después de haber procesado ese *input*–.

El primer modelo, las *Máquinas de Estado*, consiste en un formalismo con dos clases de entidades básicas: los *estados*, o estructura de datos, que ofrecen una descripción del problema, y la *función de transición* en virtud de la cual se pasa de un estado a otro. El *reconocimiento de patrones* se puede establecer como la equiparación entre las estructuras correctas de los estados de una máquina de estados con los distintos patrones.

El planteamiento de este problema en un *espacio* supone adoptar un formalismo que, mas que representar el conocimiento, representa la estructura de este problema en términos de alternativas disponibles en cada uno de sus posibles estados. El mayor inconveniente está en que las posibles transiciones entre estados puedan ser demasiadas y difícilmente codificables en el diseño de los algoritmos. Este problema lo desencadena una vez más la ambigüedad, según la cual un *input* puede tener distintas interpretaciones. Por esta razón, son muchos los esfuerzos que se hacen para minimizar y limitar el número de alternativas asociadas a cada estado. Pero antes de entrar en el análisis en profundidad de este modelo, se van a examinar los trabajos realizados en torno al problema de la representación lingüística y al problema de la ambigüedad

Otro aspecto importante a considerar es que la descripción lingüística en el nivel sintáctico no puede consistir en estructuras tan simple como las que presentan los mecanismos finitos, aunque sea este el formalismo que vamos a adoptar. Es necesario que la descripción en este nivel se formule en términos de análisis de constituyentes, o *parsing*. La gramática que permite una descripción de este tipo es la denominada *Gramática de Estructura de Frase*. A la vista de estas afirmaciones, los próximos apartados se van a dedicar a describir las bases desde las más simples a las más poderosas que se definen bajo la noción de gramática y a exponer cómo se interrelacionan los distintos modelos.

2.1. Teoría científica de la representación lingüística

Partiendo de la definición generalizada de que la lengua es un conjunto de sentencias con una longitud finita y construidas a partir de un conjunto de elementos finitos, el objeto de estudio de la sintaxis es el desarrollo de métodos de análisis y su meta es la elaboración de una *gramática* considerada como mecanismo o dispositivo para permita generar las sentencias de la lengua que se pretenda analizar. Siguiendo con esto, el problema es encontrar el mecanismo por el cual el hablante de una lengua determinada ejecuta su capacidad para hacer un uso *potencialmente infinito* con medios *finitos* (Chomsky 1957). Con este objetivo, se parte de una hipótesis básica: el principio de *competencia*, según el cual un hablante ideal tiene un conocimiento interior e inconsciente de su lengua, en virtud del cual es capaz de formar, de entender, de identificar, o de decidir sobre la pertenencia, o no, de una secuencia o enunciado a su propia lengua. La gramática de una lengua pretende ser una descripción de la competencia intrínseca de ese hablante. Sobre la noción de competencia, y con el rigor de las matemáticas, Chomsky busca un método para discernir la gramaticalidad, o no-gramaticalidad, de las secuencias, dicho de otro modo, intenta encontrar una teoría que justifique la gramática como si se tratara de una ciencia empírica.

A partir de aquí, se estable un paralelismo entre una gramática y una teoría científica: *una gramática de la lengua L es básicamente una teoría de L, toda teoría científica está basada en un número de observaciones finito e intenta relacionar los fenómenos observados y de predecir fenómenos nuevos construyendo leyes generales en términos de constructos hipotéticos.* De forma análoga, la gramática de una lengua está basada en un *corpus de locuciones, observaciones, finito, y contiene determinadas reglas gramaticales, leyes, formuladas en términos de palabras, frases, etc., particulares de esa lengua, constructos hipotéticos.* Las reglas expresan *relaciones estructurales entre las observaciones del corpus y*

el número infinito de oraciones generadas por la gramática, predicciones, más allá del corpus (Chomsky 1957).

La finalidad del análisis lingüístico de una lengua (L) es hallar una *gramática*, es decir, encontrar una teoría, o algún tipo de mecanismo, que permita generar y separar las secuencias gramaticales que son oraciones de (L), de las secuencias no-gramaticales que no son oraciones de (L). A su vez, siguiendo con la relación de semejanza a una ciencia empírica, el objetivo del análisis no se puede centrar sólo en la gramática o teoría particular de una lengua sino en una gramática o teoría general del lenguaje. Y aunque este último punto quede fuera del presente trabajo es interesante para fundamentar los métodos en los que se puede justificar el establecimiento de una teoría científica de la estructura lingüística.

Por tanto, la primera cuestión que hay que resolver es encontrar los criterios o la base sobre la cual se pueda determinar qué secuencias son, o no, gramaticales para seleccionar la teoría correcta de una lengua. Chomsky (Chomsky 1957) esquematiza tres procedimientos, que no son necesariamente correctos, para hallar esa base a partir de la relación entre la teoría general de la estructura lingüística y las gramáticas particulares que se puedan derivar de ella:

1. El primer método consiste en deducir de un *corpus* de expresiones de una lengua la gramática de dicha lengua. Se trata de un método muy ambicioso para construir gramáticas ya que la teoría se deduciría sólo a partir del *corpus*, por eso se le denomina *procedimiento de descubrimiento –discovery procedure–* para las gramáticas: *la teoría de la estructura lingüística* tendría *que proporcionar un método práctico para construir la gramática, dado un corpus.* Este procedimiento se rechaza porque la noción de gramaticalidad en L, en términos de locuciones observadas en L, ofrece sólo una solución parcial para aspectos concretos de la competencia lingüística, y, además, aunque el método sea correcto es inabordable en la práctica.

2. El segundo método se basa en decidir si una gramática es válida para un *corpus* de expresiones. Consiste en un procedimiento menos ambicioso, en el cual la *teoría* tendría que *proporcionar un método práctico para determinar si una gramática propuesta para un corpus es, o no, la mejor gramática para la lengua* de la que se ha extraído el *corpus*. A este método se le denomina *procedimiento de decisión –decision procedure–*. Los criterios para decidir si una gramática conviene, o no, al *corpus* se basan en parámetros sintácticos y semánticos. Esta forma de proceder no es correcta porque una oración puede tener sentido o significado y estar mal formada sintácticamente, y viceversa. Por tanto, la noción de gramaticalidad no puede ser identificada con ningún aspecto, ni de forma ni de contenido, con la semántica

3. El tercer procedimiento consiste en evaluar o graduar la gramaticalidad de un *corpus* de expresiones según su aproximación estadística a la lengua que se esté examinando. Así, las secuencias con *probabilidad cero*, o muy baja, se consideran *imposibles* y las secuencias con *probabilidades más altas*, se consideran *posibles*. Dicho de otro modo, se asume *posible* en el sentido de *altamente probable* y, de esta forma, las expresiones que se aproximen *estadísticamente a una lengua* dada pueden contemplarse como gramaticales y las que no se aproximan se consideran como no gramaticales. Se trata del método más débil, en el que la teoría sólo tiene que expresar cuál es la mejor gramática de la lengua, proporcionando meramente un procedimiento de evaluación –*evaluation procedure*–. Este planteamiento también es incorrecto porque si se realizara la ordenación anterior aparecerían tanto oraciones con probabilidades altas no-gramaticales, y viceversa. Por tanto, no existe una relación directa entre orden de *aproximación y gramaticalidad*. Sin embargo, las investigaciones estadísticas de la lengua han dado lugar a importantes modelos probabílisticos en los que dada una gramática se puede evaluar la aproximación de la lengua a dicha gramática. De hecho, las primeras fases de muchos sistemas que emplean técnicas de PLN, como es la desambiguación morfológica o la asignación de

categorías a las palabras, usan procedimientos probabilísticos. Por tanto, la relación entre gramática y estadística es la que más soluciones prácticas ha producido y, para dar respuesta a determinados problemas, se puede considerar un procedimiento adecuado.

El esquema de los tres modelos chomskyanos se representa en un gráfico (Fig. 2.1) en el cual la teoría se concibe como una máquina, o *caja translúcida*, que tiene distintas entradas, *inputs*, y salidas, *outputs*, según el modelo del que se trate. En la teoría que proporciona un método o procedimiento de descubrimiento *la máquina tiene como entrada un corpus de expresiones y como salida una gramática*. En el procedimiento de decisión, la máquina tiene *un corpus de expresiones y una gramática como entrada* y, dependiendo de que la gramática sea correcta o no, la salida será *si, o no, es correcta*. En el procedimiento de evaluación, la máquina tiene como *entrada las gramática* G_1 *y* G_2, o un número mayor, además de un *corpus* de expresiones y como salida la *gramática* seleccionada.

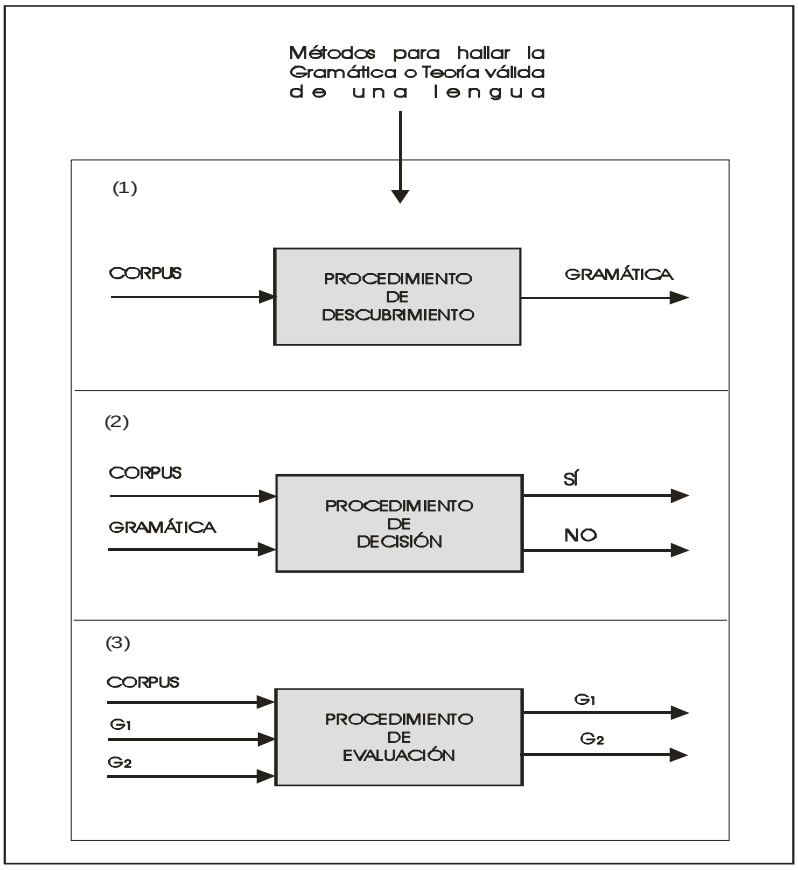


Fig. 2.1: Modelos de investigación para hallar teorías válidas de la estructura lingüística (Chomsky 1957)

A partir de este planteamiento, Chomsky deduce que la gramática es independiente de la semántica y de los modelos probabilísticos. Aunque éstos últimos son los preferibles por estar más sujetos a los datos y porque se basan en hechos empíricos. Por tanto, el *procedimiento de evaluación,* aún siendo el más modesto, es el más adecuado para dar respuesta a los problemas puedan surgir en la descripción de la estructura lingüística por basarse, entre otras cosas, en los datos.

Después de descartar total, o parcialmente los métodos anteriores, la descripción de la estructura lingüística de una lengua se concibe como una tarea muy compleja, por esta razón se plantean dos cuestiones para hacerla abordable en la práctica:

- Considerar la *gramática* no como una lista de secuencias de una lengua, que es infinita, sino como un mecanismo finito que permita generar secuencias infinitas.
- Buscar la solución práctica a la descripción de una lengua en términos de un sistema de *niveles de representación de estructuras lingüísticas,* y de constituyentes o componentes dentro de cada nivel.

Con este nuevo enfoque se desarrollaron distintos modelos:

1. Uno que tiene su origen en la *Teoría de la Comunicación*: Gramática de Estado-Finito.
2. Otro de gran repercusión en aplicaciones prácticas, que surge a partir del análisis de constituyentes de las sentencias, denominado Gramática de Constituyentes, o de Estructura Sintagmática, *Phrase Structure Grammar* (*PSG*) –la importancia de este formalismo fue fundamental para el desarrollo de la *Teoría del Lenguaje* y *Gramática Formal*, ya que tanto lenguajes artificiales de la lógica como los lenguajes de programación se generan por medio de *PSG*–.

Pero antes de justificar cada uno de estos modelos, es preciso mencionar una cuestión metodológica previa relativa a la forma en la que se obtienen los datos de la lengua que se esté examinando, y a este asunto se va a dedicar el epígrafe siguiente.

2.1.1. Niveles de representación lingüística

Siguiendo con la analogía de la investigación lingüística a una ciencia empírica cuyos métodos se basan fundamentalmente en la observación de datos, y en los fenómenos de la realidad, éste debiera ser el procedimiento a seguir. Sin embargo, la realidad lingüística es tan compleja que es preciso distinguir distintos niveles de análisis. En consecuencia, Chomsky (Chomsky 1965) distingue tres niveles generales de representación ligüística:

1. *Nivel de observación*: tiene como finalidad presentar meramente los datos producto de la observación de una lengua. La adecuación de la observación –*observational adequacy*– depende de una correcta representación de esos datos junto a una posible ordenación de los mismos en categorías. Pero, los datos obtenidos en este nivel no tendrían ningún valor, o sólo un valor instrumental, si no estuvieran orientados a los niveles superiores.

2. *Nivel de descripción*: teniendo como base la observación de los datos lingüísticos, a una gramática se la puede considerar descriptivamente adecuada –*descriptive adequacy*–. En este nivel, la *gramática está justificada en la medida en que describa correctamente su objeto de estudio*, es decir, la competencia del hablante ideal. Por tanto, la gramática está justificada por motivos *externos*, sobre la base de la *correspondencia* con los *hechos lingüísticos*.

3. *Nivel de explicación*: para satisfacer la adecuación explicativa –*explaratory adequacy*– la teoría lingüística tiene que seleccionar una gramática descriptivamente adecuada sobre la base de los datos observacionales. Constituye el nivel más alto de la teoría lingüística y tan sólo en él se pueda considerar a ésta como una auténtica teoría científica. Además, se trata de un nivel más profundo y más difícil de alcanzar, en este sentido la *gramática está*

> *justificada por motivos internos, sobre la base de su relación con la teoría lingüística que constituye una hipótesis explicativa de la lengua*.

Si hemos dicho que las ciencias observacionales se basan en la creencia de que hay un orden constituyente en todos los fenómenos de la realidad y, consecuentemente, en los lingüísticos, se puede decir que las estructuras del lenguaje están formadas por constituyentes y componentes dentro de determinados niveles. La noción de nivel es utilizada y explotada principalmente por la rama americana del *paradigma estructuralista* (Harris 1951), pero con la visión de nivelar no se agota la metodología analítica del estructuralismo sino que hay que tener en cuenta la noción de constituyente, reservado para los elementos que integran las estructuras dentro de cada nivel.

Según el enfoque anterior, se considera que el lenguaje está formado por capas que, en realidad, no son más que un conjunto de subsistemas jerarquizados. Aunque no hay inconveniente en fijar tantos niveles como se crea necesario, desde un punto de vista práctico, para cumplir el objetivo de la investigación y aportar una metodología útil de análisis lingüístico, habitualmente se consideraron tres niveles básicos: el *fonético*, el *morfológico* y el *sintáctico* –a los que se pueden sumar el nivel *semántico* y *el nivel pragmático*–. A partir de los niveles básicos, se obtendría el siguiente planteamiento:

- En el *nivel* de la fonología, los *componentes* son el fonema y los alófonos.
- En el *nivel* de la morfología, los *componentes* son el morfema y los alomorfos.
- En el *nivel* de la sintaxis, los *componentes* lo forman los denominados llamados *constituyentes inmediatos*. Un constituyente es cualquier secuencia de elementos que funciona como unidad en una construcción más larga, las partes inmediatas de esa construcción se denominan *constituyentes inmediatos*, y se indican por medio de un diagrama arbóreo.

Las metas de investigación, hasta la propuesta de Chomsky, se ocupaban de las descripciones de los elementos de una lengua a partir de la observación y con un objetivo taxonómico y

clasificatorio empleando *procedimientos de descubrimiento* –esta corriente esta representada principalmente por el estructuralismo y su fuerte conexión con el método científico–. La descripción de la lengua en esta corriente se realiza fundamentalmente por medio de estructuras, relaciones y componentes. En este contexto, surge la ambiciosa metodología de chomskyana, cuyos objetivos de investigación se dirigen hacia formalizaciones explícitas que describan la competencia lingüística y hacia la construcción de teorías que expliquen la mencionada competencia por medio de *procesos recursivos* y *reglas*. Aunque la idea original ha sido objeto de críticas y revisiones constantes, en las que se ha cuestionado la mayor parte de los presupuestos de partida, apareciendo múltiples reformulaciones, su aportación fue esencial para la construcción de hipótesis y modelos derivados de este poderoso planteamiento. En el ámbito de la investigación lingüística, el poder del nuevo procedimiento residió en el gran salto que supuso pasar de métodos basados en la observación a métodos que alcanzaron el nivel de *explicación científica*. A raíz de esto, se formalizaron distintos modelos que progresaron en su intento de superar las limitaciones del modelo precedente.

2.2. La representación de estructuras lingüísticas por medio de Gramáticas

Sobre el nivel explicativo de la descripción lingüística se reformula la noción de nivel lingüístico, o planos de descripción de la gramática. Un plano de descripción constituye un conjunto de recursos descriptivos para la construcción de gramáticas y se considera un procedimiento metodológico para representar las frases, normalmente se habla de: plano del fonema, de la palabra, de la frase o del texto. Cada uno de estos niveles se sitúan en un orden ascendente de complejidad, de tal forma que una *teoría lingüística es adecuada* según la *gramática correspondiente al conjunto de niveles que la teoría contiene* (Chomsky 1957). Esto es, una gramática puede ser adecuada en el nivel descriptivo, o una teoría puede ser adecuada en el nivel explicativo.

Tradicionalmente la representación de los fenómenos lingüísticos se plantean en distintos niveles, fonológico, morfológico, sintáctico y semántico. En el modelo de Chomsky de 1957 la teoría de la estructura lingüística se formula en el nivel sintáctico, quedando excluido el nivel semántico porque según este autor la sintaxis es independiente del significado –debido a que hay la gramaticalidad, o no gramaticalidad, de una frase no depende de aspectos semánticos–. Por otra parte, el nivel fonológico y morfológico también se excluye por la complejidad que supone operar con reglas que generen todas las combinaciones fonológicas y morfológicas de las palabras. En este punto, es preciso aclarar que la exclusión del componente fonológico y morfológico, denominado estructura morfofonémica o simplemente morfológica, es sólo por razones de simplificación porque dicho componente realmente forma parte del modelo en un nivel más bajo. Precisamente este componente será clave en el reconocimiento de patrones léxico como iremos tratando de demostrar.

Siguiendo con lo anterior, según la formulación del modelo de competencia de Chomsky se considera que: *a)* la descripción de los fenómenos lingüísticos de establece en nivel sintáctico; y *b)* la teoría explicativa de estos fenómenos es la denominada *Gramática Generativo-Transformacional*, o *Gramática Sintagmática*. La noción generativa de la descripción lingüística no pretende ofrecer un inventario de los fenómenos lingüísticos sino aportar una explicación de dichos fenómenos haciendo un uso ilimitado de medios limitados, como son las *reglas*. En un principio, el modelo sintagmático, o análisis de constituyentes, es el que se configuró como teoría científica en el nivel de explicación lingüística. Posteriormente, el modelo sintagmático se perfecciona con otro modelo más potente denominado *modelo transformacional* que persigue complementar el funcionamiento del modelo anterior concebido exclusivamente a nivel de estructura sintagmática, por eso a este modelo también se le denomina *Teoría Estándar* o *Gramática Generativo Transformacional.*

A partir de lo anterior, han sido constantes las correcciones y adiciones al modelo chomskyano de 1957, una de las más críticas más influyentes fueron las que partieron de Katz-Fodor y Katz-Postal (Katz y Fodor 1963; Katz y Postal 1964) que provocaron que se reformulara el modelo inicial con la inclusión de un componente semántico, dando lugar al

modelo chomskyano de 1965 (Chomsky 1965), también denominado *Teoría Estándar Ampliada*. Las críticas no se quedaron aquí, sino que hubo continuas revisiones que tuvieron como consecuencia directa que la imagen monolítica del modelo inicial desapareciera. A la vista de todas estas modificaciones, no es pertinente hacer aquí un resumen apresurado de todas las transformaciones que sufrió el modelo inicial. Por el contrario, nos vamos a centrar en la eficaz metodología que supuso el primer modelo de Chomsky por contener las bases de la denominada *Teoría Estándar* y por constituir el auténtico intento por aportar una explicación de la descripción lingüística en términos de análisis de constituyentes en el nivel sintáctico, quedando excluidos por ahora otros niveles.

Una vez delimitado el nivel de análisis que nos ocupa, el planteamiento de Chomsky fue investigar qué mecanismos computacionales serían capaces de representar la sintaxis del lenguaje natural. Todo esto condujo al desarrollo de la *Teoría de los Lenguajes Formales*, por medio de la cual se intentó crear modelos descriptivos de los fenómenos lingüísticos. Uno de los modelos más modesto del lenguaje formal se implanta con bastante fuerza por ser especialmente adecuado en la representación de aspectos del lenguaje natural, se trata de los mecanismos de estado-finito. Aunque estos mecanismos no sean apropiados para modelar la complejidad de las estructuras sintácticas del lenguaje natural son bastante eficaces para representar reglas fonologías, morfológicas y determinadas construcciones sintácticas del lenguaje natural, debido a que, dada su sencillez, se pueden implementar computacionalmente sin demasiada dificultad. Los próximos apartados van a estar dedicados a examinar modelos y teorías que parten de las consideraciones anteriores

2.2.1. Gramática de Estado-Finito

El análisis matemático de la información fue el punto de partida de la *Teoría de la Comunicación* (Shannon y Weaver 1945). De forma general, se trata de una teoría lingüística pre-chomskyana en la cual la transmisión de la información se considera un procedimiento de

reducción de incertidumbre según un formalismo de descripción de la estructura lingüística que se conoce como *Modelo de Markov*. En una *cadena de Markov* existe sólo un número finito (k) de estados posibles ($S_0, S_1, S_2, ..., S_k$) y en cualquier instante de tiempo la cadena debe estar en uno de estos estados, a una cadena de este tipo se la denomina comúnmente *Cadena de Markov Finita* (DeGroot 1989). Una máquina Markov puede generar cadenas de una lengua por medio de distintos pasos o etapas a través de determinados estados, proporcionando a su vez una descripción de la estructura lingüística según ese número finito de estados. Según lo anterior, una máquina de este tipo y una gramática se podrían considerar mecanismos equivalentes, por esta razón la *máquina de Markov* tuvo mucha importancia en el desarrollo de la teoría de autómatas y de las gramáticas formales.

Un autómata o máquina es una idealización usada como un dispositivo que es capaz de procesar símbolos o cadenas de entrada y cuyo objetivo es decidir si esas cadenas pertenecen o no a un lenguaje, o, también, dada una cadena de entrada generar otra de salida. Un componente fundamental en la estructura de los autómatas es la noción de estado que se aplica a la configuración global de todas las partes del autómata. Para definir un autómata finito se parte del siguiente supuesto: tenemos una máquina que puede estar en cualquiera de un número *finito de estados internos* y que la máquina pasa de un estado a otro al leer un símbolo determinado, además, de entre todos los estados internos, uno es un *estado inicial* y otro es un *estado final*. A partir del estado inicial el autómata recorre una secuencia de estados *produciendo una palabra en cada transición y terminando en el estado final*, a su vez la secuencia de palabras producida constituye una oración.

Según lo anterior, este autómata define una *lengua a partir del conjunto de oraciones que puede ser generado mediante este procedimiento*, se denomina *lengua de estados finitos* a toda lengua que puede ser producida por una autómata de este tipo y se llama *Gramática de Estado-Finito a la máquina misma* (Chomsky 1957). A su vez, una Gramática de Estado-Finito se puede representar en un *gráfico o diagrama de estados* (Shannon 1949), en el que cada nodo corresponde a un estado del autómata y cada arco a una transición. A partir de un diagrama de estados (Fig. 2.2) se pueden generar cadenas o frases de *izquierda-a-derecha*, o

desde el estado inicial, a la derecha, hasta el estado final, a la izquierda, siguiendo la dirección de las flechas, como *«un dominio representa entidades»* o *«un dominio identifica entidades»*.

Una Gramática de Estado-Finito se establece como el mínimo modelo de teoría lingüística, en la que se pueden representar simples listas oraciones como secuencias finitas de unidades generadas de *izquierda* a *derecha*. Con este planteamiento, se podría construir una gramática aunque fuera muy extensa, pero una lengua no es únicamente una lista de oraciones, y si lo fuera, en todo caso, sería una lista infinita En consecuencia, este tipo de gramática estaría formada por listas infinitas de oraciones y, por tanto, se rechaza como teoría de la estructura lingüística debido a su inaplicabilidad práctica.

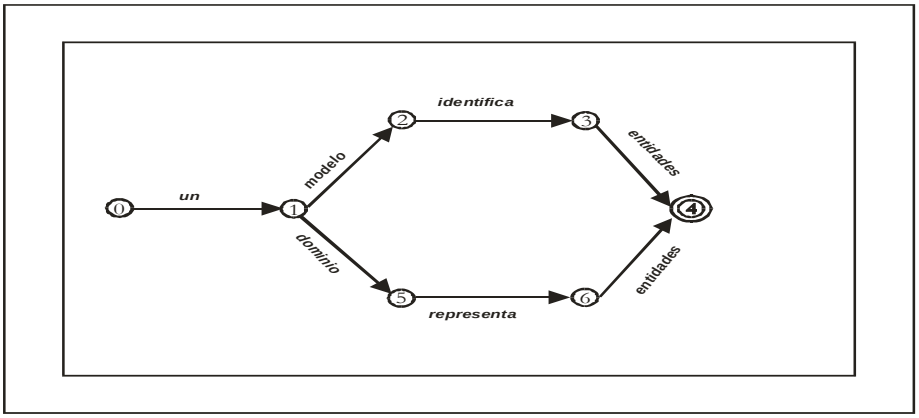


Fig. 2.2: Generación de cadenas por medio de una Gramática de Estado-Finito

Sin embargo, se trata de un modelo muy útil porque los procesos tienen un límite finito, y por tanto es posible constituir un dispositivo de este tipo que pueda generar y reconocer un número infinito de oraciones. El problema es encontrar los mecanismos recursivos que permitan generar el conjunto de oraciones infinitas de una lengua, para ello se necesita emplear métodos distintos y una concepción de la gramática más poderosa, como es la Gramática de Estructura Sintagmática (*Phrase Structure Grammar*, PSG), que se describirá en el apartado siguiente.

Pero siguiendo con la Gramática de Estado-Finito, para completar el modelo se puede asignar una probabilidad a cada transición entre estados. En un proceso de este tipo, al generar una cadena las diversas etapas de su producción pueden considerarse como distintos estados de un sistema en el que los cambios de estado sucesivos están condicionados por la probabilidad de un estado y la probabilidad de transición de un estado al siguiente. Para Weaver *un sistema que genera una sucesión de símbolos de acuerdo con ciertas probabilidades se llama un proceso estocástico*, y en el caso de que las probabilidades dependan de los sucesos anteriores, se denomina *Proceso de Markov* o *Modelo de Markov* (Shannon y Weaver 1945). Así, partiendo de un número finito de estados, $S_0, S_1, S_2, ..., S_k$, y de un conjunto de probabilidades de transición $P_i(j)$, existe una determinada probabilidad de que S_i pase al estado S_j.

Con un *Proceso de Markov* se puede determinar estadísticamente cuál es la probabilidad con la que aparecen determinadas unidades siguientes en el estadio de generación de las cadenas de una lengua. Si tuviéramos las siguientes probabilidades de que determinadas categorías gramaticales de las cadenas siguieran a otras en una oración:

$$P(un|determinante) = 0.7$$
$$P(modelo|sustantivo) = 0.6$$
$$P(dominio|sustantivo) = 0.6$$
$$P(identifica|verbo) = 0.4$$
$$P(representa|verbo) = 0.4$$
$$P(entidades|sustantivo) = 0.4$$

se podría obtener un Diagrama de Transiciones (Fig. 2.3) –que correspondería a un *Modelo de Markov*–. Las transiciones se interpretarían de la siguiente forma: la probabilidad de que la categoría determinante siga a la categoría sustantivo es 0.7, o la probabilidad de que la categoría sustantivo siga a la categoría verbo es 0.6 –en el capítulo siguiente describiremos como un proceso markoviano es equivalente a un Autómata Finito Probabilístico–. Con este

procedimiento se podría calcular la *incertidumbre asociada con cada estado* y la estructura lingüística de una lengua se podría definir como el *promedio de incertidumbre, medido por la probabilidad de estar en los estados asociados* (Chomsky 1957).

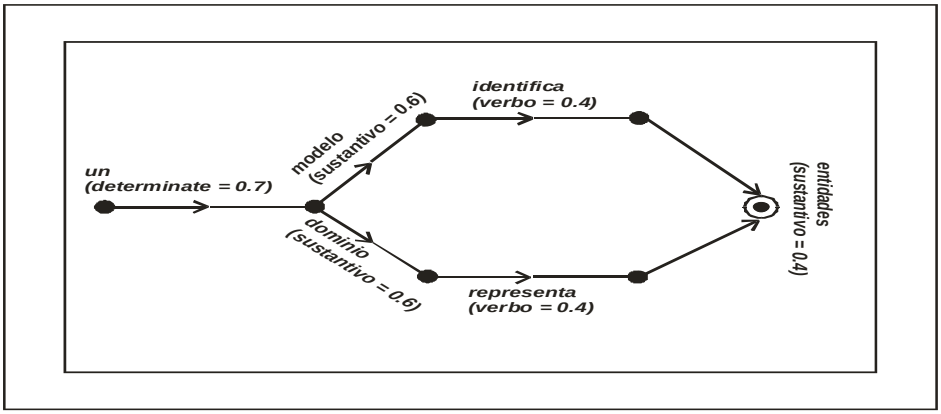


Fig. 2.3: Representación gráfica de un *Proceso de Markov*

Sin embargo, aunque los modelos probabilísticos se aplican principalmente para hallar las categorías gramaticales de las palabras, o *tokens*, y asignarles una etiqueta –en esta línea se encuentran distintos métodos estocásticos para la identificación y asignación de marcas a las categorías gramaticales (Kupiec 1992; Cutting et al. 1992; Church 1998)– no son adecuados para fundamentar la descripción lingüística. Según el punto de vista chomskyano, se persigue encontrar la estructura gramatical de una lengua y no su estructura estadística. Por tanto, los modelos probabilísticos no resuelven el problema básico de la determinación de la *gramaticalidad*, o *no-gramaticalidad*, de las oraciones, por esta razón el análisis lingüístico de un lengua se tiene que realizar con un modelo diferente, como es el que se describe en la sección siguiente.

Aceptando las limitaciones de las Gramáticas de Estado-Finito –en las que las cadenas y oraciones se generan de *izquierda*-a-*derecha* en un sólo nivel finito, ordenado de arriba abajo, de tal forma que se podrían generar todas las oraciones de una lengua de estados-finitos simplemente construyendo niveles más altos– tenemos que considerar que hay otras estructuras lingüísticas cuya descripción no se puede realizar por medio de un conjunto de

niveles finitos sino por un modelo diferente y más poderoso basado en el análisis de constituyentes, y en un concepto totalmente distinto de nivel lingüístico como es la *Gramática de Estructura de Frase o Sintagmática* (PSG) que se describirá a continuación.

2.2.2. Gramática de Estructura Sintagmática

Si una lengua puede ser descrita de manera elemental de *izquierda*-a-*derecha*, en términos de un solo nivel, esto es, si es una lengua de estados-finitos, entonces la descripción lingüística se podría simplificar únicamente construyendo niveles más altos. Pero para generar lenguas de estados-no-finitos, es decir, lenguas en las que haya oraciones o cadenas que no puedan ser representadas solamente como una secuencia finita de elementos se necesitan métodos distintos y un concepto más general de nivel lingüístico. En este sentido, se adopta una nueva forma de gramática, denominada de Gramática de Estructura Sintagmática, o *Phrase Structure Grammar* (PSG), en la que la teoría de la estructura lingüística se basa en el análisis de *constituyentes*, o *parsing*, y se configura como un conjunto no ordenado de *reglas de sustitución*, o reglas sintagmáticas, que son capaces de *asignar a una cadena una descripción estructural*.

Este nuevo método de análisis se expresa formalmente por un *conjunto finito* Σ *de cadenas* y un *conjunto finito* F *de reglas,* así dada la gramática $[\Sigma, F]$ se define una *derivación como una secuencia de cadenas finita*, empezando por una cadena inicial de Σ, y siendo derivada cada cadena de la secuencia a partir de la cadena precedente mediante la aplicación de una de las reglas de F, si una cadena es la última línea de una derivación, se dice que es una *cadena terminal* y se denomina lengua terminal al conjunto de cadenas terminales para alguna gramática (Chomsky 1957).

En la Gramática de Estructura Sintagmática cada frase se representa como un conjunto de cadenas y la noción de nivel lingüístico asociada a ella es diferente a la Gramática de

Estados-Finitos: en la estructura sintagmática se considera un solo nivel, el sintáctico, en el cual cada frase u oración está representada con un conjunto de representaciones. El análisis sintáctico consistirá en asignar una estructura de constituyentes a la frase, con este objetivo es necesario establecer las relaciones estructurales entre las palabras y otros constituyentes más amplios, como sintagmas y cláusulas.

El análisis de constituyentes se basa en la idea de que entre las cadenas, o palabras, y la frase hay grados intermedios en orden jerárquico que forman los miembros de la frase, es decir, hay relaciones entre esos miembros que forman unidades estructurales mayores. Una de esas unidades es el sintagma formado por un grupo de palabras y definido como el constituyente lingüístico que representa el grado máximo de proyección de un núcleo o categoría léxica mayor: cuando el núcleo sea un nombre estaremos ante un *Sintagma Nominal* (SN), cuando sea un adjetivo, un Sintagma Adjetivo (SA), cuando sea una preposición, un *Sintagma Preposicional* (SP) y un verbo, un *Sintagma Verbal* (SV). Otra unidad sintáctica mayor la forma la cláusula que se construye por medio de la unión de un SN y un SV, muchas veces se puede suprimir el SN o quedar implícito, en este sentido se consideraría una unidad lingüística vinculada básicamente a un verbo.

Habitualmente las gramáticas se especifican mediante un formalismo conocido como BNF (*Backus-Naur Form*) –se trata de una notación generalizada para describir la sintaxis tanto de los lenguajes naturales como de los formales introducida a partir de los trabajos de J. Backus y P. Naur (Naur 1963)–. Una gramática del tipo BNF consta de cuatro elementos:

1. Conjunto de *símbolos terminales*: formado por los símbolos o palabras que constituyen las cadenas del lenguaje.
2. Conjunto de *símbolos no-terminales* que categorizan los constituyentes de las frases, como puede ser el símbolo no-terminal Sintagma Nominal (SN) o *FraseNominal*, *NounPhrase* (NP), que denota un conjunto infinito de cadenas. En la notación FBN, los símbolos no-terminales van encerrados entre paréntesis angulares ⟨⟩ .

3. Un *símbolo de inicio* que es un símbolo *no-terminal* que denota cadenas completas del lenguaje, como es una oración o una sentencia : ⟨O⟩ ó ⟨S⟩ .
4. Un conjunto de *reglas de re-escritura* o *producciones* que definen las categorías sintácticas en función de otras categorías sintácticas y de los símbolos terminales del lenguaje. Una regla de producción con la forma $a \to b$ tiene el siguiente significado: si se encuentra a como parte de cualquier palabra c, se puede sustituir a por b en c, lo que permite transformar palabras en otras. En esta notación, el símbolo $\to$ de las reglas de producción se sustituye por $::=$.

Partiendo de que un lenguaje es un *conjunto de cadenas* tomadas de un conjunto finito de símbolos denominados terminales, o *tokens*, en el nivel más bajo, y que una gramática es un mecanismo formal que especifica qué cadenas pertenecen al lenguaje. Tendríamos que los *tokens* designarían las *categorías léxicas del lenguaje*, como puede ser la interpretación elemental de cadenas de caracteres del input (λ, a, b, c), tales como identificadores, números y operadores. Pero además de los *tokens*, una gramática usa otro conjunto de símbolos no-terminales en el nivel más elevado, estos símbolos no-terminales designarían las *categorías sintácticas del lenguaje* $(A, B, C, ...)$, como la interpretación de secuencias de *tokens*, tales como expresiones o enunciados. Los *tokens* y los símbolos no-terminales darían lugar al vocabulario de un lenguaje. Además, una gramática se compone de un símbolo no-terminal que representa el inicio de la frase, S , denominado raíz o axioma de la gramática, y un conjunto de reglas de producción $(X ::= Y \text{ ó } X \to Y)$.

El método de la *Gramática de Estructura de Frase*, *o Sintagmática*, se ha adoptado para caracterizar tanto las lenguas naturales como las formales, un ejemplo de este tipo de gramática se define formalmente como una tupla de cuatro elementos (Grishman 1986):

$$\textit{Gramática de Estructura de Frase} = (\Sigma_T, \Sigma_N, S, P)$$

donde

- Σ_T es el vocabulario o alfabeto terminal compuesto por las palabras o símbolos de la lengua que se esté definiendo.
- Σ_N es el alfabeto de los símbolos no-terminales que se utilizan para especificar la gramática. De tal forma que el vocabulario Σ se define como la unión del vocabulario terminal y no terminal:

$$\Sigma = \Sigma_T \cup \Sigma_N$$

- S es un símbolo inicial perteneciente al vocabulario no-terminal $(S \in \Sigma_N)$ y se denomina el *axioma* de la gramática.
- P es el conjunto finito de reglas de sustitución, denominadas producciones. Cada regla tiene la forma:

$$(a ::= b)$$

La única restricción que tienen estas reglas es que en la parte izquierda debe haber al menos un símbolo no-terminal, por tanto, a es un secuencia de uno o más símbolos de Σ, es decir $(a \in \Sigma^+)$ y b es una secuencia de cero o más símbolos de Σ, es decir $(b \in \Sigma^*)$. De tal forma que:

$$P = \left\{ (a ::= b) \mid a = c\ A\ d,\ a \in \Sigma^+, b,c,d \in \Sigma^*, A \in \Sigma_N \right\}$$

La aplicación sucesiva de reglas de re-escritura o producciones lleva desde la cadena inicial, S, a través de cadenas intermedias hasta llegar a la cadena terminal. Cada línea se obtiene de la precedente aplicando cada vez una regla de producción a un solo elemento, a este proceso se le denomina, como ya se dijo anteriormente, *derivación*. Las cadenas XAY y XZY pueden sucederse en una derivación si existe una regla $A \to Z$. Según esto, dada una gramática, se puede decir que una secuencia de cadenas es una *derivación – W* de V, si W es la primera y V la última cadena de la secuencia, y cada cadena de la secuencia es derivada de la precedente aplicando una de las reglas de sustitución –este proceso se representa con una doble flecha $\Rightarrow$ y se interpreta como genera o deriva–. Además, una derivación se considera *terminada* si la última cadena está formada sólo por cadenas terminales y, en consecuencia,

no es posible aplicar otra regla de sustitución, o no existe ninguna regla que pudiera transformar la última cadena. Por otra parte, hay que tener en cuenta que las producciones o reglas son recursivas en el sentido de que pueden utilizarse más de una vez en la generación de una misma frase.

Si tenemos la siguiente gramática:

$$G = (\Sigma_T, \Sigma_N, S, P)$$

donde

$$\Sigma_T = \{a, b, c, d, e, f\}$$
$$\Sigma_N = \{S, A, B, C, D\}$$
$$S = S$$

y donde las reglas de producción, P, son:

Producción 1 — $\langle S \rangle ::= \langle A \rangle \langle B \rangle$
Producción 2 — $\langle A \rangle ::= a$
Producción 3 — $\langle A \rangle ::= b$
Producción 4 — $\langle B \rangle ::= \langle C \rangle \langle D \rangle$
Producción 5 — $\langle C \rangle ::= c$
Producción 6 — $\langle C \rangle ::= d$
Producción 7 — $\langle D \rangle ::= e$
Producción 8 — $\langle D \rangle ::= h$

Si queremos saber si una cadena *a d h* se puede generar con esta gramática, partimos de S aplicando las sucesivas reglas de sustitución. Se puede concluir que *adf* es una derivación de

S después de realizar las distintas producciones (la aplicación de una secuencia de reglas a una palabra se expresa formalmente como $S \rightarrow * a\ d\ h$):

Producción 1 — $A\ B$

Producción 2 — $a\ B$

Producción 4 — $a\ C\ D$

Producción 6 — $a\ d\ D$

Producción 8 — $a\ d\ h$

o agrupando todas las producciones de forma simplificada como:

$$S \rightarrow \mathrm{A\ B} \mid a\ \mathrm{B} \mid a\ \mathrm{C\ D} \mid a\ d\ \mathrm{D} \mid a\ d\ h$$

La derivación de la cadena $a\ d\ h$ se representa de la forma siguiente:

S
$\Downarrow$
$\langle \mathrm{A} \rangle \langle \mathrm{B} \rangle$
$\Downarrow$
$a \langle \mathrm{B} \rangle$
$\Downarrow$
$a \langle \mathrm{C} \rangle \langle \mathrm{D} \rangle$
$\Downarrow$
$a\ d \langle \mathrm{D} \rangle$
$\Downarrow$
$a\ d\ h$

o agrupando todas las derivaciones:

$$S \Rightarrow \mathrm{A\ B} \Rightarrow a\ \mathrm{B} \Rightarrow a\ \mathrm{C\ D} \Rightarrow a\ d\ \mathrm{D} \Rightarrow a\ d\ h$$

Una gramática que describiera un subconjunto de estructuras se podría definir formalmente como:

$$G = (\Sigma_T, \Sigma_N, S, P)$$

donde

- Los símbolos terminales o palabras se tomarían de un vocabulario o diccionario definido como una lista de palabras permitidas. En este diccionario las palabras se agrupan de acuerdo con las categorías a las que pertenecen:

$$\Sigma_T = \left\{ \begin{matrix} \text{un, la, de, a, para, } \textit{mod}\text{elo, do}\textit{mini}\text{o, clases, entidades, identifica,} \\ \text{deno}\textit{mina}\text{r, nombrar,..} \end{matrix} \right\}$$

- Los símbolos no-terminales podrían ser:

$$\Sigma_N = \left\{ \begin{matrix} \text{Oración, S}\textit{int}\text{agma No}\textit{min}\text{al, S}\textit{int}\text{agma Verbal,} \\ \text{S}\textit{int}\text{agma }\textit{Pr}\text{eposicional, Deter}\textit{min}\text{ante, Verbo,} \\ \text{Nombre, }\textit{Pr}\text{eposición, Adjetivo,...} \end{matrix} \right\}$$

- El *axioma*, o símbolo inicial, pertenece a los símbolos no-terminales:

$$S = \texttt{ORACIÓN}$$

- Las reglas de producción o sustitución, P, podrían ser las siguientes (el símbolo $|$ se utiliza para separar las distintas alternativas del lado derecho y también se usa para agrupar todas las producciones de cada símbolo no-terminal):

$$\langle O\rangle ::= \langle SN\rangle + \langle SV\rangle$$
$$\langle SN\rangle ::= \langle Det\rangle + \langle NOMBRE\rangle$$
$$\langle SN\rangle ::= \langle NOMBRE\rangle$$
$$\langle SN\rangle ::= \langle Det\rangle + \langle NOMBRE\rangle + \langle SP\rangle$$
$$\langle SP\rangle ::= \langle Prep\rangle + \langle SN\rangle$$
$$\langle SV\rangle ::= \langle Aux\rangle + \langle VERBO\rangle + \langle SN\rangle$$
$$\langle Aux\rangle ::= \langle T\rangle + \langle M\rangle + (\text{haber} + \text{do}) + (\text{estar} + \text{do})$$
$$\langle T\rangle ::= F + \left\{\begin{matrix}\text{Pres}\\ \text{Pas}\end{matrix}\right\} + \text{Pna} + N^{o}$$
$$\langle \text{Pna}\rangle ::= \left\{\begin{matrix}1^{a}\\ 2^{a}\\ 3^{a}\end{matrix}\right\}$$
$$\langle N^{o}\rangle ::= \left\{\begin{matrix}\text{Sing.}\\ \text{Plural}\end{matrix}\right\}$$
$$\langle M\rangle ::= \text{poder}\,|\,\text{deber}\,|\ldots|$$
$$\langle \text{Det}\rangle ::= \text{un}\,|\,\text{una}\,|\,\text{uno}\,|\,\text{el}\,|\,\text{la}\,|\,\text{los}\,|\ldots|$$
$$\langle \text{NOMBRE}\rangle ::= \text{entidades}\,|\,\text{dominio}\,|\,\text{modelo}\,|\,\text{clases}\,|\ldots|$$
$$\langle \text{Prep}\rangle ::= \text{a}\,|\,\text{para}\,|\,\text{de}\,|\,\text{por}\,|\ldots|$$
$$\langle \text{VERBO}\rangle ::= \text{identificar}\,|\,\text{nombrar}\,|\,\text{denominar}\,|\ldots|$$

Si quisiéramos saber si una determinada cadena o frase, como «*un dominio representa entidades*», se puede generar con esta gramática tendríamos que partir del axioma aplicando las sucesivas reglas de producción:

$$O \rightarrow SN + SV$$
$$SN \rightarrow \text{Det} + N$$
$$SN \rightarrow N$$
$$SV \rightarrow \text{Aux} + V + SN$$
$$\text{Aux} \rightarrow \text{Presente} + 3^{a} + \text{Sing.}$$
$$\text{Det} \rightarrow \text{un}$$
$$N \rightarrow \text{dominio}\,|\,\text{entidades}$$
$$V \rightarrow \text{representar}$$

Por tanto, la frase anterior es una derivación del axioma después de aplicar una secuencia de reglas de sustitución. La derivación de las cadenas terminales se realiza de la forma siguiente:

⟨O⟩
⇓
⟨SN⟩⟨SV⟩
⇓
⟨Det⟩⟨N⟩⟨SV⟩
⇓
un⟨N⟩⟨SV⟩
⇓
un dominio⟨SV⟩
⇓
un dominio⟨V⟩⟨SN⟩
⇓
un dominio⟨Aux⟩⟨V⟩⟨SN⟩
⇓
un dominio Pres. 3ª Sing⟨V⟩⟨SN⟩
⇓
un dominio Pres. 3ª Sing. representar⟨SN⟩
⇓
un dominio representa⟨N⟩
⇓
un dominio representa entidades

La aplicación de las producciones se puede efectuar a los símbolos situados más a la izquierda de la frase o más a la derecha, según el caso se denomina *derivación más a la izquierda* o *más a la derecha*. A su vez, las derivaciones se pueden representar en forma de los denominados *árboles de análisis sintácticos* (*parse tree*), o *árboles de derivaciones* (Fig. 2.4), como el siguiente:

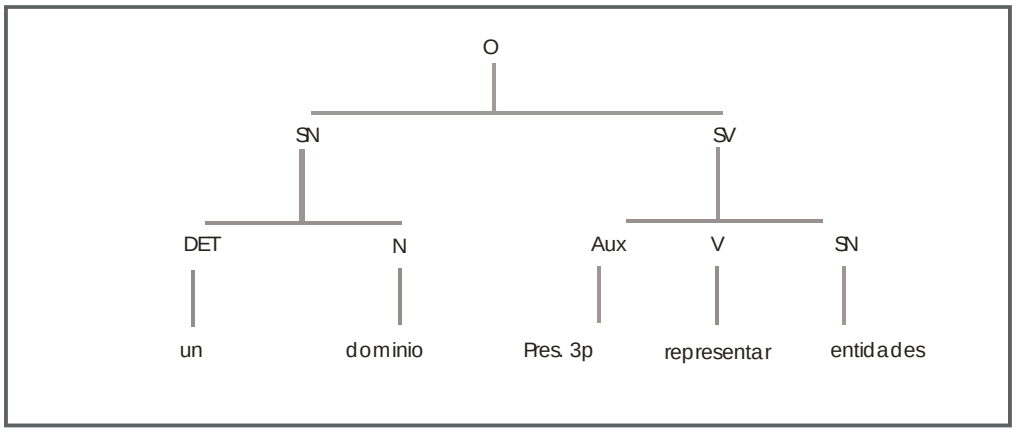


Fig. 2.4: Árbol de derivación sintáctica

Los árboles de derivación se utilizan tanto en los compiladores de los lenguajes de programación, para definir la estructura sintáctica de las frases o sentencias de los programas fuente, como en el PLN para la representar el análisis sintáctico de las oraciones. Un árbol de análisis, también denominado *diagrama ramificado*, consta de los siguientes elementos:

- Nodo superior situado en la *raíz del árbol* etiquetado con el símbolo inicial o axioma de la gramática.
- Nodos intermedios etiquetados con símbolos no-terminales de la gramática.
- Nodos inferiores, *hojas del árbol*, etiquetados con símbolos terminales.
- Derivaciones representadas por arcos o *ramas*: parten del axioma y de los símbolos no-terminales situados a la izquierda de las reglas de producción y tienen tantas derivaciones como símbolos terminales y no-terminales aparezcan a la derecha de las reglas.

Después de que la derivación se considere terminada, se puede afirmar que la gramática anterior genera una cadena de terminales desde el símbolo inicial, cambiando los patrones que se hallan en el lado izquierdo de las reglas de producción con las expresiones correspondientes de la derecha. Esto se expresa formalmente del siguiente modo: X es una oración o frase si existe una derivación desde el axioma a los símbolos terminales de esa frase después de aplicar una secuencia de producciones $S \rightarrow *X$.

El sistema de re-escritura del analizador sintáctico utiliza la derivación partiendo de un objetivo, axioma S, hasta llegar a unos datos o hechos que no son otra cosa que la frase a analizar. Para generar esta frase a partir del símbolo inicial se usan básicamente dos estrategias de análisis: *a)* dirigida por el objetivo de forma descendente, *top-down*, comenzando por el axioma y aplicando las producciones hacia delante hasta que los símbolos terminales de árbol se correspondan con los constituyentes de la frase; y *b)* dirigida por los datos o hechos de forma ascendente, *bottom-up*, comenzando por la frase a analizar y aplicando las reglas hacia atrás, hasta llegar al nodo raíz. La elección de una estrategia u otra dependerá de las características de la gramática o del grado de ambigüedad léxica, aunque muchas veces estos dos métodos se pueden combinar.

En la derivación es fundamental la aplicación de un sistema de reglas, un conjunto de las cuales se encarga de re-escribir símbolos categoriales, o reglas sintagmáticas, mientras otro re-escribe formantes léxicos, o reglas morfológicas. A esto se añade otro conjunto de reglas transformacionales que funcionan a partir de las cadenas generadas por las reglas sintagmáticas. Desde un punto de vista metodológico la importancia de las reglas es fundamental porque formalizan del modo más explícito posible la descripción de los fenómenos lingüísticos. En el próximo apartado se va a profundizar en este aspecto porque, debido a nuestro objetivo, aunque los mecanismos de estado-finito no sean adecuados para representar las reglas sintagmáticas y transformacionales dada su complejidad, no ocurre lo mismo con las reglas que operan con los elementos léxico-terminales.

2.2.2.1. Componentes y reglas de la Gramática de Estructura Sintagmática

El sistema de reglas de re-escritura del modelo de Chomsky de 1957 permite derivaciones de cadenas en un nivel lingüístico totalmente distinto al nivel propuesto por el modelo de estado-finito. La teoría lingüística de la Gramática de Estructura de Frase o Sintagmática (PSG), se basa en el análisis en constituyentes y se define básicamente por un conjunto finito de

cadenas Σ y un conjunto de reglas F del tipo $X \rightarrow Y$ que se interpretan como *re-escríbase* X *como* Y.

Según esta gramática, una derivación es una secuencia de cadenas finitas que se obtienen aplicando las reglas de F, se considera *derivación terminada a una cadena final que no puede ser re-escrita ni una vez más por las reglas de* F (Chomsky 1957). Sin embargo, las secuencias $\underline{un\ \langle N\rangle\langle SV\rangle}$ o $\underline{un\ \ dominio\langle SV\rangle}$, del ejemplo anterior, no se consideran derivaciones terminadas aunque la secuencia de cadenas finales «*un dominio representa entidades*» constituya una cadena terminal de la gramática. Así, esta secuencia terminal estaría representada por un conjunto de cadenas: $\underline{\langle O\rangle}$, $\underline{\langle SN\rangle\langle SV\rangle}$, $\underline{\langle Det\rangle\langle N\rangle\langle SV\rangle}$, y no sería posible establecer *niveles finitos,* ordenados del más alto al más bajo, con una representación para cada cadena dentro de estos *subniveles*. En otras palabras, no hay una *correspondencia o biyección entre los conjuntos de representaciones* (Chomsky 1957), muchas veces un sintagma nominal puede estar contenido en un sintagma verbal, o viceversa, y no hay forma de representar esa relación en un nivel finito.

Por esta razón, la estructura sintagmática se considerada como un solo nivel, el nivel sintáctico, con un conjunto de representaciones para cada oración de la lengua, y se asume que algunas de estas representaciones no constituyan cadenas terminales que puedan ser reconocidas por procesos de estados-finitos. Además, para completar la gramática también se pueden hacer en derivaciones sobre los elementos léxico-terminales, o morfemas, aplicando reglas fonológicas que conviertan una cadena de morfemas en una cadena de fonemas. Pero hay una diferencia sustancial con respecto a las reglas sintagmáticas y es que los elementos de estas reglas sí pueden ser clasificados en niveles finitos. Pero, antes de desarrollar esta idea es preciso hacer una distinción de los componentes y una subdivisión de las reglas que actúan en dicho modelo. Básicamente en la Gramática Sintagmática, también denominada Gramática *Generativo-Transformacional*, intervienen tres componentes:

1. *Componente Sintáctico*
    a. *Subcomponente de base*

- *Reglas Sintagmáticas*
- *Reglas Léxicas*

b. Subcomponente transformacional

- *Reglas transformacionales*

2. Componente Semántico

3. Componente Fonológico

Limitándonos por ahora al componente sintáctico, por constituir el eje central de la metodología planteada en este modelo, tenemos que la descripción de los fenómenos lingüísticos en este componente se realiza en dos niveles de descripción, *estructura superficial / estructura profunda*, según el cual todo fenómeno observable o superficial tiene su correspondencia en otro no observable, o profundo. Esto conlleva que la descripción de las oraciones se efectúe en una doble estructura: una profunda, constituida por las reglas sintagmáticas, y otra superficial constituida por formantes léxicos, estando ambas relacionadas por medio de las *reglas de transformación*. Por esta razón el componente sintáctico se divide en: un subcomponente de base y un subcomponente transformacional. El subcomponente de base incluye:

- *Reglas Sintagmáticas*, o reglas de base que formalizan el análisis en constituyentes inmediatos de las oraciones. Son reglas generativas del tipo $X \rightarrow Y$ (independiente del contexto) o $WXV \rightarrow WYV$ (dependiente del contexto), que se interpretan como *re-escríbase* X *como* Y. Básicamente re-escriben símbolos categoriales, como $\mathrm{O} \rightarrow \mathrm{SN} + \mathrm{SV}$, $\mathrm{SN} \rightarrow \mathrm{Art} + \mathrm{N}$,...
- *Reglas Léxicas* que se encargan de insertar elementos léxicos en los símbolos categoriales. Se trata de reglas generativas que re-escriben formantes léxicos, como $\mathrm{Det} \rightarrow \mathrm{un}$, $\mathrm{N} \rightarrow \mathrm{dominio} \mid \mathrm{entidades}$,... En última instancia, estas estructuras superficiales o léxicas habría que interpretarlas a continuación en matrices de rasgos fonológicos y rasgos sintácticos. Los rasgos fonológicos se caracterizan en representaciones binarias –presentes $[+]$ o ausentes $[-]$, como $[+\texttt{vocálico}]$ $[-\texttt{consonántico}]$–. Los rasgos sintácticos se caracterizan

representaciones del tipo [+Nombre] [+Común] [+Masculino]. Por último, estas unidades léxicas se podrían interpretar también por medio de rasgos semánticos organizados en diccionarios que aportarían datos de contenido del tipo [+Objeto físico] [+Actividad social] –como ya se hizo alusión, el componente semántico se incorporó a segundo modelo de Chomsky después de las críticas de Katz y Postal (Katz y Postal 1964)–.

A su vez, el subcomponente transformacional se encarga de relacionar la estructura profunda con la estructura superficial por medio de una serie de reglas que nacen del intento por ofrecer una *teoría explicativa* de los fenómenos lingüísticos. El resultado de la aplicación de las reglas transformacionales a las estructuras sintagmáticas es un conjunto de estructuras superficiales a las que se les aplica reglas léxicas. Básicamente el subcomponente transformacional incluye:

- *Reglas Transformacionales* que se encargan de transformas o convertir cadenas con estructura sintagmática en cadenas superficiales. Esto es, derivan la estructura superficial mediante inserciones, eliminaciones o permutaciones a partir de la estructura profunda. Se pueden aplicar, además, en cualquier nivel de derivación y explicarían por ejemplo la transformación de una oración activa en otra pasiva ($O \rightarrow SN_1 + Aux + V + SN_2$ en $O \rightarrow SN_2 + Aux + ser + do + V + por + SN_1$)

El planteamiento del modelo generativo se fundamenta no sólo en la descripción de los fenómenos lingüísticos sino en explicar por medio de reglas la formalización de dichos fenómenos. Por esta razón, este modelo no sólo recoge las importantes investigaciones estructurales y taxonómicas realizadas anteriormente sobre análisis formal de los constituyentes de las estructuras fonológicas, sintácticas y semánticas sino que va más allá persiguiendo formalizar y explicar las transformaciones que experimentan dichas estructuras desde la estructura profunda a la superficial. Y, aunque el modelo trate esencialmente de la sintaxis generativa, se puede hablar también de fonología y semántica generativa. Si las dos últimas quedaron fuera del modelo fue porque se consideró, por un lado, que la gramática, o

teoría de una lengua, no se podía basar en aspectos de significado y, por otro, que resultaba muy complejo intentar producir por medio de reglas todas las combinaciones fonológicas de una lengua.

Centrándonos en la causa de la exclusión de la fonología comprobamos que se debió esencialmente a cuestiones prácticas. Pero, esto no quiere decir que las unidades fonológicas no se puedan representar de forma individual en matrices de rasgos y en reglas que especifiquen las distintas condiciones en las que aparecen dichos rasgos. El desarrollo de la fonología generativa se realizó en *The Sound Pattern of Englih* (Chomsky y Halle 1968) y fue el germen de una serie de estudios que pretenden demostrar la adecuación de los métodos de estado-finito en la representación de las reglas fonológicas.

2.2.2.2. Reglas de dos-niveles frente a reglas generativas

Si los métodos de estado-finito no son los más apropiados para representar la sintaxis completa del lenguaje natural, no ocurre lo mismo con la fonología. La idea original de que las reglas fonológico-generativas sí son susceptibles de ser representadas por medio de modelos de estado-finito partió de un importante trabajo realizado por Johnson (Johnson 1972). Tomando como herencia la fonología generativa clásica, este lingüista computacional demuestra que determinadas alteraciones fonológicas se pueden representar con técnicas de estado-finito, y a este formalismo se le denomina comúnmente *fonología de dos-niveles*.

La fonología generativa intentaba aportar una explicación de los procesos que iban de un nivel de abstracción máximo, como son los rasgos fonológicos mencionados en el epígrafe anterior, hasta las realizaciones fonéticas en el nivel superficial. Ambos niveles se encontrarían relacionados mediante una serie de reglas aplicadas de forma secuencial que harían posible la transformación de un nivel en el otro. Frente a este planteamiento, uno de los efectos más significativos aportados por Johnson (Johnson 1972) fue que estos dos niveles

se podría implementar usando técnicas de estado-finito, pero para ello la aplicación de las reglas secuenciales se tendría que sustituir por la aplicación de reglas simultáneas. La consecuencia directa es que la fonología de dos-niveles se puede considerar una *alternativa coherente a la fonología generativa* y lo más importante es que *se puede tratar de forma computacional* (Antworth 1995), aunque en ningún caso se pueda establecer como una teoría completa de la fonología.

La distinción de las reglas de la fonología generativa y de la fonología de dos-niveles se concibe aquí como una cuestión relevante porque, dada la naturaleza de este trabajo, tal distinción nos ayudará a entender cómo se representa este formalismo con técnicas de estado-finito y, de forma indirecta, nos permitirá sentar las bases para hacer extrapolable o adaptar dichas representaciones a otras posibles aplicaciones. Clave en este punto es una revisión pormenorizada de las propiedades formales que distinguen a ambas reglas realizada por Antworth (Antworth 1995):

A. Propiedades formales de las reglas generativas

1. ***Transforman*** un símbolo en otro, es decir, las reglas generativas determinan la conexión entre dos niveles de representación, subyacente y superficial, y tienen la forma general de $A \rightarrow B/\_\_Z$ que se interpreta como: A se convierte en B cuando está precedida por Z. Se trata de reglas transformacionales, también denominadas *reglas de producción*, que re-escriben un símbolo en otro. Esto es, la relación entre A y B se define como un cambio dinámico en el que un símbolo se transforma en otro símbolo, en consecuencia después de aplicar esta operación dicho símbolo deja de estar disponible para otra regla. En este sentido, las reglas de re-escritura generativas intentan caracterizar la relación *entre niveles de representación especificando cómo transformar representaciones de un nivel en representaciones de otro nivel*.

2. Se aplican de forma ***Secuencial***, o una después de otra, convirtiendo formas subyacentes en formas superficiales por medio de un número indeterminado de niveles intermedios. Esto es, cada regla aplicada secuencialmente *crea como output un nuevo nivel intermedio de representación que sirve como input a la nueva regla*, esto significa además que la forma subyacente deja de estar inaccesible para las reglas posteriores.

3. Son ***Unidireccionales*** y se aplican de forma ordenada, esto es, transforman formas subyacentes en formas superficiales, no viceversa. La interacción entre un par de reglas está controlada por el requerimiento de que se aplican en orden secuencial, y en el caso de que se aplicaran *en otro orden podrían dar como resultado un output incorrecto*.

B. Propiedades formales de las *reglas de dos-niveles*

1. Definen una ***Correspondencia*** entre dos niveles de representación, subyacente o léxico y superficial, y tienen la forma de: $A:B \Rightarrow \_\_Z$ que se interpreta como: la forma léxica A corresponde a la forma superficial B cuando está precedida de Z. Se trata de reglas declarativas no transformacionales, en consecuencia el símbolo A no cambia a B sino que permanece después de aplicar la regla. En esencia, las *reglas de dos-niveles expresan una correspondencia estática y no una regla de re-escritura*.

2. Se aplican de forma ***Simultánea***, es decir, la aplicación de las reglas de dos-niveles de forma simultánea o en paralelo se produce porque expresan una correspondencia y no una transformación, por tanto siempre se aplican con éxito incluso si algunas de las formas, subyacente o superficial, está vacía. Por tanto, no hay niveles intermedios de representación con lo cual sólo se permiten dos-niveles, *nivel subyacente* y *nivel superficial*, de ahí su denominación. Además, la relación entre estos dos niveles *se expresa por*

posicionamiento directo de un símbolo en otro dando, lugar a una correspondencia estática entre pares de símbolos. Las reglas de dos-niveles no cambian A en B, y de esta forma, A podrá estar disponible a otras reglas. Es otras palabras, mientras las reglas generativas acceden sólo a la forma intermedia en cada etapa de la derivación, las reglas de dos-niveles acceden tanto al nivel subyacente o léxico como al nivel superficial, y, como después de aplicar la regla, los símbolos permanecen se puede establecer una correspondencia tanto del nivel subyacente-al-superficial como del nivel superficial-al-subyacente.

3. Son ***Bidireccionales*** operando tanto en la dirección subyacente-a-superficial como en la dirección superficial-a-subyacente. A la primera dirección se la denomina *modo de generación según la cual las reglas de dos-niveles aceptan una forma subyacente como input y devuelven una forma superficial*, a la segunda dirección se la denomina *modo de reconocimiento según el cual las reglas aceptan una forma superficial como input y devuelven una forma subyacente*. Además, y lo que es más relevante para nuestro objetivo, por tratarse de reglas bidireccionales cuando se implementen computacionalmente no estarán limitadas al modo de generación para producir palabras sino que también se pueden aplicar para analizar palabras en el modo de reconocimiento. Por otra parte, la interacción de las reglas aquí no se controla por una ordenación secuencial sino *especificando el contexto como cadenas de correspondencias de dos-niveles*.

El formalismo en el que se fundamenta la fonología de dos-niveles es especialmente valioso porque, como se ha dicho, se puede desarrollar computacionalmente de forma relativamente sencilla y eficaz con técnicas de estado-finito. Esta idea teórica fue decisiva para el progreso de otros modelos, pero su implementación práctica más significativa fue servir de base para el desarrollo del denominado *analizador morfológico de dos-niveles* (Koskenniemi 1983),

donde el término morfología incluye tanto la morfología propiamente dicha, o estudio de la descomposición de palabras, como la fonología entendida en su acepción de morfonología, o estudio de las alteraciones fonológicas de la morfología. Estos analizadores tienen *dos-niveles* uno de entrada o superficial que corresponde a la palabra que se analiza y otro de salida o léxico que corresponde al *stem* y afijo –esta característica hará posible que se puedan representar como transductores, tal y como se describirá en capítulos sucesivos–.

2.2.3. Gramáticas Formales

El estudio matemático de las propiedades de los lenguajes formales dio origen a las *Gramáticas Formales* que describen aquellos fenómenos estructurales de las frases que se expresan formalmente. Por tanto, las descripciones no observables y verificables, como puede ser el contenido o el significado, quedan excluidas de su consideración. En relación con esto, un analizador sintáctico o *parser* es un programa que toma una gramática y una secuencia lingüística y determina si esa secuencia es *gramatical*, o *no-agramatical*, es decir, si la gramática es capaz de determinar la derivación de esa secuencia.

Una propiedad matemática de los lenguajes formales como de los lenguajes naturales es la *recursividad* definida como la posibilidad de aplicar las reglas de producción un número indefinido de veces, mediante esta propiedad las gramáticas están capacitadas para generar estructuras sin ningún limite. En este sentido, hay dos conceptos a tener en cuenta que son los de: *lengua enumerable recursivamente* y *lengua recursiva* (Grishman 1986). Una lengua es *enumerable recursivamente* si el *parser* es capaz de generar una enumeración secuencial de las secuencias del lenguaje; y una lengua es *recursiva* si el *parser* es capaz de distinguir si las secuencias recursivas son *gramaticales*, o *no-gramaticales*. Los programas o analizadores sintácticos se diseñan para aceptar gramáticas pero no todas las gramáticas porque, de no ser así, serían capaces de generar estructuras infinitas sin las limitaciones, que en la práctica o actuación lingüística, tienen las lenguas naturales. Por esta razón, es preciso imponer

restricciones a los formalismos gramaticales, y dependiendo de esas restricciones las gramáticas se consideran más o menos poderosas, o expresivas.

En relación con lo anterior, como el objetivo de las gramáticas es generar sólo *secuencias gramaticales* aceptables de una lengua natural es preciso restringir el poder de las reglas de producción, o lo que es lo mismo, limitar la capacidad generativa de las gramáticas para que produzcan sólo secuencias gramaticales de una lengua. Todo esto conduce a que las gramáticas se clasifiquen de acuerdo con su capacidad productiva, esto es, según el conjunto de lenguajes que son capaces de generar.

Chomsky (Chomsky 1957) establece distintas clases de formalismos gramaticales que difieren sólo en los tipos de reglas de producción. Las clases se disponen en un orden de inclusión, *Tipo 3* $\subseteq$ *Tipo 2* $\subseteq$ *Tipo 1* $\subseteq$ *Tipo 0*, que se denomina *Jerarquía de Chomsky*, en la cual cada clase, o gramática, sirve para describir un tipo de lenguaje así como los lenguajes que pueden describirse utilizando una gramática de menor capacidad. En consecuencia, la clasificación de las gramáticas según la capacidad expresiva para definir lenguajes introducirá a su vez el criterio para clasificar los mecanismos o autómatas que son capaces de reconocerlas (Fig. 2.5). De *menor a mayor restricción del formalismos gramatical* y *de mayor a menor capacidad para definir lenguajes* (Cohen 1991), en la jerarquía anterior se distinguen cuatro tipos de gramáticas según el criterio que imponen las restricciones de las reglas de producción $(X ::= Y \quad \text{ó} \quad X \rightarrow Y)$:

- *Tipo 0* (*Gramáticas sin Restricciones, de Estructura de Frase o Sintagmática*): en la parte izquierda, X, puede haber cualquier cadena con al menos un símbolo no-terminal. En la parte derecha de las reglas, Y, puede aparecer cualquier cadena, incluida la cadena vacía, por lo tanto no tiene ningún tipo de restricciones. Las reglas de re-escritura podrían ser del tipo:

$S \rightarrow a\ A$
$A \rightarrow a\ B\ b$
$a\ B \rightarrow \lambda$
$a\ C\ d \rightarrow e\ B\ h$

- *Tipo 1* (*Gramáticas Sensibles al Contexto*, *o Dependientes del Contexto*): en la parte izquierda, X , puede haber cualquier cadena con símbolos no-terminales y en la parte derecha, Y , cualquier cadena, tan larga o mayor que la que aparece en la parte izquierda. Según esto, tanto la parte izquierda como la derecha tienen que tener una parte común de símbolos. Sólo admite la cadena vacía en el axioma $(S::=\lambda)$. Además, dependen del contexto en el sentido de que se tiene en cuenta lo que está antes y después del símbolo o la cadena que se sustituye. Las restricciones de las reglas serían del tipo:

$S \rightarrow \lambda$
$A \rightarrow a\ B\ b$
$a\ B \rightarrow b\ c\ d$
$a\ C\ d \rightarrow e\ B\ h$

- *Tipo 2* (*Gramáticas Libres*, *o Independientes del Contexto*): en la parte izquierda, X , debe haber solo un símbolo no-terminal. En la parte derecha, Y , puede haber cualquier cadena, incluida la cadena vacía. Además, son independientes del contexto por lo que no se tiene en cuenta lo que hay antes y después de la cadena que se sustituye. La sintaxis de los lenguajes de programación se suelen especificar por medio de este tipo de gramáticas. Las producciones de este tipo serían:

$S \rightarrow A$
$A \rightarrow a\ B\ b$
$B \rightarrow b\ c\ d$
$C \rightarrow \lambda$

- *Tipo 3* (*Gramáticas Regulares o Lineales*): en la parte izquierda, X, debe haber un solo símbolo no-terminal, mientras que en la parte derecha, Y, puede haber un terminal seguido de un no-terminal, o un solo terminal, o la cadena vacía. Además, pueden ser *Lineales por la Izquierda* si el símbolo no-terminal aparece el primero en la parte derecha $(A ::= Bc)$, o *Lineales por la Derecha*, si el símbolo no-terminal aparece el último en la parte derecha $(A ::= cB)$. Las reglas podrían tener la siguiente forma:

$$S \rightarrow \lambda$$
$$A \rightarrow a\ B$$
$$B \rightarrow c$$
$$C \rightarrow D\ h$$
$$D \rightarrow \lambda$$

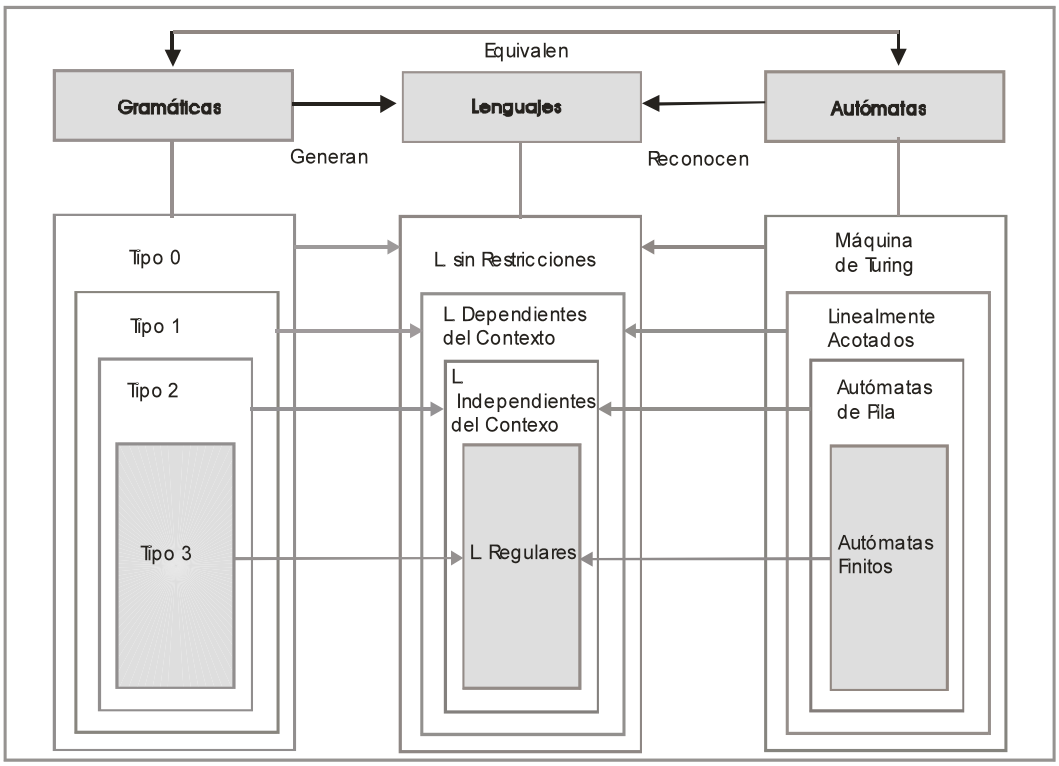


Fig. 2.5: Equivalencia de Gramáticas y Autómatas

Las gramáticas anteriores son capaces de producir lenguas *enumerables recursivamente* pero no todas las gramáticas son capaces de producir *lenguas recursivas*. En relación con esto, las *Gramáticas Regulares* generan lenguajes *enumerables recursivamente*, es decir, disponen de mecanismos capaces de enumerar las secuencias de un lenguaje natural una tras otra aunque sean infinitas, pero no son capaces de generar *lenguajes recursivos* porque se basan en un formalismo con muchas restricciones.

Para el reconocimiento de *patrones lingüísticos* se podrían utilizar las gramáticas y los mecanismos más *poderosos*, sin embargo, vamos a utilizar el formalismo *menos expresivo*, *Gramáticas Regulares*, y el mecanismo *más débil*, *Autómatas de Estado-Finito*, porque consideramos que para nuestro objetivo los sistemas más sencillos son los más adecuados y eficaces. El problema surgirá cuando tengamos que representar algunos fenómenos lingüísticos recursivos, como son determinadas estructuras de Sintagmas Nominales, y este tipo de mecanismo sea incapaz de reconocerlos. A pesar de esta limitación, vamos a adoptar este formalismo y para los problemas de recursividad propondremos una solución *aceptable*, que se desarrollará en un capítulo posterior.

Capítulo 3

TÉCNICAS DE ESTADO-FINITO: AUTÓMATAS Y TRANSDUCTORES

El PLN se ha servido de métodos provenientes de los lenguajes artificiales, específicamente de los Lenguajes de Programación, que parten de la Teoría Formal de Lenguajes y de la Teoría de la Compilación. Sin embargo, el lenguaje natural es mucho más difícil de abordarr que el lenguaje formal, por esta razón no se puede hacer una mera adaptación de las técnicas de tratamiento de los lenguajes artificiales. La teoría de los lenguajes formales constituye sólo un marco de trabajo que permite tratar problemas propios del lenguaje natural pero no basta para darles respuesta, ya que los procesos implicados en éste son más complejos y requieren el manejo de una gran cantidad de conocimiento de distinta índole. Por esta razón, el estudio de los lenguajes naturales y formales se circunscribe a disciplinas diferenciadas.

A pesar de lo anterior, se pueden señalar importantes similitudes: de forma general, tanto los lenguaje naturales como los formales están constituidos por un conjunto de símbolos perteneciente a un alfabeto, o vocabulario. Los símbolos se combinan formando cadenas del lenguaje según determinadas reglas, o gramática, que describen las distintas posibilidades de combinación de esos símbolos. No obstante, son muchos los aspectos en que los lenguajes naturales difieren de los formales, entre otros los que están relacionados con el origen de la gramática: los lenguajes formales se rigen por especificaciones gramaticales, o reglas establecidas de forma previa, desarrolladas con un objetivo concreto, mientras que las estructuras gramaticales que explican las diversas combinaciones de palabras o sentencias en los lenguajes naturales se crean a partir de su uso, y no antes. Aún así, las afinidades entre ambos lenguajes han propiciado la utilización de técnicas y métodos parecidos para su descripción.

Con el objetivo de caracterizar tanto los lenguajes naturales como los formales se ha adoptado un método denominado *Gramática Formal*, *Gramática de Estructura de Frase o Gramática Sintagmática, Phrase Structure Grammar* (PSG), que pretende describir la estructura de las frases o sentencias de ambos lenguajes. Como ya se ha expuesto, los componentes de esta gramática básicamente lo integran: *símbolo inicial* denominado axioma de la gramática, *símbolos del alfabeto* denominados vocabulario terminal, *símbolos especiales* denominados no-terminales que se utilizan para especificar las distintas fases de generación de palabras y, por último, *reglas o conjunto de producciones* que permitirán transformar los símbolos no-terminales en palabras del lenguaje–.

Una gramática consiste en un conjunto de reglas, también denominadas *producciones*, que permiten derivar cadenas del lenguaje concreto por medio de la sustitución de unas cadenas por otras, a partir de la aplicación repetida de las distintas producciones. Así, el lenguaje generado por una gramática específica se define en función de la aplicación de las reglas de producción y el *parser*, que realiza la función de reconocimiento del lenguaje, sólo puede determinar la estructura de las cadenas de ese lenguaje a partir de la gramática que se haya definido para él, es decir, sólo podrá reconocer las cadenas de palabras que pertenezcan al

lenguaje generado por dicha gramática. En este orden de cosas, esta es la razón por la cual la mayor parte de las aplicaciones del PLN necesitan un módulo de *parsing*, o programa de análisis gramatical, que se define como una función que recibe como *entrada* gramáticas y cadenas de palabras y devuelve como *salida* la estructura que dicha gramática asigna a las cadenas, lo que se conoce propiamente dicho como *parsing*, o análisis gramatical.

De acuerdo con lo anterior, según la cantidad de lenguajes que el formalismo gramatical pueda generar se valorará como más o menos expresivo, o poderoso. Si el objetivo es procesar lenguajes naturales por ordenador, será necesario crear un programa que acepte una gramática y una cadena de símbolos terminales y resuelva si la secuencia de cadenas responde a una regla de producción, esto es, si pertenece a un lenguaje concreto. Sin embargo, no se puede escribir un programa que acepte cualquier gramática de forma arbitraria, por esta razón se tienen que establecer limitaciones, es decir, es obligatorio *imponer restricciones al formalismo gramatical* (Grishman 1986) que permitan crear programas capaces generar análisis del lenguaje específico. Si el objetivo es construir un compilador para un lenguaje formal, o de programación, será necesario construir una máquina abstracta que acepte una gramática y una cadena de símbolos y decida si tal cadena pertenece, o no, al lenguaje, en otras palabras, un mecanismo que reconozca todas las sentencias pertenecientes a ese lenguaje específico. Y esto tampoco se puede hacer sin imponer restricciones, porque no existe ningún dispositivo teórico, o autómata, que pueda reconocer cualquier lenguaje.

La forma de clasificar los formalismos gramaticales de acuerdo con el conjunto de lenguajes que pueden representar, o según su denominada *capacidad expresiva*, se estableció en la mencionada *Jerarquía de Chomsky* (Chomsky 1957). Según esta jerarquía existen cuatro clases de formalismos que difieren sólo en la forma de las reglas de producción, cada clase se dispone en un orden de inclusión que va de menor a mayor capacidad para definir lenguajes, de forma que existe una correlación entre menor capacidad-mayor restricción gramatical y mayor capacidad-menor restricción gramatical: *Gramáticas Regulares* (Tipo 3) $\subseteq$

Gramáticas Libres de Contexto (Tipo 2) $\subseteq$ *Gramáticas Dependientes del Contexto* (Tipo 1) $\subseteq$ *Gramáticas Sintagmáticas* (Tipo 0).

Los lenguajes generados por las Gramáticas Tipo 3 se denominan Lenguajes Regulares –frente a la mayor cantidad de lenguajes generados por Gramáticas de Estructura de Frase, o por Gramáticas Libres de Contexto–. Con este presupuesto, tenemos que la identificación de determinadas cadenas terminales pertenecientes a un dominio es una tarea fundamental de algunas de las aplicaciones de la Lingüística Computacional, como es la Recuperación y Extracción de Información, por esa razón es imprescindible contar con algún mecanismo que las genere y reconozca. Ese mecanismo suele ser un *parser* o analizador léxico y sintáctico, de tal forma que en las aplicaciones prácticas se establece la siguiente relación: cuanto menos restrictiva es una gramática más complejo es su *parser* y cuanto más restrictiva es una gramática más sencillo es su *parser*.

En relación con lo anterior, si partimos de que cada gramática de un tipo es también del tipo anterior y de que tanto las gramáticas como los autómatas son en realidad los dos extremos de un mismo mecanismo que se usa como punto de referencia para tratar dos aspectos distintos de la misma lengua. Nuestro objetivo práctico hará que nos centremos en gramáticas *poco expresivas*, como son las de Tipo 3, que pueden ser reconocidas por autómatas sencillos, como son los de estado-finito –aunque este proceso se podría realizar por medio de autómatas más complejos que reconocen gramáticas menos restrictivas–.

Por otra parte, los autómatas, o programas empleados básicamente en los compiladores de lenguaje de programación, se suelen ordenar de acuerdo a tres paradigmas (Floyd y Beigel 1993), a los que se puede añadir un componente probabilístico:

- Un aceptador, *acceptor*, o programa no determinista que toma una cadena de *input* y la acepta, o no. Se usa esencialmente para comprobar la pertenencia de una cadena a un lenguaje.

- Un reconocedor, *recognizer*, o programa determinista que toma una cadena como *input* y la acepta, o la rechaza. Se emplea también para comprobar la pertenencia, o no, de una cadena a un lenguaje.
- Un transductor, *transducer*, o programa no determinista en el cual cada procesamiento completo equipara una cadena de *input* con una cadena de *output*. Se aplica fundamentalmente para procesar relaciones entre cadenas de dos lenguajes.

Los Autómatas de Estado-Finito pertenecen a la categoría de reconocedores de cadenas y los Transductores de Estado-Finito pertenecen a la categoría de transformadores de cadenas. Las diferencias entre ellos se basa en la forma adoptada por las transiciones que puede ser: como una *función*, o como una *relación*, tal y como se va a comprobar en lo que resta de capítulo en el que se van a exponer, además, los planteamientos básicos de la tecnología de Estado-Finito.

3.1. Introducción general a las Técnicas de Estado-Finito

Los Autómatas Finitos son los que reconocen menos lenguajes debido a la mayor restricción del formalismo gramatical que los genera. Pero esta debilidad para el tratamiento del lenguaje natural se convierte en una ventaja para determinadas aplicaciones prácticas, ya que uno de los mayores obstáculos para extraer información útil de los textos sin restricciones es la falta de cobertura de los sistemas que aplican técnicas de PLN: no hay *parsers* que puedan realizar un análisis completo de los textos reales, fundamentalmente por falta de información léxica, o por falta de reglas gramaticales, a lo que se suma los problemas que plantea la ambigüedad lingüística.

Sin embargo, los reconocedores de estado-finito admiten mayores posibilidades de análisis de frases que las *Gramáticas Libres de Contexto* porque aplican más restricciones sintácticas y,

por tanto, son más fáciles de construir. Además, son más eficaces y tolerantes a los fallos que otras gramáticas, por esta razón no se consideran las técnicas más adecuadas para el PLN (Chomsky 1957). La observación de Chomsky se basa la consideración de que el lenguaje natural es demasiado complejo para que se pueda describir por medio de Gramáticas Regulares de Estado-Finito, esto explica que las técnicas de estado-finito no hayan tenido hasta ahora un uso generalizado en el PLN. Pero, precisamente, esa característica hace que sea una modelo especialmente eficaz en los sistemas de RI, cuyo objetivo es analizar textos reales sin límite de extensión, en los que cabe la posibilidad de que surjan tanto palabras desconocidas como frases no-gramaticales. A esto se añade que, según Roche y Schabes (Roche y Schabes 1995): *a)* se pueden construir de forma determinista, y *b)* permiten reducir la complejidad del PLN .

El modelo de dominio va a controlar los mecanismos de estado-finito con el objetivo de que se obtengan los resultados esperados. El éxito de estos reconocedores depende de forma fundamental de la habilidad que se tenga para expresar suficiente conocimiento léxico y de dominio. Mientras que *parsers* o analizadores más potentes tienden a controlar principalmente restricciones lingüísticas, los reconocedores de estado-finito obedecen básicamente a restricciones léxicas para seleccionar la mejor interpretación de un texto. En dominios limitados, estas restricciones son parte del modelo de dominio. En dominios más amplios, el éxito en la interpretación, aplicando técnicas de procesamiento superficial de frases, depende más de los datos léxicos que del conocimiento de dominio (Jacobs 1995), esos datos se podrían obtener de un *corpus* usando métodos estadísticos (Church et al. 1991).

Según lo anterior, es posible desarrollar una gramática que describa un subconjunto del lenguaje natural en un dominio de conocimiento y que la reconozca cualquier mecanismo de estado-finito –como un autómata o un transductor– con el objetivo de reconocer únicamente segmentos o fragmentos concretos de los documentos. La idea fundamental es utilizar analizadores parciales, o fragmentales, denominados *chunkers* (Abney 1996), a partir de un pequeño grupo de reglas gramaticales con el objetivo de detectar Expresiones léxicas y Sintagmas Nominales, y con la función de que se puedan utilizar en la generación de índices

de los sistemas de RI. Además, otro argumento a favor de los mecanismos de estado-finito es que no requieren un análisis *robusto* de textos, esto es, se pueden analizar las secuencias sin necesidad de que la sintaxis y las reglas se hayan limitado totalmente, lo cual demandarían por otra parte un gran volumen de información léxica y gramatical.

Con el planteamiento anterior, adoptaremos la perspectiva de análisis según la cual determinados métodos del lenguaje formal son apropiados para tratar aplicaciones concretas relacionadas con la gran complejidad del lenguaje natural. En consecuencia, al considerar este paralelismo adoptamos un procedimiento eficaz para la descripción de los procesos implicados en aspectos parciales del PLN, como es el reconocimiento de patrones léxicos y sintácticos. Esta consideración no es nueva, las representaciones de estado-finito se aplican, como se vio anteriormente, a muchos niveles del PLN: análisis fonológico (Johnson 1972), (Kaplan y Kay 1994), análisis morfológico (Koskenniemi 1983), (Karttunen, Kaplan y Zaenen 1992) y *parsing* sintáctico (Abney 1996) (Roche 1993). Esta aproximación constituye para algunos investigadores el *elemento clave* para el desarrollo de métodos de interpretación de textos en la actualidad (Jacob et al. 1993; Pereira 1990).

El vínculo que se establece entre los Lenguajes Regulares y los Autómatas Finitos (AF) es de interés fundamental para la mayoría de aplicaciones que requieren *técnicas de emparejamiento de patrones* (Brookshear 1993). El emparejamiento o la equiparación de patrones, *patterns matching*, consiste en localizar las frecuencias de un patrón en un fichero de texto. El problema está en encontrar en un fichero de texto (t), un patrón, cadena (o conjunto de cadenas) de símbolos determinada construida sobre un determinado alfabeto, que se puede especificar de varias formas, como puede ser por medio de una Expresión Regular. A su vez, el conjunto de cadenas constituye el *lenguaje* construido sobre ese determinado alfabeto. En la búsqueda de patrones se puede usar un mecanismo que procese todas las cadenas del fichero hasta que identifique el patrón. Ese mecanismo puede ser un AF al cual se asociada una Expresión Regular, que le permitirá reconocer determinadas cadenas o patrones.

La importancia que tienen tanto los Lenguajes y como las Expresiones Regulares se debe a su aplicación práctica en la *construcción de analizadores léxicos*, es decir, aquellos *programas* que analizan un texto y *extraen las unidades léxicas* que aparecen en el mismo (Kelley 1995), además de su uso en analizadores sintácticos. Por su parte, los AF se suelen emplear en cuestiones en las que intervengan búsqueda, identificación y reconocimiento de cadenas de caracteres –esta característica hace que se utilicen en la compilación de programas de ordenador–. De esta forma, por una lado las Expresiones Regulares se usan con el objetivo de *especificar las unidades léxicas presentes en un lenguaje de programación*, y, por otro lado, los AF asociados a estas Expresiones se destinan a *reconocer dichas unidades*, también denominadas *componentes léxicos* (Kelley 1995).

El emparejamiento, o equiparación, de patrones basados en el uso de Autómatas de Estado-Finito se realiza en virtud de la capacidad de éstos para especificar un Lenguaje como el conjunto de cadenas que los hacen pasar del estado inicial a cualquiera de los denominados estados finales, o de aceptación. Una Maquina o Autómata de Estado-Finito, *Finite-State Machine* (FSM), o *Finite-State Automata* (FSA), consiste en un conjunto de estados y un conjunto de transiciones de un estado a otro según reciben los símbolos de entrada seleccionados de un alfabeto, o vocabulario Σ. Este mecanismo se puede clasificar en *Autómatas Finitos Deterministas* (AFD) y *Autómatas Finitos No-Determinista* (AFND). La diferencia básica entre ambos se basa en la capacidad para cambiar de estado dependiendo de la entrada, y del estado en el que se encuentre en ese momento.

En los AFD, a partir de un símbolo de entrada, se produce sólo una transición a otro estado, de esta forma se puede determinar a qué estado se va a llegar. En los AFND, a partir de un símbolo de entrada, se pueden producir una, ninguna o varias transiciones de un estado a otro, en consecuencia no se puede determinar cuál será el estado siguiente. Aún así, como se verá más adelante, se puede establecer una equivalencia entre ambos autómatas. De cualquier forma, tanto uno como otro constituyen un dispositivo para el reconocimiento de cadenas a través del patrón, o Expresión Regular vinculada a ellos.

Por consiguiente, las Expresiones Regulares y los AF aportan dos medios para especificar o definir lenguajes. Las Expresiones Regulares proporcionan una *plantilla o patrón para las cadenas del lenguaje*, a su vez todas las cadenas se corresponden con un patrón en particular, y estas cadenas serán *las únicas que formarán dicho lenguaje* (Kelley 1995). Por su parte, un AF suministra el otro medio para especificar un lenguaje como el conjunto de todas las cadenas que le permitirán transitar del estado inicial a uno de sus estados de aceptación. En otras palabras, un AF es de alguna forma un agrupador de cadenas potenciales, que después de someterse al procedimiento de análisis del autómata quedarían dividas en dos: *cadenas aceptadas*, o *no aceptadas*. Pero antes de entrar en los detalles de este proceso es necesario profundizar en las nociones básicas de Lenguajes y Expresiones Regulares.

Tanto los autómatas como los transductores reconocen Lenguajes Regulares manipulando conjuntos de cadenas. Una cadena se construyen a partir de un alfabeto, que está formado por un conjunto finito de símbolos. Si asumimos que una cadena $u \in \Sigma^*$ (conjunto de símbolos o caracteres incluyendo la palabra vacía λ), que los conjuntos de cadenas dan lugar a lenguajes $L_1 \subset \Sigma^*$ y $L_2 \subset \Sigma^*$ (con los cuales se pueden realizar las operaciones básicas de *unión*, *intersección*, *complementación*, *concatenación*, o *clausura de Kleene*), y que una Expresión Regular se forma a partir de un alfabeto Σ y se define aplicando un conjunto finito de reglas de la forma siguiente (Hopcroft y Ullman 1979):

- $\varnothing$ *es una Expresión Regular*.
- $\forall\ x \in \Sigma$, *x es una Expresión Regular*.
- Si p y q son Expresiones Regulares, $p + q$ es una Expresión Regular.
- Si p y q *son Expresiones Regulares,* $p \cdot q$ *es una Expresión Regular (*o pq*)* .
- Si p *es una Expresión Regular,* p^* *es una Expresión Regular (*o *clausura de Kleene, formada por todas las potencias del lenguaje incluyendo la potencia cero que da lugar a la palabra vacía), y por tanto* λ*, o palabra vacía, es una Expresión Regular*.

Tendríamos que, dado un alfabeto Σ, cada Expresión Regular, r, representa de forma sintetizada un Lenguaje Regular $L(r)$ –es decir, cada Expresión Regular permite establecer cómo se genera el conjunto de cadenas perteneciente al Lenguaje Regular– que se define recursivamente aplicando las siguientes reglas:

- Si $r = \varnothing$, entonces $L(r) = \varnothing$
- Si $r = \lambda$, entonces $L(r) = \{\lambda\}$
- Si $r = x$ entonces $L(r) = \{x\}$
- Si r y s son Expresiones Regulares que denotan los lenguajes L_1 y L_2 respectivamente, entonces $(r + s), (rs)$ y (r^*) son Expresiones Regulares que constituyen los Lenguajes Regulares $L_1 \cup L_2$, L_1L_2 y L^* respectivamente.

Para saber si una **Cadena** corresponde o no al ***Lenguaje*** representado por una ***Expresión Regular***, y establecida sobre un ***Alfabeto***, se podría proceder de dos formas: *a)* generando una lista con todas las cadenas del lenguaje que se compararía o equipararía con la cadena dada; o *b)* construyendo un autómata. De tal forma que, si r es una Expresión Regular, entonces hay un Autómata Finito, A, que acepta r, produciéndose la siguiente equivalencia (Hopcroft y Ullman 1979): $L(r) = L(A)$. En otras palabras, el Lenguaje representado por una Expresión Regular, $L(r)$, equivale sólo a la colección de cadenas pertenecientes al Lenguaje reconocido o aceptado por el autómata, $L(A)$.

En el caso de que se quisiera determinar si una cadena sencilla, como `analizar`, pertenece o no al lenguaje representado por la Expresión Regular `(anali)(zar + sis + tico)`, que sería del tipo $a(b+c+d)$, construida sobre un alfabeto integrado por todos los símbolos de una lengua, $\Sigma = \{a, á, b, c, d, e, é, f, g, h, \ldots\}$, o por un alfabeto que integrado por combinaciones de símbolos, como formas canónicas y terminaciones flexivas, $\Sigma = \{\text{anali}, \text{constru}, \text{asocia}, \text{-zar}, \text{-ir}, \text{-ar}, \text{-sis}, \text{-tico}, \ldots\}$, se procedería del modo siguiente:

1. Especificando una lista con todas las cadenas que genera el lenguaje formado a partir del alfabeto, $L = (\{anali\})(\{zar\} \cup \{sis\} \cup \{tico\})$, esto es, $L = \{analizar, análisis, analítico\}$.

2. Construyendo un *AF que permita diseñar algoritmos* de reconocimiento de patrones léxicos. De tal forma que el autómata reconocerá la pertenencia de las cadenas al Lenguaje Regular si los símbolos del alfabeto con los que están etiquetados las transiciones originan una serie de cambios de estado que le llevan de un estado inicial a un estado de aceptación (Fig. 3.1).

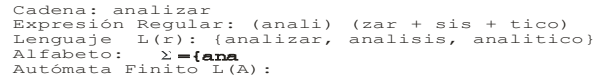


Fig. 3.1: Equivalencia de $L(r) = L(A)$

En consecuencia, dada una Expresión Regular (r), que representa distintos patrones, existen dos métodos para realizar la operación de equiparación de patrones, *pattern matching*, que requieren una etapa de pre-procesamiento consistente en la construcción de un autómata que represente el conjunto de patrones descrito por la Expresión Regular (Mohri 1997): el primer método se basa en la construcción de un AFD mientras que el segundo se basa en la elaboración de un AAFND. Estos autómatas se usarían para reconocer las frecuencias de los patrones en un texto (t) –un problema similar al de la equiparación de cadenas, *string*

matching, consistente en buscar la frecuencia de una palabra (x) en un texto (t)–. Los próximos apartados van a estar dedicados a caracterizar en términos más formales el desarrollo de los autómatas y su relación con los analizadores léxicos, así como su vinculación a las Gramáticas Regulares.

3.2. Autómatas Finitos Deterministas (AFD)

Un Autómata Finito Determinista (AFD), *Finite-State Automata* (FSA), también denominado Autómata Finito Determinista Acíclico, *Deterministic Acyclic Finite-State Automata* (DAFSA), se define como una colección o tupla de cinco elementos (Hopcroft y Ullman 1979):

$$\text{AFD} = (\Sigma, \text{Q}, f, \text{s}, \text{F})$$

donde

- Σ es el alfabeto de símbolos finito de entrada
- Q es el conjunto finito de estados
- f es la función de transición entre los estados definida como:

$$f: Q \times \Sigma \rightarrow Q$$

- s es el estado inicial y pertenece a Q $(s \in Q)$
- F es el conjunto de estados finales y un subconjunto de Q, $(F \subseteq Q)$

El lenguaje que acepta esta máquina está integrado por conjuntos de cadenas formadas por la concatenación de símbolos (caracteres, o palabras) extraídos de un vocabulario o alfabeto finito. El proceso de reconocimiento de símbolos funciona de la siguiente forma: las cadenas de entrada al autómata se analizan como una secuencia de símbolos; la fuente de esta secuencia se denomina *flujo de entrada* (Brookshear 1993). Según vaya llegando cada

símbolo del *flujo de entrada*, el proceso de reconocimiento desencadena un cambio de un estado a otro, a partir de un número finito de estados, o una continuación en el estado actual. El autómata finito se fundamenta en un mecanismo que puede estar en cualquiera de los estados finitos, uno de los cuales es el estado inicial y, por lo menos, uno es el estado final, o de aceptación. Si el autómata está en alguno de los estados finales, se dice que la máquina ha aceptado la cadena de símbolos, o que la cadena pertenece al lenguaje que la máquina reconoce.

El cambio de un estado a otro está determinado por unaa *función de transición*, es decir, los autómatas tienen la capacidad para detectar los símbolos según vayan llegando y basándose en el estado actual y el símbolo recibido ejecutar una transición de un estado a otro. A su vez, un *mecanismo de control* de la máquina determina qué transiciones se deben ejecutar al recibir un símbolo de entrada, ya que está programada para conocer cuál debe ser el nuevo estado en función de la combinación del estado actual y del símbolo de entrada (Brookshear 1993). Conforme a la descripción anterior, un AFD sólo puede generar un tipo de salida: *aceptación*, si se llega a un estado del conjunto F , o *no aceptación* si no se llega a un estado del conjunto F . Esto se puede representar de dos formas distintas, por medio de un *diagrama de transiciones*, o por medio de una *tabla de transiciones*.

1) Un *diagrama o gráfico de transición* –también denominado gráfico de transiciones (GF) o *red de transiciones*– está compuesto de tres elementos (Cohen 1991) :

- Un conjunto finito de estados en el que al menos uno se designa como estado inicial y algunos de los cuales se designan como estados finales.
- Un alfabeto Σ de posibles símbolos que forman las cadenas de entrada.
- Un conjunto finito de transiciones que describen cómo pasar de un estado a otro conforme se leen las sub-cadenas específicas de los símbolos de entrada. Estas transiciones se representan por medio de arcos etiquetados con los símbolos que permiten pasar de un estado a otro.

Los elementos anteriores se sitúan en un diagrama dirigido que se asocia a un AF, en el cual los vértices –representados a menudo por círculos– corresponden a los estados. Si se produce una transición de un estado a otro cuando se recibe un símbolo de entrada, se conectan ambos estados por medio de un arco, o flecha etiquetada, con el símbolo correspondiente a la cadena que se pretende analizar, o reconocer. Por tanto, en cada posición estará el estado que determine la función de transición compuesta por *estado actual- símbolo de entrada*.

Cada vértice del diagrama se etiqueta con la letra q_i , donde $q_i \in Q$ y donde $i = 0, 1, 2, 3, ...$ y cada arco se etiqueta con n, donde $n \in \Sigma$. La función de transición f se define para todos los pares (q_i, n) de $Q \times \Sigma$, e implica que sea cual sea el estado actual y el símbolo de entrada, siempre se tiene que producir el paso a un estado siguiente. Expresado de otra forma, hay *uno y sólo* un valor de función y por esta razón *el estado siguiente está totalmente determinado por la información que proporciona el par* (q_i, n) (Kelley 1995). Además, uno de los vértices se señala como el correspondiente a la posición inicial por medio de un apuntador $\rightarrow$, y al menos uno, o más vértices, se rodean de un doble círculo para designar las posiciones finales en las que se reconoce la cadena de entrada.

Con el objetivo de elaborar un diagrama que acepte Expresiones Regulares de un lenguaje podemos partir de la siguiente suposición: tenemos un alfabeto $\Sigma = \{a, b\}$ y un lenguaje asociado a este alfabeto $L = \{(a, b)^*\}$, formado por todas las potencias del lenguaje incluyendo la potencia 0, que da lugar a la palabra vacía λ. Para saber qué expresiones corresponden a un lenguaje, se construye un diagrama de transición que determina los miembros de ese lenguaje a partir de los elementos que se hayan definido para el autómata concreto, en este caso A.

Así, dado el AFD:

$$A = (\Sigma, Q, f, s, F)$$

en donde

$$\Sigma = \{a, b\}$$
$$Q = \{q_0, q_1, q_2, q_3\}$$
$$s = \{q_0\}$$
$$F = \{q_0, q_1, q_2\}$$

y, en donde la función de transición

$$f: Q \times \Sigma \rightarrow Q$$

se define como:

$$f(q_0, a) = q_0$$
$$f(q_0, b) = q_1$$
$$f(q_1, a) = q_0$$
$$f(q_1, b) = q_2$$
$$f(q_2, a) = q_0$$
$$f(q_2, b) = q_3$$
$$f(q_3, a) = q_3$$
$$f(q_3, b) = q_3$$

se crea el siguiente gráfico de transiciones (Fig. 3.2):

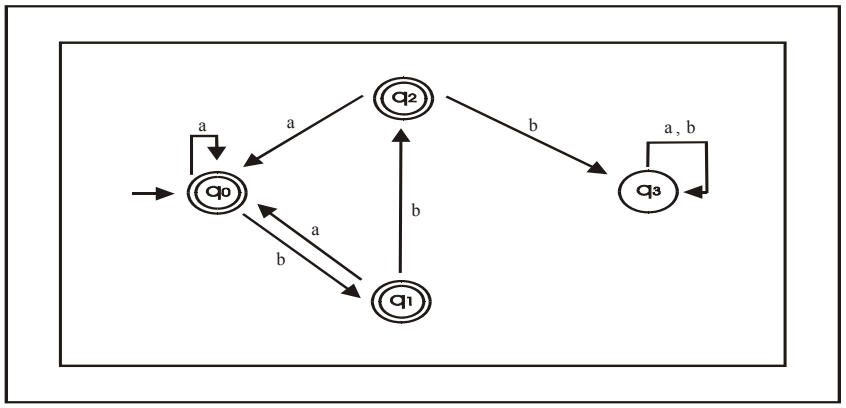


Fig. 3.2: Diagrama de transiciones del AFD A

2) Una *tabla de transiciones* se define como una *matriz bidimensional* (Brookshear 1993) cuyos elementos se toman de la información que proporciona la función de transición del autómata (Fig. 3.3). En las filas están los estados tomados de $q_i \in Q$, y en las columnas están los símbolos tomados de $n \in \Sigma$, de tal forma que en la posición (q_i, n) estará el estado que determine $f(q_i, n)$. Además, se debe señalar cuál es el estado inicial y qué estado, o estados, forman el estado final que permita realizar la salida de aceptación de la cadena de entrada, para ello normalmente el estado inicial se marca con el apuntador $\rightarrow$ y cada estado final va precedido del símbolo $*$.

f	a	b
$\rightarrow$ $*q_0$	q_0	q_1
$*q_1$	q_0	q_2
$*q_2$	q_0	q_3
q_3	q_3	q_3

Fig. 3.3: Tabla de transiciones del AFD A

Con la denominada *extensión de f a palabras* , se amplia la definición no sólo a los símbolos de entrada sino a cadenas, o palabras, de la siguiente forma recursiva: $f(q, aX) = f(f(q, a)\, X) \quad \forall\ a \in \Sigma, \forall\ X \in \Sigma^*$. Según esto, un autómata acepta una

cadena X si la ejecución de las funciones de transiciones correspondientes a los símbolos de X conducen de un estado inicial al estado final de aceptación. Se dice que una palabra $X \in \Sigma^*$ es reconocida o aceptada por el autómata A si $f(q_0, X) \in F$. A su vez, el lenguaje que acepta el autómata anterior se define como $L(A) = \{X \mid X \text{ es una cadena aceptada por A}\}$. En este caso, el lenguaje aceptado está formado por las cadenas siguientes $L(A) = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, baa, bab, ...\}$.

Si se quiere saber si una cadena, «*abaa*», pertenece, o no, al lenguaje aceptado por el automáta, se parte del circulo inicial, o estado inicial, y se comprueba que los símbolos de la cadena se corresponden con los arcos etiquetados que conducen del estado inicial a un estado de aceptación (Fig. 3.4). Si al finalizar el proceso se logra analizar todos los elementos de la cadena, en sus posiciones relativas, hasta llegar a un estado final, se dice que la cadena pertenece al lenguaje aceptado por el autómata. Si la palabra de entrada es la cadena anterior, el autómata iría transitando entre los estados siguientes:

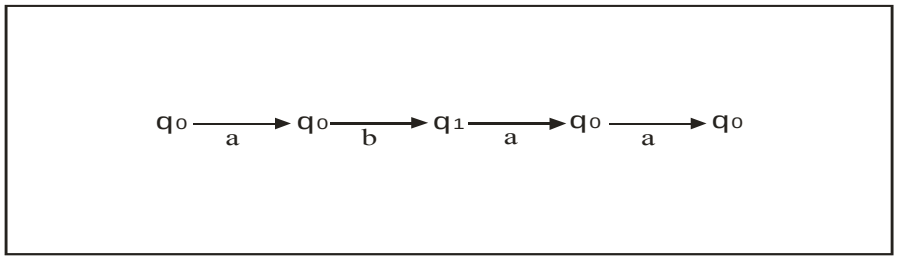


Fig. 3.4: Reconocimiento de la cadena *abaa*

En este caso, se puede afirmar que dicha cadena pertenece al lenguaje que el AFD A acepta. De forma análoga, siguiendo la información que aporta la tabla de transición se puede saber si el autómata acepta o no cualquier otra cadena.

Antes de continuar es preciso hacer la siguiente aclaración: este tipo de autómatas se dice que son *acíclicos*, esto es no contienen ciclos, y esto significa que no es posible llegar al mismo estado dos veces cuando ejecutan la función de transición, o dicho de otro modo, tras recibir

un mismo símbolo no es posible que transite a más de un estado. Aunque la mayoría de las veces se les denomina simplemente *Autómatas Finitos Deterministas*, y no *Autómata Finito Determinista Acíclico*. Por otra parte, y en virtud de esta característica, esta clase de autómatas tiene mucho interés en el PLN porque es el adecuado para los procesos de *reconocimiento de patrones*, tal y como veremos.

3.2.1. Equivalencia y minimización de AFD

Partiendo del supuesto de que el lenguaje aceptado por el AFD A se designa como $L(A)$ y está formado por el conjunto $\{X|\ f(q_i, X)$ está en $F\}$ (Hopcroft y Ullman 1979), y de que una cadena X es aceptada por un AFD $A = (\Sigma, Q, f, s, F)$ si $f(q_i, X) = p$, para algún $p \in F$ y $q_i \in Q$, se hace la siguiente interpretación: el lenguaje aceptado por el autómata estaría compuesto por todas las cadenas aceptadas por A, de tal forma que el conjunto de cadenas haría pasar al autómata de un estado inicial a un estado de aceptación, lo cual se expresa formalmente como:

$$L_{AFD} = \{X|\ X \in \Sigma^*\ \ y\ \ f(q_i, X) \in F\}$$

Si el AFD tiene como entrada la cadena *X* formada por $n_1 n_2 n_3$ y como estado inicial q_0, el estado final se logra por medio de la aplicación recursiva de $f(q_i, n)$ hasta llegar a un estado de aceptación:

$$f(f(f(q_0, n_1), n_2), n_3) \in F$$

Cuando en apartados anteriores se describía cómo un autómata determinista actuaba al analizar la cadena « *abaa* » de alguna forma se seguía el proceso siguiente:

$$f(q_0, abaa) = q_0$$

$$f\ f(q_0, a) = f(q_0, baa) = f(q_1, aa) = f(q_0, a) = q_0$$

En relación con esto, la equivalencia entre autómatas se define a partir de la equivalencia entre estados, esto es, si las transiciones entre estados de dos autómatas hacen el recorrido de un estado inicial a otro final, al leer una misma cadena, se puede decir que ambos autómatas son equivalentes. De tal forma que dos autómatas $A_1 = (\Sigma, Q, f, s, F)$ y $A_2 = (\Sigma, Q, f, s, F)$ son equivalentes si para una cadena $X \in \Sigma^*$ existen transiciones desde p, $p \in Q$ en el autómata A_1, y transiciones desde q, $q \in Q$ en el autómata A_2, que llegan a un estado final:

$$\forall\ X \in \Sigma^*, \quad f(p, X) = f(q, X)$$

En el caso de que las transiciones desde p y q con la cadena de entrada X lleguen a un estado final, se dice que pEq. A su vez, si $L(A)$ está formado por el conjunto de cadenas que originan que A pase de un estado inicial a otro de aceptación, se puede afirmar que los AFD A_1 y A_2 son equivalentes si $L(A_1) = L(A_2)$, y para ello se tiene que demostrar:

1. La *equivalencia entre los estados de un mismo autómata*, que se utiliza en la minimización de AFD.
2. La *equivalencia entre los estados de dos autómatas*, que se utiliza para comprobar la equivalencia entre autómatas.

En el primer caso, dado el AFD A_1 (Fig. 3.5):

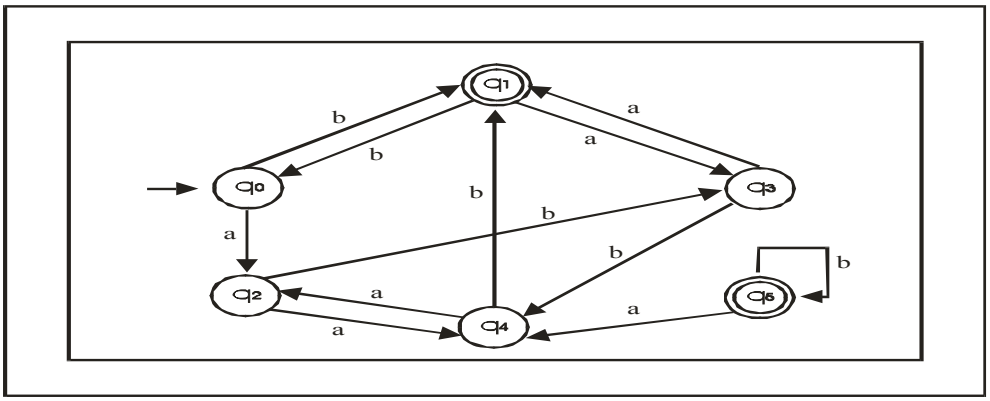


Fig. 3.5: Diagrama de Transiciones del AFD A_1

$$A_1 = (\{a,b\},\{q_0,q_1,q_2,q_3,q_4,q_5\},f,q_0,\{q_1,q_5\})$$

en donde f se define como:

$f(q_0,a)=q_2$
$f(q_0,b)=q_1$
$f(q_1,a)=q_3$
$f(q_1,b)=q_0$
$f(q_2,a)=q_4$
$f(q_2,b)=q_3$
$f(q_3,a)=q_1$
$f(q_3,b)=q_4$
$f(q_4,a)=q_2$
$f(q_4,b)=q_1$
$f(q_5,a)=q_4$
$f(q_5,b)=q_5$

En el AFD A_1, los estados equivalentes son $q_0 E q_4$ y $q_1 E q_2$ debido a que:

$f(q_0,a)=q_2 \notin F$ $\quad$ $f(q_4,a)=q_2 \notin F$
$f(q_0,b)=q_1 \in F$ $\quad$ $f(q_4,b)=q_1 \in F$

$f(q_1,a)=q_3 \notin F$ $\quad$ $f(q_2,a)=q_4 \notin F$
$f(q_1,b)=q_0 \notin F$ $\quad$ $f(q_2,b)=q_3 \notin F$

Sin embargo, se puede añadir una nueva distinción que está en relación con la longitud de las palabras de entrada $|X| \leq n$, esto quiere decir que dos estados pueden ser equivalentes pero cuando se restringe la equivalencia a una determinada longitud de palabras de entrada, $q_1 E_n q_2$ o $q_1 E_{n+1} q_2$, pueden dejar de serlo. Así, aunque $q_1 E q_2$ sean equivalentes dejan de serlo cuando la longitud de palabra es 2, $q_1 E_2 q_2$:

$$f(q_1, ab) = (f(q_1, a), b) = f(q_3, b) = q_4 \notin F$$
$$f(q_2, ab) = (f(q_2, a), b) = f(q_4, b) = q_1 \in F$$

Las relaciones de equivalencia entre estados permiten construir el denominado conjunto *cociente de estados*, Q/E, que estará compuesto por las distintas clases de equivalencias entre estados y cuya aplicación es fundamental para obtener el *autómata mínimo equivalente* a uno dado. La metodología para la obtención del conjunto cociente es la siguiente:

- Dividir el conjunto de estados en dos clases de equivalencia de longitud 0, definiendo el conjunto cociente Q/E en estados finales y no finales.
- Comprobar la clase de equivalencia en la que están incluidas los estados dentro de una misma clase, utilizando la equivalencia de longitud $n+1$. Si los estados son iguales, después de calcular el conjunto cociente, se agrupan en una clase de equivalencia, si no lo son se crea una nueva clase, y así sucesivamente.

Con el objetivo de demostrar la equivalencia entre un autómata y otro con un número mínimo de estados pero que reconozca el mismo lenguaje, $L(AFD) = L(AFD_m)$, se aplica el siguiente procedimiento:

1. El primer paso es suprimir los estados no accesibles desde el estado inicial porque no puede existir ninguna palabra que conduzca hacia él. Dado que un autómata con estados no accesibles desde el estado inicial tiene su equivalente en un autómata sin estados no accesibles desde el estado inicial –al que se denomina *autómata conexo*–,

si quisiéramos obtener el autómata sin estados no accesibles desde el inicial bastaría con eliminarlos.

2. El segundo paso es construir el *conjunto cociente* de los estados formado por las distintas clases de equivalencias de estados según el procedimiento anterior.

Con la aplicación del proceso anterior al AFD A_1 se obtiene el AFD mínimo A_1', en el que se puede comprobar que el número de estados no supone ningún problema para que los dos autómatas reconozcan el mismo lenguaje, cumpliéndose, por tanto, que son equivalentes, así:

- dado el autómata $A_1 = (\{a,b\}, \{q_0,q_1,q_2,q_3,q_4,q_5\}, f, q_0, \{q_1,q_5\})$ el autómata conexo equivalente lo forma $A_1' = (\{a,b\}, \{q_0,q_1,q_2,q_3,q_4\}, f, q_0, \{q_1\})$.
- dado el autómata conexo A_1' el *conjunto cociente* es $Q/E = [C_0 = \{q_1\}, C_1 = \{q_0,q_4\}, C_2 = \{q_2\}, C_3 = \{q_3\}]$
- el conjunto cociente se obtiene aplicando el procedimiento descrito más arriba, a su vez partiendo del autómata conexo $A_1' = (\{a,b\}, \{q_0,q_1,q_2,q_3,q_4\}, f, q_0, \{q_1\})$, se obtienen los siguientes conjuntos cocientes:

$$Q/E_0 = [C_0 = \{q_1\}, C_1 = \{q_0,q_2,q_3,q_4\}]$$

$Q/E_1 =$

$f(q_o, a) = q_2 \in C_1$ $\quad$ $f(q_2, a) = q_4 \in C_1$
$f(q_0, b) = q_1 \in C_0$ $\quad$ $f(q_2, b) = q_3 \in C_1$

$f(q_o, a) = q_2 \in C_1$ $\quad$ $f(q_3, a) = q_1 \in C_0$
$f(q_o, b) = q_1 \in C_0$ $\quad$ $f(q_3, b) = q_4 \in C_1$

$f(q_o, a) = q_2 \in C_1$ $\quad$ $f(q_4, a) = q_2 \in C_1$
$f(q_o, b) = q_1 \in C_0$ $\quad$ $f(q_4, b) = q_1 \in C_0$

$f(q_2, a) = q_4 \in C_1$ $\quad$ $f(q_3, a) = q_1 \in C_0$
$f(q_2, b) = q_3 \in C_1$ $\quad$ $f(q_3, b) = q_4 \in C_1$

$f(q_2, a) = q_4 \in C_1$ $\quad$ $f(q_4, a) = q_2 \in C_1$
$f(q_2, b) = q_3 \in C_1$ $\quad$ $f(q_4, b) = q_1 \in C_0$

$f(q_3, a) = q_1 \in C_0$ $\quad$ $f(q_4, a) = q_2 \in C_1$
$f(q_3, b) = q_4 \in C_1$ $\quad$ $f(q_4, b) = q_1 \in C_0$

según esto,
$Q/E_1 = [C_0 = \{q_1\}, C_1 = \{q_0, q_4\}, C_2 = \{q_2, q_3\}]$

$Q/E_2 =$

$f(q_0, a) = q_2 \in C_2$ $\quad$ $f(q_4, a) = q_2 \in C_2$
$f(q_0, b) = q_1 \in C_0$ $\quad$ $f(q_4, b) = q_1 \in C_0$

según esto,
$Q/E_2 = [C_0 = \{q_1\}, C_1 = \{q_0, q_4\}, C_2 = \{q_2, q_3\}]$

$Q/E_3 =$

$f(q_2, a) = q_4 \in C_1$ $\quad$ $f(q_3, a) = q_1 \in C_0$
$f(q_2, b) = q_3 \in C_2$ $\quad$ $f(q_3, b) = q_4 \in C_1$

según esto,
$Q/E_3 = [C_0 = \{q_1\}, C_1 = \{q_0, q_4\}, C_2 = \{q_2\}, C_3 = \{q_3\}]$

- Así, dado el conjunto cociente del autómata A_1', el autómata mínimo equivalente es:

$$A_{1m}' = (\{a,b\},\{C_0,C_1,C_2,C_3\},f,C_1,\{C_0\})$$

en donde f se define como:

$$\begin{array}{lll}
f(C_0,a)=C_3 & \acute{o} & f(q_1,a)=q_3 \\
f(C_0,b)=C_1 & \acute{o} & f(q_1,b)=\{q_0,q_4\} \\
f(C_1,a)=C_2 & \acute{o} & f(\{q_0,q_4\},a)=q_2 \\
f(C_1,b)=C_0 & \acute{o} & f(\{q_0,q_4\},b)=q_1 \\
f(C_2,a)=C_1 & \acute{o} & f(q_2,a)=\{q_0,q_4\} \\
f(C_2,b)=C_3 & \acute{o} & f(q_2,b)=q_3 \\
f(C_3,a)=C_0 & \acute{o} & f(q_3,a)=q_1 \\
f(C_3,b)=C_1 & \acute{o} & f(q_3,b)=\{q_0,q_4\}
\end{array}$$

La función de transición del AFD mínimo anterior, A_1', se puede representar en una tabla de transiciones (Fig. 3.6) o en una tabla equivalente en la se han renombrado los estados (Fig. 3.7), así como en un diagrama de transición (Fig. 3.8) que sería equivalente al de la Figura 3.5.

f	a	b
* C 0	C 3	C 1
→ C 1	C 2	C 0
C 2	C 1	C 3
C 3	C 0	C 1

Fig. 3.6: Tabla de transiciones del AFD_m A_1'

f	a	b
* $\{q_1\}$	$\{q_3\}$	$\{q_0, q_4\}$
→ $\{q_0, q_4\}$	$\{q_2\}$	$\{q_1\}$
$\{q_2\}$	$\{q_0, q_4\}$	$\{q_3\}$
$\{q_3\}$	$\{q_1\}$	$\{q_0, q_4\}$

Fig. 3.7: Tabla de transiciones del AFD_m A_1' con estados renombrados

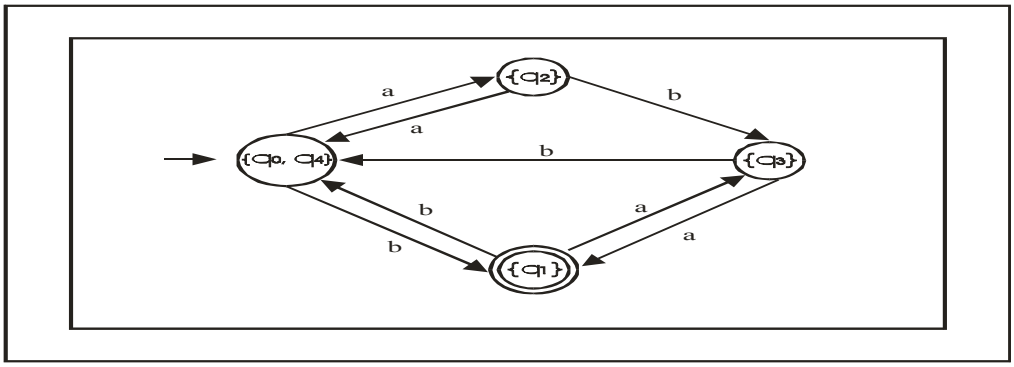


Fig. 3.8 : Diagrama de transiciones del AFD_m A_1'

3.2.2. Equivalencia de AFD y Gramáticas Regulares

Tanto los lenguajes formales como los naturales están integrados por conjuntos de cadenas que a su vez están constituidas por la concatenación de símbolos tomados de un conjunto finito. Los conjuntos de cadenas componen sintagmas y sentencias, también denominadas oraciones o cláusulas. El número de sentencias que se pueden construir en lenguaje natural es infinito, por tanto si quisiéramos definir las sentencias del lenguaje natural tendríamos que recurrir a algún procedimiento que nos permitiera describir esas sentencias. Existen dos procedimientos básicos para esta descripción que están en relación con la complejidad de los patrones con los que las sentencias se pueden equiparar (Kaplan 1995):

1. *Especificando una gramática*, definida por un conjunto de reglas de emparejamiento de patrones, *pattern-matching*, que se usarían tanto para producir sentencias del

lenguaje determinado como para reconocer si una cadena pertenece o no al lenguaje. Esta especificación utilizaría por tanto un formalismo basado en *reglas*.

2. *Especificando un autómata*, *o dispositivo conceptual*, que se utilizaría tanto para producir como para reconocer las sentencias u oraciones del lenguaje.

A su vez, existe una equivalencia entre Gramáticas, Autómatas y Lenguajes, de tal forma que el lenguaje asociado a un determinado autómata está constituido por el conjunto de todas las cadenas aceptadas, o reconocidas, por ese autómata. Los AF reconocen patrones que pertenecen a la categoría de los Lenguajes Regulares y éstos se describen por medio de Gramáticas Regulares. Como ya se ha mencionado, las cadenas o patrones que reconocen los AF se denominan Expresiones Regulares que representan de forma concisa Lenguajes Regulares y denotan, entre otras cosas, el orden en el cual los símbolos o cadenas se pueden combinar. La importancia de este fenómeno merece que nos detengamos en explicar, a continuación, cómo se produce la vinculación entre AF y Gramáticas Regulares (Tipo 3).

Los AF se emplean fundamentalmente para reconocer cadenas del lenguaje generado por una gramática o para generar cadenas o palabras de un lenguaje partiendo del estado inicial, con la cadena vacía, y logrando una cadena del lenguaje cuando se alcanza algún estado final. Para lograr las aplicaciones anteriores se tiene que comprobar la equivalencia entre el lenguaje generado por una gramática y el lenguaje reconocido por un autómata. Con esta finalidad, Brookshear (Brookshear 1993) establece la siguiente equivalencia, dado un alfabeto Σ :

$$\{L(G) \ : \text{G es una Gramática } \textit{Re}\text{gular de } \Sigma\} = \{L(A) \ : \text{A es una Autómata Finito de } \Sigma\}$$

A partir de aquí, se puede demostrar que partiendo de una Gramática Regular se puede construir un autómata que reconozca las palabras o expresiones del lenguaje generadas por dicha gramática, y a la inversa partiendo de un autómata, que genere cadenas de un lenguaje, se puede conocer la gramática que produce dichas cadenas. La forma de establecer las equivalencias en la siguiente:

I. Dada una Gramática Tipo 3 se construye el AF equivalente que reconozca las cadenas del lenguaje generado por dicha Gramática:

Partiendo de la Gramática $G_3 = (\Sigma_T, \Sigma_N, S, P)$ se diseña un AF:

$$AFD = (\Sigma, Q, f, q_0, F)$$

en donde

$\Sigma = \Sigma_T$ (Σ = colección de terminales de G_3)

$Q = \Sigma_N$ (Q = colección de no-terminales de G_3)

$q_0 = S$ (el estado inicial es igual al axioma de G_3)

$F =$ es la colección, o el conjunto de símbolos noterminales de G_3 que aparecen en el lado izquierdo de alguna regla $X::=\lambda$ ó $X::=a$ (se trataría de un nuevo símbolo no terminal que no pertenece a Σ_N)

y donde la función de transición, f , se define como una regla que relaciona tres elementos (P, a, Q) para la cual G_3 contiene una regla de producción (del tipo $P::=aQ$) de tal forma que:

Si $S::=bA$ entonces $f(S,b)=A$

Si $S::=\lambda$ entonces $f(S,\lambda)=F$

Si $B::=a$ entonces $f(B,a)=F$

Teniendo en cuenta lo anteriror, la función de transición del AF se define como:

Si $S::=bA$ entonces $f(S,b)=A$

Si $S::=c$ entonces $f(S,c)=F$

Si $A::=\lambda$ entonces $f(A,\lambda)=F$

Si $A ::= aB$ entonces $f(A, a) = B$

Si $B ::= bC$ entonces $f(B, b) = C$

Si $C ::= bA$ entonces $f(C, b) = A$

Si $C ::= cA$ entonces $f(C, c) = A$

El AF equivalente que reconoce el lenguaje generado por la gramática G_3 estará formado por una tupla con los siguientes elementos:

$$A_{G3} = (\{a, b, c\}, \{S, A, B, C, F\}, f, S, \{S\})$$

Este AF se puede representar en un *diagrama de transiciones* (Fig. 3.9), o en una *tabla de transiciones* (Fig. 3.10):

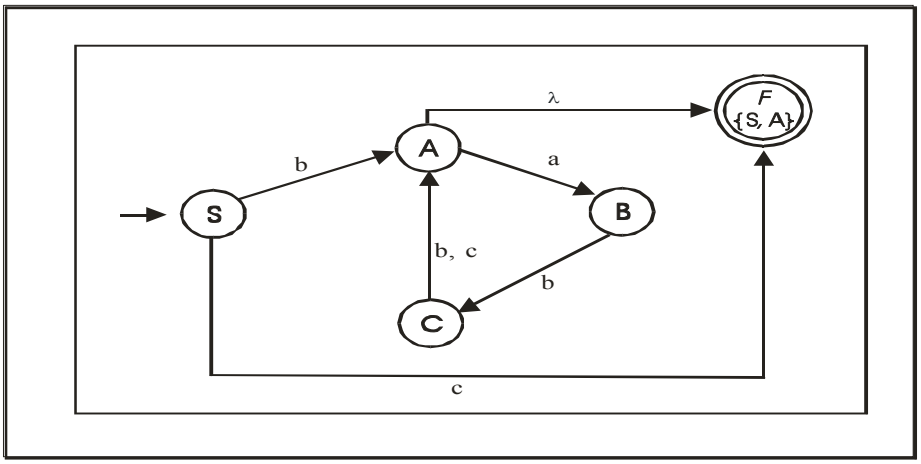


Fig. 3.9: Diagrama de transiciones correspondiente al AF A_{G3}

f	a	b	c	λ
→ **S**		**A**	***F***	
A	**B**			***F***
B		**C**		
C		**A**	**A**	
F				

Fig. 3.10: Tabla de transiciones correspondiente al AF A_{G3}

II. Dado un AF se construye la Gramática Regular equivalente que genere las cadenas del lenguaje empezando con la cadena vacía, en el estado inicial, y generando una cadena del lenguaje cuando se llega a un estado de final, o de aceptación:

Partiendo del AF $A=(\Sigma,Q,f,q_0,F)$ se define una Gramática Regular equivalente, así, dado el AF A donde:

$$\Sigma=\{a,b\}$$
$$Q=\{q_0,q_1,q_2,q_3\}$$
$$F=\{q_0,q_1,q_2\}$$

y donde la función de transición, f , se define como:

$$f(q_0,a)=q_0$$
$$f(q_0,b)=q_1$$
$$f(q_1,a)=q_0$$
$$f(q_1,b)=q_2$$
$$f(q_2,a)=q_0$$
$$f(q_2,b)=q_3$$
$$f(q_3,a)=q_3$$
$$f(q_3,b)=q_3$$

se construye la Gramática Regular equivalente $G_3=(\Sigma_T,\Sigma_N,S,P)$, donde:

$\Sigma_T = \Sigma$
$\Sigma_N = Q$
$S = q_0$

y, donde las reglas de producción, P, se construyen a partir de la función de transición del autómata, de forma tal que:

Si $f(q,a) = p$	entonces	$q ::= ap$
Si $f(q,a) = p$ y $p \in F$	entonces	$q ::= a$
Si $q_0 \in F$	entonces	$q_0 ::= \lambda$

Según lo anterior, las reglas de producción, P, quedarían como:

Si $f(q_0,a) = q_0$	entonces	$q_0 ::= a$
Si $f(q_0,b) = q_1$	entonces	$q_0 ::= b$
Si $f(q_1,a) = q_0$	entonces	$q_1 ::= a$
Si $f(q_1,b) = q_2$	entonces	$q_1 ::= b$
Si $f(q_2,a) = q_0$	entonces	$q_2 ::= a$
Si $f(q_2,b) = q_3$	entonces	$q_2 ::= bq_3$
Si $f(q_3,a) = q_3$	entonces	$q_3 ::= aq_3$
Si $f(q_3,b) = q_3$	entonces	$q_3 ::= bq_3$

La Gramática Regular G_3 equivalente al AF A estará formada por una tupla con los siguientes elementos:

$G_{3AFD} = (\{a,b\}, \{q_0,q_1,q_2,q_3\}, q_0, P)$

3.2.3. Autómatas Finitos No Deterministas (AFND)

Este nuevo modelo se diferencia de los AFD en que pueden admitir: *una*, *ninguna*, o *varias transiciones* entre estados por cada par $f(q_i, n)$. Además, puede desplazarse entre los estados sin necesidad de aceptar ningún símbolo (o, lo que es lo mismo, leyendo la palabra vacía), por medio de lo que se denomina transiciones-λ, es decir, $f(q_i, \lambda) = q_j$.

La definición formal es la siguiente (Hopcroft y Ullmann 1979): un AFND, al igual que un AFD, está compuesto de una tupla de cinco elementos:

$$\text{AFND} = (\Sigma, Q, f, s, F)$$

donde

- Σ es el alfabeto de símbolos finito de entrada
- Q es el conjunto finito de estados
- s es el estado inicial y pertenece a Q $(s \in Q)$
- F es el conjunto de estados finales y un subconjunto de Q, $(F \subseteq Q)$

pero donde

- La función de transición, f, es una relación sobre $(Q \times \Sigma) \times Q$ que se denomina *relación de transición.* Esto significa que una regla hace corresponder pares (q_i, n) con conjuntos de estados:

$$f(q_i, n) = \varnothing$$
$$f(q_i, n) = \{q_j\}$$
$$f(q_i, n) = \{q_j, q_k, \ldots\}$$

Según lo anterior, la función de transición, f, se define como una regla que relaciona pares (q_i, n) con ninguno, uno, o más estados:

$$f : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$$

de tal forma que 2^Q constituye el conjunto de todos los subconjuntos que se pueden formar con elementos de Q.

Al igual que los AFD, los AFND se pueden describir por medio de dos tipos de representaciones:

1) Diagrama de transiciones: si A' es un AFND y se quiere determinar si una cadena pertenece al lenguaje que acepta el autómata, $L(A')$, se puede diseñar un diagrama correspondiente a A'. El diagrama de un autómata no determinista está compuesto por los siguientes elementos:

- Un conjunto finito de estados, en el que al menos uno se designa como estado inicial y uno, o algunos, de los cuales corresponden a los estados finales. El estado inicial se puede indicar con una flecha entrante no etiquetada $(\rightarrow)$ y los nodos correspondientes a los estados finales se rodean de un doble círculo.
- Un alfabeto Σ de posibles símbolos que forman las cadenas correspondientes al lenguaje $L(\Sigma)$.
- Un arco etiquetado con $n \in (\Sigma \cup \{\lambda\})$ entre el vértice q_i y el vértice q_j si $q_j \in f(q_i, n)$.
- Un conjunto finito de transiciones que describen de forma no determinista cómo se pasa de un estado a otro conforme se leen las sub-cadenas específicas de los símbolos de entrada (incluida la cadena, o palabra vacía, λ). La diferencia con respecto a los autómatas deterministas es que para un par *estado actual-entrada*, (q_i, n), el autómata tiene la posibilidad de transitar a más de un estado, o a ninguno. Además, no existe nada en el diagrama que determine la elección y, en consecuencia, el procedimiento que sigue el autómata cuando lee las cadenas de

entrada no está determinado, por esta razón en el recorrido de reconocimiento el autómata puede elegir entre distintas posibilidades de análisis.

2) Tabla de transiciones que se representa de la misma forma que la de los AFD, con dos diferencias: en las posiciones (q_i, n) de las celdas deben aparecer los conjuntos que determine la relación de transición $f(q_i, n)$, incluyendo el conjunto vacío $\{\varnothing\}$, cuando no exista ninguna transición entre estados después de recibir la entrada correspondiente, y en las columnas deben aparecen los símbolos $n \in \Sigma \cup \{\lambda\}$.

Con la finalidad de diseñar un diagrama que acepte expresiones de un lenguaje, partimos de la siguiente suposición: tenemos un alfabeto o vocabulario $\Sigma = \{a, b, c\}$ y un lenguaje asociado a ese vocabulario $L = \{(a^* b^* \cup c)\}$ y queremos saber qué expresiones corresponden a ese lenguaje. Para ello, construimos, de forma intuitiva por ahora, un diagrama de transición que permita determinar los miembros de ese lenguaje según los elementos que se hayan definido para el autómata. Así, dado el autómata AFND A':

$$A' = (\Sigma', Q', f', s', F')$$

en donde

$$\Sigma' = \{a, b, c\}$$
$$Q' = \{q_0, q_1, q_2\}$$
$$s' = q_0$$
$$F' = \{q_0, q_1, q_2\}$$

y donde la función de transición, $f': Q' \times (\Sigma \cup \{\lambda\}) \to 2^Q$, se define como los conjuntos siguientes:

$$f'(q_0, a) = \{q_0, q_1\}$$
$$f'(q_0, b) = \varnothing$$
$$f'(q_0, c) = \varnothing$$
$$f'(q_0, \lambda) = \varnothing$$
$$f'(q_1, a) = \varnothing$$
$$f'(q_1, b) = \{q_1, q_0\}$$
$$f'(q_1, c) = \{q_2\}$$
$$f'(q_1, \lambda) = \{q_2\}$$
$$f'(q_2, a) = \varnothing$$
$$f'(q_2, b) = \varnothing$$
$$f'(q_2, c) = \varnothing$$
$$f'(q_2, \lambda) = \varnothing$$

Con estos datos se crea el correspondiente diagrama (Fig. 3.11) y tabla de transiciones (Fig. 3.12). La interpretación del diagrama es la siguiente: un AFND acepta una cadena X, si la ejecución de las funciones de transición correspondientes a los símbolos de X conducen de un estado inicial a un estado de aceptación.

El lenguaje que acepta el autómata, A', estará formado por las expresiones siguientes $L(A') = \{\lambda, a, aa, ab, abc, aab, aabb, aabc, aabbc, aaabbb, ...\}$. Si queremos saber si una palabra pertenece o no a este lenguaje, se parte del estado inicial y se recorre el trayecto entre los estados hasta consumir todos los elementos de la cadena, si se alcanza un estado final de aceptación se puede decir que la cadena pertenece al lenguaje que reconoce el autómata, o que es un nombre de variable aceptable por el autómata, como es el caso.

El rasgo que distingue estos autómatas de los deterministas es que la aceptación de palabras se produce existiendo más de una transición, o ninguna, entre estados para el mismo símbolo, y, por lo tanto, el modelo debe optar por una, o por no poder realizar ninguna transición. Esto da lugar no sólo a que la ejecución del análisis se realice mediante *indicios* o aproximaciones

sino que es necesario llevar a cabo una búsqueda exhaustiva por el diagrama. Por tanto, este análisis implica ciertas c*onjeturas,* que obedecen a una característica propia del no determinismo: *cuando se debe realizar una elección y dicha elección no puede ser determinada por el modelo*, se asume que siempre se hace la *elección correcta* (Kelley 1995). Este problema se resuelve por medio de un algoritmo muy útil en el que un AFND se transforma en un AFD, como se verá en el siguiente epígrafe.

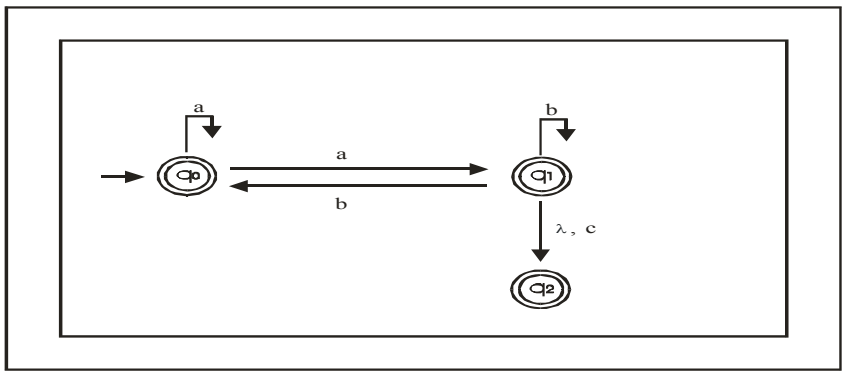


Fig. 3.11: Diagrama de Transiciones del AFND A′

f′	a	b	c	λ
→ *q_0	$\{q_0, q_1\}$	∅	∅	∅
*q_1	∅	$\{q_1, q_0\}$	$\{q_2\}$	$\{q_2\}$
*q_2	∅	∅	∅	∅

Fig. 3.12: Tabla de Transiciones del AFND A′

3.2.3.1. Equivalencia de AFND y AFD

Dos autómatas se consideran equivalentes si reconocen el mismo lenguaje. La equivalencia se puede demostrar entre dos AFD, o entre un AFND y un AFD, es decir, se puede demostrar que para cualquier AFND se puede construir un AFD equivalente que acepte el mismo

lenguaje (Hopcroft y Ullman 1979). Con este objetivo se pone en práctica un algoritmo que transforma un autómata no determinista en uno determinista que pone de manifiesto que el lenguaje aceptado por el primero también lo puede ser por el segundo.

Establecer la equivalencia entre autómatas de la misma clase constituye un proceso sencillo, basta que acepten el mismo lenguaje. Sin embargo, demostrar la equivalencia entre un AFND y un AFD es algo más complicado. Y, esto último, es lo que se pretende describir en este apartado: que todo AFND es un AFD, y que los lenguajes aceptados por los no deterministas incluyen los aceptados por los deterministas, por tanto, los AFND no definen o reconocen más lenguajes, es decir, no son más *potentes* (Kelley 1995). La utilidad del establecimiento de esta equivalencia se reflejará en los procesos de reconocimiento de patrones lingüísticos ya que contaremos con una aplicación computacional encargada de transformar los AFND en AFD.

Teniendo en cuenta que todo AFND tiene un AFD equivalente que acepta el mismo lenguaje se parte del siguiente planteamiento: cuando se introduce una cadena X en un AFND, ésta se acepta si comenzando por el estado inicial se llega a algún estado final por medio de la función de transición, $f(q_0, X) \cap F \neq \varnothing$. El lenguaje reconocido por este autómata estará formado por el conjunto de cadenas que a través de los estados del AFND le llevan desde el estado inicial a un estado final:

$$L_{AFND} = \{X \mid X \in (\Sigma \cup \{\lambda\}) \ y \ f(q_0, X) \in F\}$$

Si un AFND tiene como entrada la cadena X, formada por $n_1 n_2 n_3$ y como estado inicial q_0, el estado final se logra por medio de la aplicación recursiva de $f(q_i, n)$ hasta llegar a un estado de aceptación. Exactamente igual que procede un AFD, pero con la diferencia de que la función de transición, $f(q_i, n)$, se puede definir como un *conjunto de estados.* Si consideramos a C como el conjunto de estados pertenecientes a Q, la función de transición, $f(C, n)$, se establece como el conjunto de estados $\{p \mid q \in C \quad y \quad p \in f(q_i, n)\}$, que

formalmente se define como el conjunto de todos los estados a los que se accede a partir de C con la entrada n (Kelley 1995):

$$f(\mathrm{C}, \mathrm{n}) = \bigcup_{q \in C} f(q_i, n)$$

Por tanto, para el análisis de la cadena X se aplica de forma recursiva $f(\mathrm{C}, n)$:

$$f(f\ (f\ (\mathrm{C}, n_1), n_2), n_3)\ \in\ F$$

Hay que considerar que en los AFND se pueden desencadenar transiciones entre estados sin leer ningún símbolo en la entrada, mediante las transiciones vacías, o transiciones-λ, por tanto se han de tener en cuenta dichas transiciones tanto en el primer y último símbolo de la entrada como entre cada dos símbolos de la entrada. Por otra parte, la relación de transición-λ es reflexiva: $f(q_0, \lambda) \cdot q_0$ y se establece entre cada uno de los estados con ellos mismos, además de entre los pares de estados en los que se efectúa esta relación. A su vez, dicha relación de transición (RT), es también transitiva de tal forma que:

- Si $q_o\ RT\ q_1$ y $q_1\ RT\ q_2$ entonces $q_0\ RT\ q_2$

El conjunto RT del AFND A' estará formados por $\{(q_0, q_0), (q_1, q_1), (q_2, q_2), (q_1, q_2)\}$. Cuando se introduce una cadena como «*aabbbc*» al AFND A', éste procede de la siguiente forma:

- Primero se tiene que calcular el estado, o conjunto de estados, vinculados al estado inicial por medio de la relación reflexiva y transitiva si la hubiera:

$$f(q_0, \lambda) = \{q_0\}$$

- Después, se calcula el estado, o conjunto de estados, a los que transita el autómata desde el estado, o conjunto de estados, anterior cuando recibe el primer símbolo de la cadena:

$$f(q_0, a) = \{q_0, q_1\} \text{ por tanto } f(q_0, \lambda \cdot a) = \{q_0, q_1\}$$

y seguidamente, se calcula dónde transita el autómata desde cada uno de estos estados cuando recibe λ, además de $f(q_0, \lambda \cdot a)$:

$$f(q_0, \lambda) = \varnothing$$
$$f(q_1, \lambda) = q_2$$
$$\text{por tanto } f(q_0, \lambda \cdot a \cdot \lambda) = \{q_0, q_1, q_2\}$$

- A continuación, se calcula el camino que recorre el autómata desde cada uno de estos estados cuando recibe el siguiente símbolo:

$$f(q_0, \lambda \cdot a \cdot \lambda \cdot a) = \{q_0, q_1\}$$

y dónde llega el autómata con las siguientes transiciones- λ

$$\text{por tanto } f(q_0, \lambda \cdot a \cdot \lambda \cdot a \cdot \lambda) = \{q_0, q_1, q_2\}$$

- Después con el siguiente símbolo y así sucesivamente hasta consumir todos los símbolos de la cadena y alcanzar un estado que, en este caso, está en F por lo que se puede afirmar que la cadena « *aabbbc* » es reconocida por el AFND A':

$$f(q_0, \lambda \cdot a \cdot \lambda \cdot a \cdot \lambda \cdot b) = \{q_1, q_0\}$$
$$\ldots$$
$$\ldots$$
$$\ldots \qquad = \{q_2\}$$

El procedimiento anterior es la base para demostrar que el análisis de una cadena por parte de un AFND lo puede realizar también un AFD: de alguna forma un AFD es un AFND en el que

la función de transición, $f(q_i, n) = p$, sólo puede pasar a un estado p, para $\forall n \in \Sigma$, y en el que no hay transiciones-λ, de lo que se deduce que cuando no se consume ningún símbolo en la entrada el autómata determinista no transita, con lo cual la función de transición sobre la palabra vacía siempre da como resultado un conjunto vacío $f(q_i, \lambda) = \varnothing$, y no como sucede en los AFND en los que $f(q_i, \lambda) = p$. Según esto, para cada AFND $A' = (\Sigma', Q', f', s', F')$ existe un AFD $A = (\Sigma, Q, f, s, F)$ que reconoce el mismo lenguaje. Con el objetivo de demostrar la equivalencia entre estos autómatas se tienen que efectuar las siguientes correspondencias (Kelley 1995):

- Cada estado del AFD Q se debe corresponder con el conjunto de estados del AFND Q':

  $Q = 2^Q$

- El mismo proceso con el vocabulario:

  $\Sigma = \Sigma'$

- La función de transición f del AFD se tiene que definir de la misma forma que la función de transicón f' del AFND, de modo que permita transitar a un conjunto de estados:

  $f = \{f'\}$, es decir, $f(C, n) = \{C' \mid C' = \bigcup_{q \in C} f'(q_i, n)\}$

- Se debe relacionar también el estado inicial, es decir, el estado inicial del AFD se debe corresponder con el subconjunto de Q' que contenga el estado inicial del AFND:

  $s = \{s'\}$, es decir, $q_0 = f'(q_0, \lambda)$

- Los estados finales, o estado final, del AFD $F \in Q$ se deben corresponden con el conjunto de estados del AFND $\{F'\} \in Q'$, de tal forma que F constituya el conjunto de todos los subconjuntos de Q' que contengan estados de F':

  $F = \{F'\}$

Para comprobar el algoritmo anterior, se parte del AFND del apartado anterior $A' = (\Sigma', Q', f', s', F')$, en donde $A' = (\{a,b,c\}, \{q_0, q_1, q_2\}, f, q_0, \{q_0, q_1, q_2\})$ y con el objetivo de construir un AFD equivalente, $A = (\Sigma, Q, f, s, F)$, se establecen las correspondencias establecidas anteriormente:

- $\Sigma = \{a,b,c\}$
- $Q = 2^Q$, es decir, $Q = \{C_0, C_1, C_2, C_3\}$ en el que:

  $C_0 = \{q_0\}$
  $C_1 = \{q_0, q_1, q_2\}$
  $C_2 = \varnothing$
  $C_3 = \{q_2\}$

  y se obtiene de la siguiente forma:

  $f'(q_0, \lambda) = \{q_0\}$

  luego $C_0 = \{q_0\}$

  a partir de C_0

  $f'(q_0, a) = \{q_0, q_1\}$

  $f'(q_0, \lambda) = \{q_0\}$
  $f'(q_1, \lambda) = \{q_1, q_2\}$

  luego $C_1 = \{q_0, q_1, q_2\}$

  $f'(q_0, b) = \varnothing$

  luego $C_2 = \varnothing$

  $f'(q_0, c) = \varnothing$

  luego $C_2 = \varnothing$

  a partir de C_1

$$f'(q_0, a) = \{q_0, q_1\}$$
$$f'(q_1, a) = \varnothing$$
$$f'(q_2, a) = \varnothing$$
$$f'(q_0, \lambda) = \{q_0\}$$
$$f'(q_1, \lambda) = \{q_1, q_2\}$$

luego $C_1 = \{q_0, q_1, q_2\}$

$$f'(q_0, b) = \varnothing$$
$$f'(q_1, b) = \{q_0, q_1\}$$
$$f'(q_2, b) = \varnothing$$
$$f'(q_0, \lambda) = \{q_0\}$$
$$f'(q_1, \lambda) = \{q_1, q_2\}$$

luego $C_1 = \{q_0, q_1, q_2\}$

$$f'(q_0, c) = \varnothing$$
$$f'(q_1, c) = q_2$$
$$f'(q_2, c) = \varnothing$$
$$f'(q_2, \lambda) = \varnothing$$

luego $C_3 = \{q_2\}$

a partir de C_3

$$f'(q_2, a) = \varnothing$$
$$f'(q_2, b) = \varnothing$$
$$f'(q_2, c) = \varnothing$$

luego $C_2 = \varnothing$

- El estado inicial del AFD se corresponde con el subconjunto que contiene el estado inicial del AFND:

$$s = \{C_0\}.$$

- El estado, o los estados finales, del AFD se corresponde con el subconjunto que contiene el estado, o los estados finales, del AFND:

$$F = \{C_0, C_1, C_3\}.$$

- La función de transición f del AFD es:

$$
\begin{aligned}
&f(C_0, a) = C_1 \\
&f(C_0, b) = C_2 \\
&f(C_0, c) = C_2 \\
&f(C_1, a) = C_1 \\
&f(C_1, b) = C_1 \\
&f(C_1, c) = C_3 \\
&f(C_2, a) = \varnothing \\
&f(C_2, b) = \varnothing \\
&f(C_2, c) = \varnothing \\
&f(C_3, a) = \varnothing \\
&f(C_3, b) = \varnothing \\
&f(C_3, c) = \varnothing
\end{aligned}
$$

El AFD $A = (\{a,b,c\}, \{C_0, C_1, C_2, C_3\}, f, \{C_0, C_1, C_3\})$ sería el equivalente al AFND $A' = (\{a,b,c\}, \{q_0,q_1,q_2\}, f, q_0, \{q_0,q_1,q_2\})$ y se puede representar en una tabla de transiciones (Fig. 3.13). Posteriormente, se reduce a su AFD mínimo y se renombran los estados dando como resultado la tabla de transiciones (Fig. 3.14) y el diagrama de transiciones (Fig. 3.15) que se muestra a continuación.

f	a	b	c
→ * C_0	C_1	C_2	C_2
* C_1	C_1	C_1	C_3
C_2	∅	∅	∅
* C_3	∅	∅	∅

Fig. 3.13: Tabla de transiciones del AFD A equivalente a la tabla del AFND A'

f	a	b	c
→ * q_0	q_1	∅	∅
* q_1	q_1	q_1	q_2
* q_2	∅	∅	∅
∅	∅	∅	∅

Fig. 3.14: Tabla de transiciones del AFD mínimo A equivalente a la tabla del AFND A'

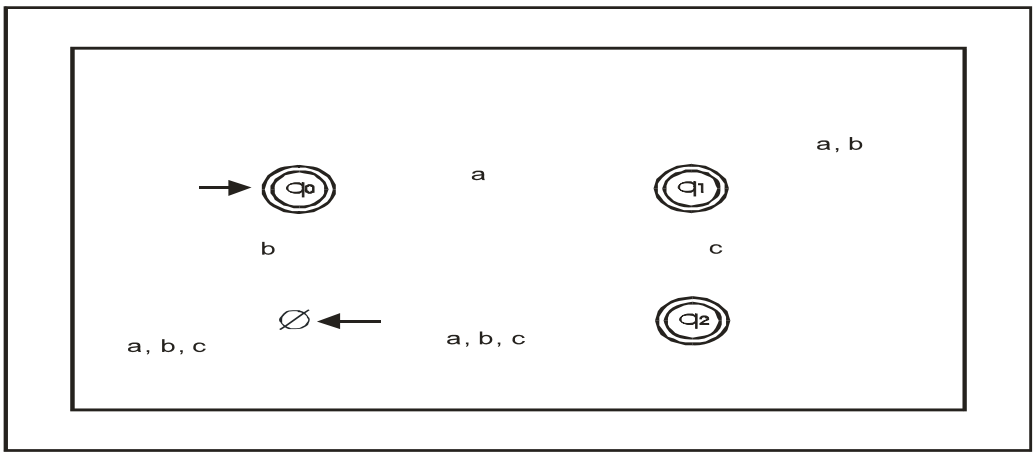


Fig. 3.15: Diagrama de transiciones del AFD mínimo A equivalente al diagrama del AFND A'

La idea básica es la siguiente: cada estado del AFD se debe corresponder con el conjunto de estados del AFND, y los estados de aceptación del determinista se deben corresponden con el subconjunto de estados del no determinista que contengan estados de aceptación. De tal forma que cada estado en el diagrama de transición del AFD se equipara con el conjunto de estados del AFND y lo mismo ocurre con los estados de aceptación. Para realizar esto, la función de transición f del AFD se lleva a cabo a partir del conjunto de todos los subconjuntos formados con estados del AFND.

Según lo anterior, sería necesario salvar dos diferencias para comprobar la mencionada equivalencia: no se puede realizar más de una transición entre estados para el mismo símbolo y no se pueden producir transiciones cuando no se reciba ningún símbolo. Sin embargo, ambos tienen que llegar a un estado de aceptación en el análisis de palabras de un mismo lenguaje, para que se pueda confirmar que son equivalentes. Como ambos autómatas aceptan

el mismo lenguaje se puede decir que reconocen las mismas palabras, o cadenas, y que por tanto son equivalentes, debido a que $L(A') = L(A)$.

3.2.4. Autómatas Finitos Probabilísticos: Modelos de Markov

Un Autómata Finito Probabilístico (AFP) es equivalente a un *Modelo de Markov,* o *proceso estocástico*. En una sucesión estocástica cada símbolo se encuentra en una relación de probabilidad con respecto a los símbolos próximos. Al aceptar, o generar una cadena, las etapas de su producción pueden considerarse como distintos estados de un sistema, en el que subyacen distintos cambios de estados sucesivos condicionados por la probabilidad de un estado y la probabilidad de transición de un estado al siguiente.

Los autómatas que permiten determinar cómo se produce la sucesión probable de símbolos, o cadenas, y fijar estadísticamente con qué probabilidad aparecen esos símbolos, en un instante concreto, se denominan *Autómatas Finitos Probabilísticos*. Un AFP se define como una quíntupla:

$$\text{AFP} = (\Sigma, \text{Q}, \text{M}, \text{P}(0), F)$$

donde

- Σ es el alfabeto o vocabulario de símbolos finito de entrada
- Q es el conjunto finito de estados
- M es el conjunto de las matrices de probabilidad de transición
- $P(0)$ es el vector probabilístico de estado inicial
- F es el conjunto de estados finales y un subconjunto de $Q(F \subseteq Q)$

Según esta definición, se introducen dos elementos nuevos con respecto a los AFD y AFND, que sustituyen a la función de transición y al estado inicial. Estos componentes nuevos son: las *matrices de probabilidad de transición* y los *vectores de estado*.

El conjunto de matrices de probabilidad de transición, M, está formado por tantas matrices como símbolos del conjunto Σ, de tal forma que si $|Q| = n+1$ entonces para cada símbolo del conjunto Σ hay una matriz cuadrada $(n+1)\times(n+1)$ donde n es el número de estado. De esta forma, para cada símbolo, n, del vocabulario de entrada Σ, $n \in \Sigma$, hay una matriz de probabilidad de transición:

$$M(n) = \begin{pmatrix} p_{00} & p_{01} & \dots & p_{0n} \\ p_{10} & p_{11} & \dots & p_{1n} \\ \dots & \dots & \dots & \dots \\ p_{n0} & p_{n1} & \dots & p_{nn} \end{pmatrix}$$

Esta matriz define la probabilidad de transición del estado, en el que se encuentra el autómata en un instante, al siguiente, después de recibir un símbolo de entrada. Se parte de un número finito de estados posibles, $q_0, q_1, \dots, q_n$, y de un conjunto de *probabilidades de transición*, $p_i(j)$, existiendo una *determinada probabilidad* de que q_i pase al siguiente estado q_j.

Por lo tanto, en un proceso estocástico la matriz de probabilidad de transición, p_{ij}, es la probabilidad de que estando en el estado q_i y recibiendo un símbolo, n, como entrada transite al estado p_j. Además, todos los elementos de la matriz están dentro del intervalo cerrado $[0,1]$, es decir, para cada p_{ij} se cumple $0 \le p_{ij} \le 1$; y para cada estado q_i la suma de cada fila es 1, es decir, para cada estado q_i de la fila se suma el valor de las probabilidades

de que pase a los estados q_j tantas veces como números de estados haya, esto se expresa formalmente como (Booth 1967):

$$\sum_{j=1}^{n} p_{ij} = 1$$

Si por ejemplo tenemos el siguiente Autómata Probabilístico definido como:

$$AFP' = (\{a, b, c\}, \{q_0, q_1, q_2\}, M, (0 \quad 0.8 \quad 0.2), \{q_1, q_2\})$$

donde $M = \{M(a), M(b), M(c)\}$ constituye el conjunto de matrices en la cual se establece las probabilidades de que cada símbolo siga a otro en una secuencia:

$$M(a) = \begin{pmatrix} 0 & 0.8 & 0.2 \\ 0 & 0.1 & 0.9 \\ 0.3 & 0.5 & 0.2 \end{pmatrix}$$

$$M(b) = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 0.5 & 0 \\ 0.2 & 0.6 & 0.2 \end{pmatrix}$$

$$M(c) = \begin{pmatrix} 0.5 & 0.2 & 0.3 \\ 0.6 & 0.1 & 0.3 \\ 0.7 & 0 & 0.3 \end{pmatrix}$$

Las matrices de probabilidad se pueden representar en tablas de transiciones (Fig. 3.16) y en un diagrama de transición (Fig. 3.17):

P_j

	M(a)	q_0	q_1	q_2
P_i	q_0	0	0.8	0.2
	q_1	0	0.1	0.9
	q_2	0.3	0.5	0.2

P_j

	M(b)	q_0	q_1	q_2
P_i	q_0	1	0	0
	q_1	0.5	0.5	0
	q_2	0.2	0.6	0.2

P_j

	M(c)	q_0	q_1	q_2
P_i	q_0	0.5	0.2	0.3
	q_1	0.6	0.1	0.3
	q_2	0.7	0	0.3

Fig. 3.16: Tabla de transiciones del AFP'

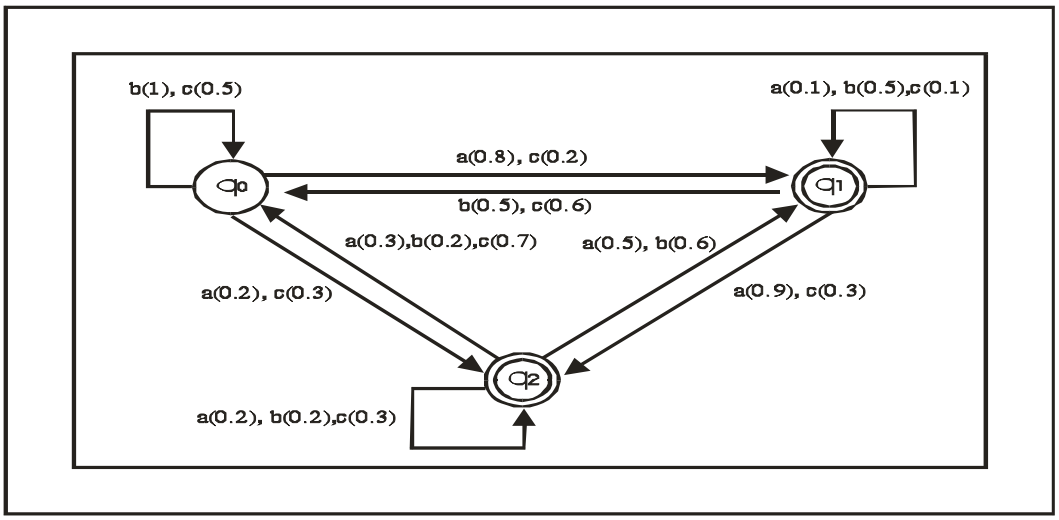


Fig. 3.17: Diagrama de transiciones del AFP'

La lectura que se le da a las matrices es la siguiente: a partir de un símbolo de entrada, no se puede precisar de forma absoluta el estado, o los estados, a los que llega el autómata. Sin embargo, sí es posible determinar estadísticamente la probabilidad de que se encuentre en cada uno de los estados del autómata, en un instante concreto, t, tras recibir un símbolo de entrada.

Se denomina *vector de estado*, $P(t)$, al vector $P(t)=(P_0(t), P_1(t),...P_n(t))$, donde $P_i(t)$ es la probabilidad de que el autómata se encuentre en el estado q_i en el instante t. Cumpliéndose que para cada instante t el valor del vector de estado es la suma de todas las probabilidades de que transite desde el estado, q_i, a tantos estados, n, como sea posible:

$$\sum_{i=1}^{n} P_i(t) = 1$$

El procedimiento para calcular la probabilidad de estar en el estado q_i en el instante t, $P_i(t)$, cuando se recibe determinado símbolo de entrada, a, tiene en cuenta dos componentes:

- El vector de estados aporta las probabilidades de que en el instante t se encuentre en cada uno de los estados j, $P_j(t)$
- La matriz de probabilidades de transición $M(a)$

Dado que ahora no se puede hablar de estado siguiente sino de probabilidad de que se encuentre en un estado. La probabilidad de estar en un estado en un instante, $P_i(t+1)$, se calcula como la probabilidad de estar en el estado 0 en el instante t multiplicado por la probabilidad de pasar desde el estado 0 al i tras recibir una a en la entrada, más la probabilidad de estar en el estado 1 en el instante t multiplicado por la probabilidad de pasar desde el estado 1 al i tras recibir una a en la entrada y así para todos los estados del autómata. Esto se expresa en la siguiente formula:

$$\mathrm{P_i(t+1)} = \sum_{\mathrm{j=1}}^{\mathrm{n}} \mathrm{P_j(t)}\ \mathrm{M_{ij}(a)}$$

El procedimiento anterior permite que $P_i(t+1)$ defina la accesibilidad del estado q_i desde el estado inicial $P(0)$ para el símbolo a, es decir, define el estado más probable en el que se

encuentra el autómata en un instante cuando recibe determinado símbolo en la entrada. La formula general para el vector completo es:

$$P(t+1) = P(t) \times M(a)$$

Si quisiéramos hallar las probabilidades que componen el vector de estados en el momento $t = 1$ cuando el AFP del ejemplo recibe una a, partiendo del vector de estados inicial $P(0) = (0 \quad 0.8 \quad 0.2)$, el procedimiento sería el siguiente:

$$P_1(1) = \sum_{j=1}^{3} P_j(0) M_{j1}(a) = 0 \times 0 + 0.8 \times 0 + 0.2 \times 0.3 = 0.06$$

$$P_2(1) = \sum_{j=1}^{3} P_j(0) M_{j2}(a) = 0 \times 0.8 + 0.8 \times 0.1 + 0.2 \times 0.5 = 0.18$$

$$P_3(1) = \sum_{j=1}^{3} P_j(0) M_{j3}(a) = 0 \times 0.2 + 0.8 \times 0.9 + 0.2 \times 0.2 = 0.76$$

Así, $P(1) = (0.06 \quad 0.18 \quad 0.76)$

Por tanto, en el instante 1 y tras recibir una a en la entrada, el autómata será más probable que se encuentre en el estado final q_2, por ser el que tiene la probabilidad más alta (0.76) –aunque también es probable que se encuentre en el otro estado final q_1, porque hasta ahora no se ha definido ningún estado final con una probabilidad mayor o igual a un valor preestablecido–.

Si quisiéramos hallar las probabilidades que componen el vector de estados en el momento $t = 2$ cuando el autómata recibe una a, partiendo del vector de estados $P(1) = (0.06 \quad 0.18 \quad 0.76)$ el procedimiento sería el mismo:

$$P_1(2) = \sum_{j=1}^{3} P_j(1) M_{j1}(a) = 0.06 \times 0 + 0.18 \times 0 + 0.76 \times 0.3 = 0.22$$

$$P_2(2) = \sum_{j=1}^{3} P_j(1) M_{j2}(a) = 0.06 \times 0.8 + 0.18 \times 0.1 + 0.76 \times 0.5 = 0.44$$

$$P_3(2) = \sum_{j=1}^{3} P_j(1) M_{j3}(a) = 0.06 \times 0.2 + 0.18 \times 0.9 + 0.76 \times 0.2 = 0.32$$

Así, $P(2) = (0.22 \quad 0.44 \quad 0.32)$, por tanto lo más probable es que se encuentre en el estado final q_1.

De esta forma, el vector de estados completo en el instante 1 se calcula a partir del vector inicial y de la matriz de probabilidad de transición como:

$$P(1) = P(0) \times M(a)$$

si fuera en el instante 2, al recibir una b, el vector de estados completo se calcularía de forma semejante:

$$P(2) = P(1) \times M(b)$$

o, lo que es lo mismo, si se hace la sustitución de $P(1)$:

$$P(2) = P(0) \times M(a) \times M(b)$$

De la misma forma si se pretendiera hallar el vector de estados completo cuando el autómata recibe una palabra o cadena X el procedimiento general sería el siguiente:

$$P(t) = P(0) \times M(a) \times M(b) \times ... \times M(n)$$

Si tuviéramos el caso concreto de una cadena $X = \text{def}$, se podría deducir sustituyendo $P(t)$ por $\mathrm{P_i}(X)$, y de esta forma $\mathrm{P_i}X$ definiría la accesibilidad del estado q_i desde el estado inicial para la palabra X, es decir, la probabilidad de que el autómata se encuentre en el estado final q_i en el instante t, después de leer la cadena «*def*»:

$$\mathrm{P_i}(X) = \mathrm{P}(0) \times \mathrm{M}(\mathrm{d}) \times \mathrm{M}(\mathrm{e}) \times \mathrm{M}(\mathrm{f})$$

Por otra parte, si se quisiera conocer el lenguaje reconocido por un *AFP* se debería ampliar la definición del autómata a una tupla de seis elementos:

$$\mathrm{AFP} = (\Sigma, \mathrm{Q}, \mathrm{M}, \mathrm{P}(0), F, u)$$

donde

- u determina un valor entre 0 y 1 denominado *umbral.*

A partir de esta nueva definición se puede decir que una palabra X es aceptada por un *AFP* si $\mathrm{P_i}(X) \geq u$, y por extensión el lenguaje aceptado por el autómata lo formarán todas las palabras que transiten a algún estado final con una probabilidad mayor o menor a un umbral prefijado, y si se modifica el umbral variará también el lenguaje aceptado.

Si se añade el umbral 0.6 al *AFP′* anterior, éste se redefiniría como:

$$AFP' = (\{a, b, c\}, \{q_0, q_1, q_2\}, M, (0 \quad 0.8 \quad 0.2), \{q_1, q_2\}, 0.6)$$

Para saber si una palabra X es aceptada por el autómata se tendría que multiplicar el vector de estado final de esa palabra por una matriz unidimensional o vector de estados, con ceros en

los estados no finales y con unos en los estados finales. Para calcular si una palabra, «*ac*», es aceptada por el autómata anterior se tendría que realizar el procedimiento siguiente:

$$P(ac) = P(0) \times M(a) \times M(c) \times \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

$$P(ac) = \begin{pmatrix} 0 \\ 0.8 \\ 0.2 \end{pmatrix} \times \begin{pmatrix} 0 & 0.8 & 0.2 \\ 0 & 0.1 & 0.9 \\ 0.3 & 0.5 & 0.2 \end{pmatrix} \times \begin{pmatrix} 0.5 & 0.2 & 0.3 \\ 0.6 & 0.1 & 0.3 \\ 0.7 & 0 & 0.3 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

$$P(ac) = \begin{pmatrix} 0.06 \\ 0.18 \\ 0.76 \end{pmatrix} \times \begin{pmatrix} 0.5 & 0.2 & 0.3 \\ 0.6 & 0.1 & 0.3 \\ 0.7 & 0 & 0.3 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

$$P(ac) = \begin{pmatrix} 0.67 \\ 0.03 \\ 0.3 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

$$P(ac) = \begin{pmatrix} 0.03 & 0.3 \end{pmatrix}$$

De lo anterior se deduce que el vector de estados finales de la palabra «*ac*» es menor que 0.6, por tanto dicha palabra no será aceptada por el AFP.

3.3. Transductores de Estado-Finito

Los AF se incluyen en el grupo de aceptadores, o reconocedores de lenguajes, sin embargo hay otro tipo de autómatas que tienen capacidad de *salida*. La superación de las limitaciones de los AF en los que la salida se limita a la señal *aceptado/no aceptado* tuvo su origen en dos modelos que tienen la capacidad de dada una cadena de entrada generar otra cadena de salida. Estos modelos están representados por la *Máquina Secuencial de Mealy* (Mealy 1955) y la *Máquina Secuencial de Moore* (Moore 1956), en los que se presentan dos procedimiento: la *salida* se asocia con la transición, como en la denominada *Máquina de Mealy*, o la salida se

asocia con el estado, como ocurre en la *Máquina de Moore*. Sin embargo, a diferencia de los AF, estos autómatas no se ocupan de *aceptar* una entrada, sino de transformar cadenas de entrada en cadenas de salida, esto explica que no haya conjuntos de estados finales, en este sentido se limitan a computar una función de Σ^* en Δ^* (Kelley 1995). Frente a ellos, los transductores se consideran autómatas de estado-finito cuyas transiciones se etiquetan con pares de símbolos y tienen la capacidad de, dada una palabra de entrada, generar otra palabra de salida.

Un Transductor de Estado-Finito, *Finite-State Transducer*, al igual que un AF acepta Lenguajes Regulares pero, a diferencia de éste, transforma una cadena aceptada en otra cadena, representando de esta forma una Relación Regular entre dos Lenguajes Regulares. En un FST, el primer símbolo es el *input* y el segundo es el *output*, la aplicación de este mecanismo a un *input* consiste en seguir un trayecto, *path*, de acuerdo a los símbolos de entrada mientras se almacenan los símbolos de salida (Roche y Schabes 1995). Con este objetivo, se definen un conjunto de estados que almacenan la parte de la palabra de entrada leída en cada momento y, a la vez que transitan entre los estados, generan un *ouput*. El resultado final lo constituye la secuencia almacenada de símbolos de salida, que se equipara a la cadena, o símbolos de entrada.

Tanto los Lenguajes como las Relaciones Regulares están representados por Expresiones Regulares que se codifican como redes de estado-finito, los lenguajes estarían codificados por autómatas y las relaciones por transductores. Las Expresiones Regulares *se compilan en una red que puede representar tanto el correspondiente lenguaje, o la correspondiente relación* (Karttunen et al. 1996). De esta forma, las *Expresiones Regulares* denotan *conjuntos*, semejantes a la *lógica de Boole*, pudiéndose distinguir dos clases de conjuntos: *a) conjuntos de cadenas*; y *b) pares de cadenas*. El término lenguaje se refiere a un conjunto de cadenas simples y el término relación se refiere a los conjuntos de pares de cadenas. A su vez, los *conjuntos que se pueden describir por medio de un Lenguaje Regular y de una Relación Regular constituyen una Expresión Regular* (Karttunen et al. 1996).

Un FST se encarga de aceptar relaciones entre dos Lenguajes Regulares, que habitualmente se les denomina *lenguaje superior* (*upper language*) y *lenguaje inferior* (*lower language*) (Karttunen 1995). La clase más simple de Expresión Regular que denota una relación la forma el par de símbolos $a:b$. Dicha relación consiste en la correspondencia de las expresiones de dos Lenguajes Regulares: el lenguaje superior estaría representado por la expresión a, denotando el lenguaje consistente en la cadena a, y el lenguaje inferior estaría representado por la expresión b, denotando el lenguaje consistente en la cadena b.

El transductor que acepta Expresiones Regulares de este tipo equipara una cadena *input* del primer lenguaje, o lenguaje superior (L_1), con la correspondiente cadena *input* del segundo, o lenguaje inferior (L_2), y viceversa. De alguna forma, se considera que los AFD procesan una cadena simple en el *input* y los FST procesan simultánemente dos cadenas en el *input*, esto es, reconocen si las dos cadenas constituyen una correspondencia válida. Habitualmente, se denomina a estas dos cadenas como cadena *input* y cadena *output*. Además, los transductores se pueden emplear también como generadores de cadenas y en este caso se encargan de recibir una cadena *input* y generar una cadena *output*.

Siguiendo con esto, como los FST codifican relaciones entre dos Lenguajes Regulares, los arcos de la red están etiquetados con pares de símbolos según la relación denotada por la Expresión Regular. En el diagrama de transiciones (Fig. 3.18) y en la tabla de transiciones (Fig. 3.19) de un Transductor Finito se muestra cómo se transforman una cadena en otra, así la cadena $X = \text{a b c}$ perteneciente al L_1 es aceptada por dicho transductor y se transforma en la cadena $X = \text{a b d}$ perteneciente al L_2.

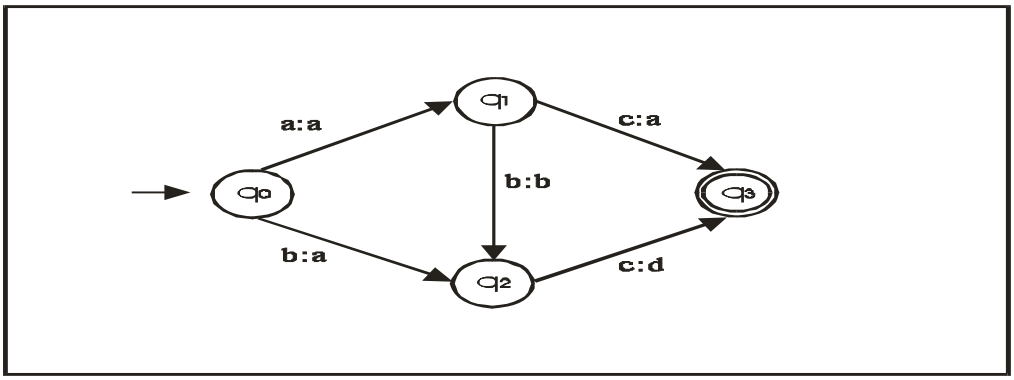


Fig. 3.18: Diagrama de transiciones de un FST

f/δ	a:a	b:a	c:a	b:b	c:d
→ q_0	q_1	q_2			
q_1			q_3	q_2	
q_2					q_3
$*q_3$					

Fig. 3.19: Tabla de transiciones de un FST

Por otra parte, los transductores encargados de representar Relaciones Regulares entre Lenguajes Regulares se clasifican en:

a. Transductores no-secuenciales: con *input* no determinista.
b. Transductores secuenciales: con *input* determinista:
   - Transductores subsecuenciales: con *input* determinista y *output* adicional.
   - Transductores *p*-subsecuenciales: con *input* determinista y con un número finito de cadenas de salida en los estados finales.

La aplicación de un transductor es parecida a la de un autómata, sin embargo, mientras que cualquier AFND se puede transformar en su equivalente AFD, un *Transductor No-Secuencial* no tiene su equivalente en un *Transductor Secuencial*. Esto genera ambigüedades en las correspondencias entre cadenas si estos transductores se aplican al PLN, en un intento por

solucionar este problema se desarrollarán distintos procesos que se describirán en el capítulo siguiente. Por ahora, nos vamos a deterner en ofrecer una definición más en profundidad de este tipo de mecanismos.

3.3.1. Transductores Finitos No-Secuenciales

A partir de la clasificación del apartado anterior, la descripción formal de un *Transductor Finito No-Secuencial* (TNS) se define como un conjunto de siete elementos (Mohri 1995):

$$TNS = (\Sigma, \Delta, Q, f, \delta, I, F)$$

donde

- Σ es un conjunto finito de símbolos, constituye el alfabeto de *input* del transductor.
- Δ es un conjunto finito de símbolos, constituye el alfabeto de *output* del transductor.
- Q es el conjunto de estados.
- f es la función de transición entre los estados y equipara $Q \times \Sigma$ a 2^Q, donde 2^Q es la potencia de Q, es decir, el conjunto de todos los subconjuntos de Q y se define como

  $$f: \quad Q \times \Sigma \to 2^Q$$

- δ es la función de salida y equipara $Q \times \Sigma \times Q$ a 2^{Δ^*}, se define como:

  $$\delta : Q \times \Sigma \times Q \to 2^{\Delta^*}$$

- $I \subseteq Q$ es el conjunto de estados iniciales.
- $F \subseteq Q$ es el conjunto de estados finales.

En estos transductores a partir de un mismo estado se pueden producir dos o más transiciones con el mismo elemento de entrada según la función: $Q \times \Sigma \rightarrow 2^Q$, a su vez de cada uno de estos estados se pueden realizar distintas salidas según la función: $Q \times \Sigma \times \quad Q \rightarrow 2^{\Delta^*}$. En este caso, el lenguaje del *input* es ambiguo debido a que una misma cadena en el *input* puede desencadenar más de una transición y el transductor se considera en este caso no-determinista o no-secuencial. Además, esto conlleva que sea necesariamente no-determinista en la salida. La propiedad de no-determinismo en el *input* hace que estos transductores no sean adecuados en las tareas relacionadas con el PLN, donde se utilizan habitualmente los transductores deterministas, o secuenciales, como los que se van a describir a continuación.

3.3.2. Transductores Finitos Secuenciales

Como sucedía con los AF un arco puede estar etiquetado con el mismo símbolo y producir más de una transición en el caso de no-determinismo, por esta razón es necesario tener en cuenta las siguientes consideraciones (Mohri 1995):

- Un *transductor* se dice que es *secuencial* cuando tiene una *entrada determinista*, es decir, cualquier estado tiene como máximo una sola transición etiquetada con un elemento del alfabeto de entrada.
- Los transductores secuenciales pueden ampliarse para permitir una *cadena de salida adicional*, o un número *finito p de cadenas de salida en los estados finales*, denominándose *transductores subsecuenciales* y *transductores p-subsecuenciales* respectivamente.
- Se usa el término *secuencial* de una forma genérica para designar la clase de todos los transductores con *input determinista,* tales como los transductores *secuenciales, subsecuenciales*, o *p-secuenciales*.
- Se llama *determinación* al *algoritmo que permite obtener un transductor p-subsecuencial* a partir de cualquier transductor secuencial.

Sin embargo, un *Transductor Finito Secuencial* sería semejante a un *Autómata Finito No-Determinista* (AFND), porque aunque tenga *input* determinista, se pueden producir dos transiciones partiendo del mismo estado, o dicho de otro modo, con la misma entrada se pueden producir distintas salidas. En este caso, al igual que un AFND se podía transformar en un AFD por un proceso de determinación, un Transductor Secuencial (TS) se puede transforma para que se pueda distinguir la misma salida distintas entradas. El proceso de determinación es bastante complejo y se puede lograr añadiendo cadenas adicionales en los estados finales que se representan por medio de *Transductores p-subsecuenciales*. Esta propiedad hace que se puedan evitar las ambigüedades en el *output,* y de esta forma se puedan aplicar en tareas relacionadas con PLN.

En un TS, o transductor con *input* determinista, cualquier estado al recibir un elemento del alfabeto realiza como máximo una sola transición y no hay, por tanto, dos arcos etiquetados con el mismo elemento, según el principio del determinismo. Sin embargo, la salida de un TS no tiene por qué ser determinista, es decir, pueden haber arcos etiquetados con los mismos símbolos de *output* (Fig. 3.20). Otra característica es que aunque el transductor reciba un solo símbolo en el *input*, puede tener una cadena como *output*, incluyendo la cadena vacía, aunque no permita la cadena vacía en el *input*. Todo esto se sintetizaría en que los TS son deterministas en el *input*, pero pueden ser no-determinista en el *output*.

Formalmente, un Transductor Secuencial de cadena-a-cadena, *Sequential string-to-string Transducer*, se define como una tupla de siete elementos (Mohri 1997):

$$TS = (\Sigma, \Delta, Q, f, \delta, i, F)$$

donde

- Σ y Δ son conjuntos finitos que corresponden respectivamente a los alfabetos de *input* y *output* del transductor.
- Q es el conjunto de estados.

- f es la función de transición de estado que equipara $Q \times \Sigma$ a Q. Se define como:

  $f: \quad Q \times \Sigma \rightarrow Q$

- δ es la función de output que equipara $Q \times \Sigma$ a Δ^*. Se define como:

  $\delta: \quad Q \times \Sigma \rightarrow \Delta^*$

- $i \in Q$ es el estado inicial
- $F \subseteq Q$ es el conjunto de estado finales

Las funciones f y δ se pueden ampliar equiparando $Q \times \Sigma^*$, de esta forma un estado $q \in Q$ admite transiciones de salida etiquetadas en el lado del *input* con la cadena vacía. Las ampliaciones de ambas funciones se establecen de la siguiente forma (Mohri 1997):

$$\forall q \in Q, \ \forall X \in \Sigma^*, \ \forall a \in \Sigma,$$
$$f(q,\lambda) = q, \ f(q, X\,a) = f(f(q,X),a);$$
$$\delta(q,\lambda) = \lambda, \ \delta(q, X\,a) = \delta(q,X)\ \delta(f(q,X), a)$$

Así, una cadena $X \in \Sigma^*$ es aceptada por TS si $f(i,X) \in F$, y en este caso la salida del transductor es $\delta(i,X)$.

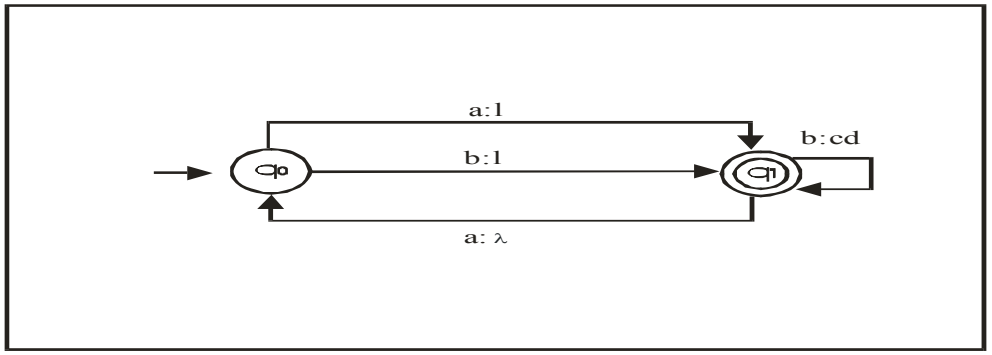


Fig. 3.20: Transductor Secuencial

Una extensión de los TS la constituye los Transductores Subsecuenciales, que al igual que los primeros no producen una salida hasta que no han aceptado una palabra, pero además pueden generar una cadena de *output* adicional en el estado final que se concatena a la salida producida por la función de *output* o producción, δ. Un Transductor Subsecuencial se define como una tupla de ocho elementos, siete de ellos idénticos a un Transductor Secuencial y uno adicional que constituye la función de emisión final (Mohri 1995):

$$T_1 = (\Sigma, \Delta, Q, f, \delta, \varphi, i, F)$$

donde

- φ es una función de *output* final que equipara F a Δ^* y se define como:

$$\varphi : F \rightarrow \Delta^*$$

La aplicación de un Transductor Subsecuencial a una cadena X introduce la posibilidad de generar un *output* según la función de producción y un *output* adicional en los estados finales que se concatena al primero:

- $f(X) = X'$ donde $X \in \Sigma^*$ y $X' \in \Delta^*$, es decir, f equipara cadenas del alfabeto de entrada con cadenas del alfabeto de salida definiéndose como $f : \Sigma^* \rightarrow \Delta^*$.
- Una cadena de salida adicional en los estados finales por medio de la función $\varphi : F \rightarrow \Delta^*$.

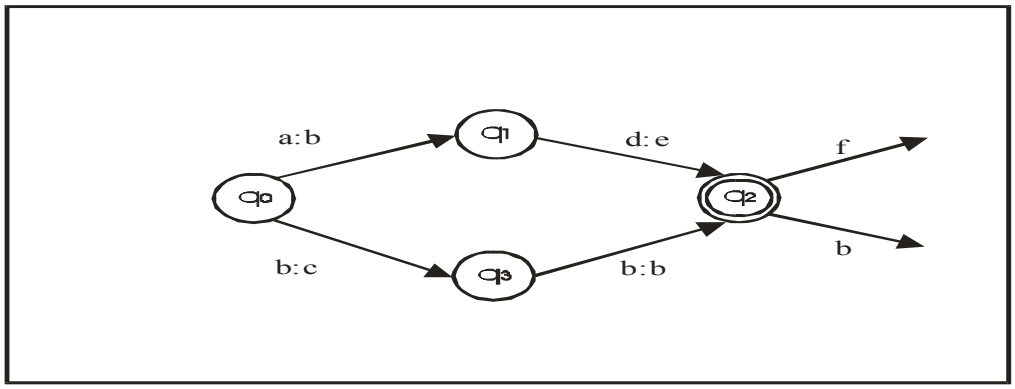


Fig. 3.21: Transductor Subsecuencial

Según lo anterior, si la cadena de *input* a un Transductor Subsecuencial (Fig. 3.21) es $X = \text{ad}$, la cadena de *output* sería $X' = \text{be}$, a la que se asocian dos salidas adicionales, $X' = \text{bef}$ y $X' = \text{beb}$. En el caso de que la cadena de entrada fuera $\chi = bb$, se obtendrían también dos salidas distintas $X = \text{cbf}$ y $X = \text{cbb}$. Los Transductores Subsecuenciales que producen un número p de cadenas de salida adicional se denominan Transductores `p-Subsecuenciales` (Mohri 1997). El ejemplo de la Figura 3.21 sería un Transductor `2-Subsecuencial` (los transductores con una sola salida adicional, `1-secuenciales`, serían exactamente Transductores Subsecuenciales).

Por otra parte, el lenguaje aceptado por cualquiera de los transductores anteriores estaría formado por el conjunto de cadenas que lo llevan de un estado inicial a un estado de aceptación. Si quisiéramos saber si una determinada cadena pertenece o no al lenguaje que acepta el transductor, el algoritmo partiría del estado inicial y recorrería todo el trayecto entre los estados hasta consumir todos los elementos de la cadena, si se alcanza un estado final se podría decir que la cadena pertenece al lenguaje que acepta dicho transductor.

En el proceso de transducción, sólo después haber aceptado una cadena el transductor es capaz de transformarla en otra, sin embargo una cadena que no pertenezca al lenguaje reconocido por el transductor puede contener subcadenas que sí formen parte del lenguaje aceptado por dicho transductor. Con el objetivo de reconocer dichas subcadenas, Roche y

Schabes (Roche y Schabes 1995) proponen *Extensiones Locales* de los transductores, o lo que es lo mismo, ampliaciones con *transiciones* $-?$ y *transiciones* $-\lambda$. La aplicación de tales ampliaciones es de gran utilidad en la traducción de cadenas como se verá a continuación.

Un transductor se puede ampliar en función de las cadenas o palabras que acepta, en este caso la función de transición estaría definida por $f(X)=X'$ y estaría representa por un transductor T. Las funciones sobre cadenas que se representan por transductores de estado-finito se denominan funciones racionales, *rational functions*, y se expresan de la siguiente forma: $f=|T|$, y si algún *input* tiene más de un *output*, $f(X)=\{X_1, X_2 ...\}$ entonces f se denomina *transducción racional, rational transduction* (Roche y Schabes 1995).

Si tuviéramos el siguiente Transductor T_2 (Fig. 3.22):

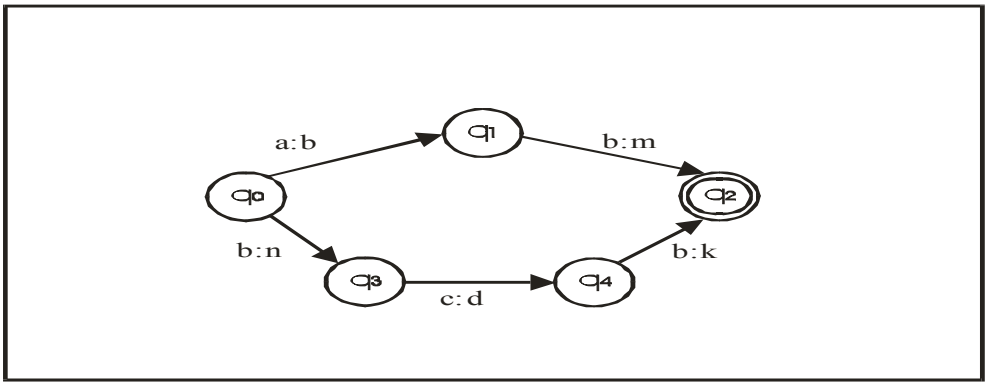


Fig. 3.22: Transductor Secuencial T_2

La función racional del transductor, f_2, se definiría como $f_2=|T_2|$ de tal forma que $f_2(ab)=bm\, f_2$ y $f_2(bcb)=ndk$, estando representadas por el transductor como $|T_2|(ab)=bm$ y $|T_2|(bcb)=ndk$. Por otra parte, el dominio de f, $dom(f)$, *denota el conjunto de palabras que tiene al menos un output de* f, si la cadena de entrada es X cada vez que dicha cadena

contenga factores que están en el $dom(f)$ esos factores se pueden transformar (Roche y Schabes 1995).

Supongamos que la cadena de entrada al transductor T_2 es $X_1 = \text{abcbab}$, los factores que están en el $dom(f_2)$ se pueden encontrar de acuerdo a dos factorizaciones:

- $X_1 = X_2 \text{cb}. X_2$

  donde $X_2 = \text{ab}$
- $X_1 = a . X_3 . X_2$

  donde $X_3 = \text{bcb}$ y

  donde $X_2 = \text{ab}$

La *Extensión Local* de f_2 sería la función que toma cada posible factorización y transforma cada factor de acuerdo a f_2, dejando las otras partes invariables. La definición formal es la siguiente (Roche y Schabes 1995): Si f es una transducción racional de $\Sigma^* \rightarrow \Delta^*$, la *Extensión local* $F = LocExt(f)$ es la transducción racional de $\Sigma^* \rightarrow \Delta^*$, definida de la siguiente forma:

si $X = a_1 b_1 a_2 b_2 ... a_n b_n \in \Sigma^*$

entonces $X' = a_1 B_1 a_2 B_2 . . . a_n B_n \in F(X)$

si $a_i \in \Sigma^* - (\Sigma^* \cdot dom(f) \cdot \Sigma^*), b_i \in dom(f)$ y $B_i \in f(b_i)$

Aplicando dicha Extensión a las subcadenas pertenecientes al lenguaje aceptado por el Transductor T_2, obtendríamos dos cadenas distintas como salida:

Input:

$X = \underline{a\ b}\ c\ b\ \underline{a\ b}$

Output:

$X' = b\ m\ c\ b\ b\ m$

primera factorización $f_2(X_2) = \text{bm}$

Input:

$X = a\ \underline{b\ c\ b}\ \underline{a\ b}$

Output:

$X' = a\ n\ d\ k\ b\ m$

segunda factorización $f_2(X_3) = \text{ndk}$ y $f_2(X_2) = \text{bm}$

De esta forma, aunque la cadena $\text{X} = \text{abcbab}$ no forme parte del lenguaje aceptado por el transductor T_2 (y, en consecuencia, no pueda ser transformada en otra cadena) sí contiene subcadenas que pertenecen al lenguaje aceptado por el transductor T_2. La transducción de esas subcadenas se puede realizar precisamente por medio de la denominada *Extensión Local* de la función de transición. Dicha extensión hará posible que las construcciones léxicas o sintácticas que no coincidan con las cadenas aceptadas por un transductor se puedan transformar en determinadas subcadenas dentro de una palabra o de una oración.

La ampliación anterior supone una gran ventaja en las aplicaciones prácticas porque muchas veces las cadenas o constituyentes sintácticos no se ajustan a las construcciones que acepta el transductor y sin embargo pueden contener subcadenas que sí lo hacen. En el caso que nos ocupa, cuando el T_2 analiza la cadena $X = abcbab$ y encuentra el símbolo c que no forma parte del lenguaje lo transduce por sí mismo c/c dejándolo sin transformación, lo mismo ocurre con los símbolos restantes. Así la cadena:

$X = \underline{a\ b}\ c\ b\ a\ b$

se transforma en

$X' = b\ m\ c\ b\ a\ b$

Pero en la cadena resultante existe una subcadena, $X' = b\ m\ c\ b\ \underline{a\ b}$, que sí forma parte del lenguaje reconocido por T_2, para ello el transductor necesitará volver al estado inicial y leer nuevamente esa subcadena para poder transformarla. Con el objetivo de que se pueda producir la transducción de un símbolo por sí mismo y con el fin de que el transductor vuelva al estado original se introducen las denominadas $transiciones-?$ y las $transiciones-\lambda$ (Roche y Schabes 1995) que harán posible la aplicación de la *Extensión Local* de la función de transición como se está comprobando. Las $transiciones-?$, etiquetadas $?/?$, van a permitir aceptar cualquier símbolo de la cadena de entrada que no sea reconocido por el transductor y transformarlo por sí mismo. Las $transiciones-\lambda$ vacías, etiquetadas λ/λ, situadas en los estados finales van a permitir que el transductor vuelva a situarse en su estado inicial. De esta forma, la $transición-\lambda$ hará posible que la cadena:

$X' = b\ m\ c\ b\ \underline{a\ b}$

se transforme en

$X' = b\ m\ c\ b\ b\ m$

Cuando el Transductor T_2 lee los símbolos del *input* se comporta de dos formas: o bien transformando un símbolo por otro, o bien transformando un símbolo por sí mismo en cuyo caso se habla de función de identidad. Para que se pueda realizar el proceso anterior se tendría que agregar a T_2 una parte que fuera capaz de realizar la identidad. Con este objetivo se diseña un algoritmo consistente en construir una copia del transductor original y al mismo tiempo añadir una parte adicional que realice la función de identificación, para ello según Roche y Schabes (Roche y Schabes 1995) se tiene en cuenta lo siguiente

- Si $F = LocExt(f)$ y $X \in \Sigma^*$, cada factor de X en $dom(f)$ *se transforma en su* imagen, si f se representa por un transductor T, esto es $f = |T|$, y la $LocExt(f)$ se representa otro transductor T', se escribe $T' = LocExt(T)$.
- Si I_T es la función de identidad sobre $\Sigma^* - (\Sigma^* \cdot dom(T) \cdot \Sigma^*)$, entonces $LocExt(T) = I_T \cdot (T \cdot I_T)^*$.

A partir de aquí se crearían un nuevo transductor $T_3 = LocExt(T_2)$ con dos clases de estados que harán que una cadena de *input* se pueda procesar de forma no determinista (Fig. 3.23) en virtud de los:

- *Estados de traducción*: cuando las transiciones sean del tipo a / b.
- *Estados de identidad*: cuando las transiciones sean del tipo a / a.

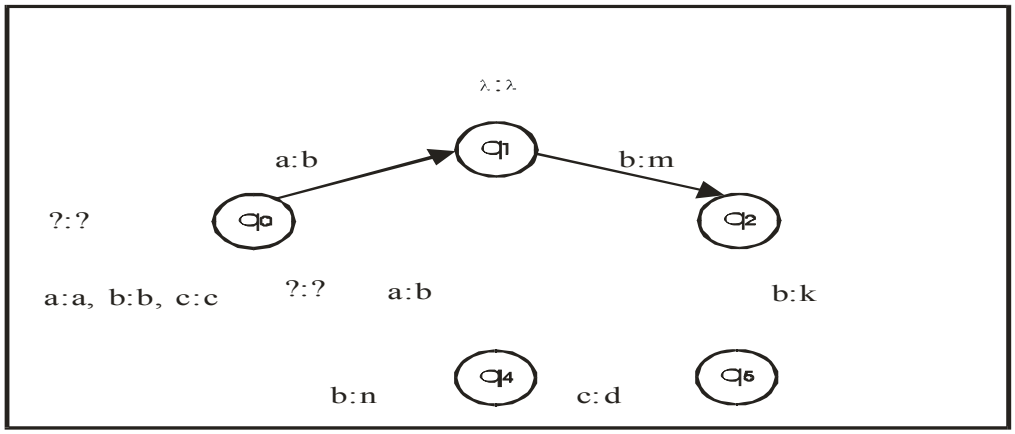


Fig. 3.23: Extensión Local T_3 de T_2

3.3.3. Transductores Finitos Probabilísticos: Modelo Oculto de Markov

Si los Lenguajes Regulares se definen por Autómatas Finitos, Gramáticas Regulares y Expresiones Regulares, los *Lenguajes Regulares Estocásticos* se definen de forma similar por *versiones estocásticas de estas tres representaciones* (Ross 2000). De la misma forma que un Autómata Finito Probabístico se define en términos de un Modelo de Markov, un Transductor Finito con probabilidades, que marcan las transiciones entre los estados, sería equivalente a un *Modelo Oculto de Markov*, *Hidden Markov Model* (HMM), dicho de otro modo, un Modelo de Markov equivaldría a un *Weighted FSA* y un Modelo Oculto de Markov a un *Weighted FST*.

Los HMM se usan también en muchas tareas relacionadas con el PLN como son el etiquetado de categorías léxico-gramaticales, *part-of-speech tagging*, (Kupiec 1992), el reconocimiento del habla (Rabiner 1989) o la identificación de información (van Mulbregt et al. 1998). Esta herramienta también se emplea en el campo de la extracción de información con el objetivo seleccionar entidades, en el caso del sistema Nymble (Bikel et al. 1997), o para extraer nombres y lugares de los *abstracts* científicos, como en el sistema propuesto por Leek (Leek 1997). Otros sistemas construyen los campos de una base de datos (McCallum et al. 1999) aplicando HMM con el objetivo de extraer de forma automática datos –como título, nombres de los autores, afiliación o dirección– del encabezamiento de los artículos científicos

Un HMM es un tipo de *Autómata de Estado Finito Probabilístico con transiciones de estado estocásticas y observaciones, o símbolos de emisión* (Rabiner 1989). El autómata utiliza un proceso generativo probabilístico a partir del cual se produce una secuencia de observaciones comenzando en el estado inicial, transitando a un nuevo estado, emitiendo una observación seleccionada por este estado, transitando de nuevo, emitiendo otra observación y así hasta alcanzar un estado final. Formalmente un *HMM* , o transductor probabilístico, se define como (McCallum, Freitag y Pereira 1999):

- *Un conjunto finito de estados* Q , con estado inicial, q_I , y estados finales, q_F .

- *Un conjunto de transiciones entre estados* $(q \to q')$
- *Un vocabulario de símbolos de output,observaciones* O, $\Sigma = \{o_1, o_2, \ldots o_m\}$
- Distintas *distribuciones de probabilidades* condicionales:
    1. Asociado a cada conjunto de estados, $Q = \{q_1, \ldots, q_n\}$, hay una matriz de probabilidad sobre los símbolos del vocabulario de emisión, $\Sigma = \{o_1, o_2, \ldots o_m\}$. Esto se denomina *Probabilidades de observación*, es decir, las probabilidades de que un estado emita un símbolo de salida particular, $P(q \uparrow o)$ para $q \in Q$ y $o \in \Sigma$.
    2. Asociado a cada estado hay una matriz sobre este conjunto de transiciones de salida, que se contiene los datos de las *probabilidades de transición de estados*, o las probabilidades de que un estado siga a otro q a q', $P(q \to q')$ para q y $q' \in Q$.
    3. Una distribución o vector de estado inicial $P_0(Q)$

El modelo genera una cadena $x = x_1 x_2 \ldots x_I$ comenzando en el estado inicial, transitando a un nuevo estado, emitiendo un símbolo de salida, transitando a otro estado, emitiendo otro símbolo y así hasta que una transición se produzca en el estado final. La probabilidad de que una cadena x sea emitida por un transductor probabilistico M se calcula como una suma de todas las posibles trayectorias:

$$P(x|M) = \sum_{q_1, \ldots q_l \in Q^l} \prod_{k=1}^{l+1} P(q_{k-1} \to q_k) P(q_k \uparrow x_k)$$

La salida de observaciones del sistema es la secuencia de símbolos que el estado emite, pero la secuencia de estados subyacente está *oculta*. Para recuperar la *secuencia de estados que tiene la probabilidad más alta de producir una secuencia de observaciones se aplica el algoritmo de Viterbi* (Viterbi 1967):

$$V(x|M) = \underset{q_1 \ldots q_l \in Q^l}{\mathbf{arg\ max}} \prod_{k=1}^{l+1} P(q_{k-1} \rightarrow q_k)\ \ P(q_k \uparrow x_k)$$

3.4. El proceso de análisis léxico y sintáctico con Técnicas de Estado-Finito

El proceso de reconocimiento de construcciones léxicas y sintácticas consiste en un conjunto de procesos que determinan si un *input* dado es aceptado o no por una máquina. Si la máquina fuera un AFD o un FST, el proceso de reconocimiento lo constituiría, en este caso, la operación por la cual se determina si un *input* dado se equipara o no con una Expresión Regular. En las máquinas de estado-finito cada *input*, o cadena, se analiza como una secuencia de símbolos y, según vaya llegando cada símbolo del *input* a la máquina, el proceso de reconocimiento desencadena bien un cambio de un estado a otro, o bien la permanencia en el estado actual.

La operación anterior se representa como una máquina que lee de izquierda a derecha los símbolo de un cinta fraccionada en celdas que contienen los elementos del alfabeto (Fig. 3.24). En un AFD cada vez que se lee un símbolo la máquina se mueve a la celda siguiente según la función de transición representada en un diagrama, o en una tabla de transiciones. En un transductor se añade una cinta adicional que va a permitir que la máquina pueda leer de una cinta el segundo elemento de cada transición y escribir en la cinta adicional el primer elemento de la transición –al igual que el autómata, los símbolos de cada una de las cintas del transductor se corresponden a la función de transición representada por el par de elementos con los que están etiquetados los arcos del diagrama de transiciones, o con la información que aporta la tabla de transiciones–.

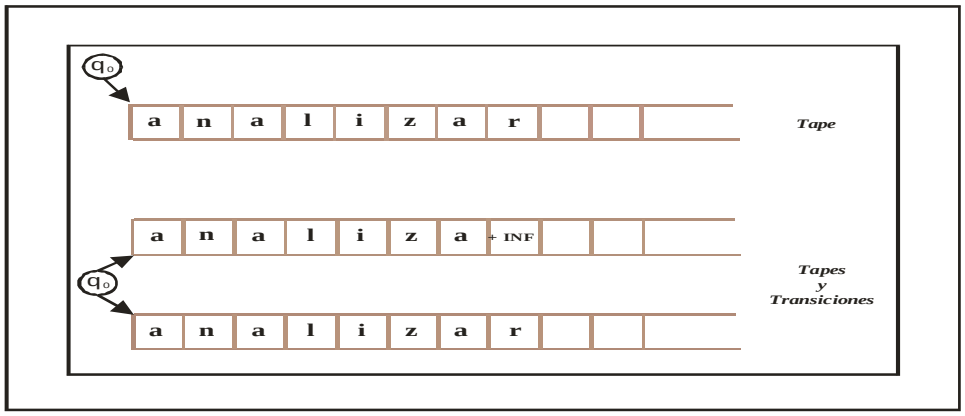


Fig. 3.24: Representación gráfica del proceso de reconocimiento de cadenas realizado por FSA y FST

A grandes rasgos, el proceso anterior es el que se utiliza en los analizadores léxicos y sintácticos de los compiladores de lenguajes de programación. Sin embargo, y a pesar de que estos mecanismos no son muy complejos, hasta hace muy poco estas técnicas no se han explotado de forma generalizada en tareas relativas al PLN (Kaplan 1995). Con el objetivo de describir el funcionamiento de los analizadores léxicos y sintácticos del lenguaje natural nos interesa plantear un paralelismo entre los módulos de los sistemas que operan con lenguaje formal y los módulos que operan en los sistemas de PLN.

Un compilador es un programa escrito en algún lenguaje de programación cuya finalidad es traducir el correspondiente programa fuente, constituido por un conjunto de instrucciones en un lenguaje de alto nivel, a su equivalente en código máquina, o programa objeto. Los componentes básicos, entre otros, del proceso de traducción del compilador son: *analizador léxico*, *analizador sintáctico*, y *generador de código*, cada uno origina una salida correspondiente. El éxito del proceso de traducción del analizador léxico y sintáctico está subordinado a la *capacidad para reconocer patrones* y, si éstos se basan en *reglas gramaticales*, el proceso de reconocimiento también va a depender de esas reglas (Brookshear 1995). El analizador léxico del compilador es un módulo que tiene como entrada cadenas de símbolos las cuales segmenta en unidades significativas como nombres, variables, constantes, operadores o palabras reservadas del propio programa fuente. Estas unidades se denominan unidades léxicas, componentes léxicos o *tokens* y salen codificadas de alguna forma para ser procesadas en el subsiguiente módulo del compilador.

Por su parte, un programa de PLN tiene básicamente como objetivo analizar el texto fuente en lenguaje natural y transferirlo a un texto objeto o representación destino según la tarea para la que haya sido diseñado. Los módulos, o grupos de módulos, de estos sistema (Fig. 3.25) varían dependiendo de las distintas aplicaciones. Los límites entre componentes son muchas veces borrosos y no es posible realizar un buen análisis léxico o morfológico sin el conocimiento que aporta el componente sintáctico, o un buen análisis sintáctico sin el conocimiento que aporta el módulos semántico. El analizador léxico se integra también como el primer proceso de los sistemas de PLN que tiene como entrada un texto en lenguaje natural el cual queda dividido en una secuencia ordenada de *tokens*, cada uno de los cuales representa un ítem no ambiguo. En este primer componente se suministra información sobre las características individuales de las palabras a través del *lexicon* y del analizador morfológico. Esta operación se realiza normalmente accediendo a un diccionario electrónico, definido como un conjunto finito de palabras, que proporciona los datos sobre aspectos de naturaleza morfológica –como categorías gramaticales, *part-of-speech* (POS), número, o género–

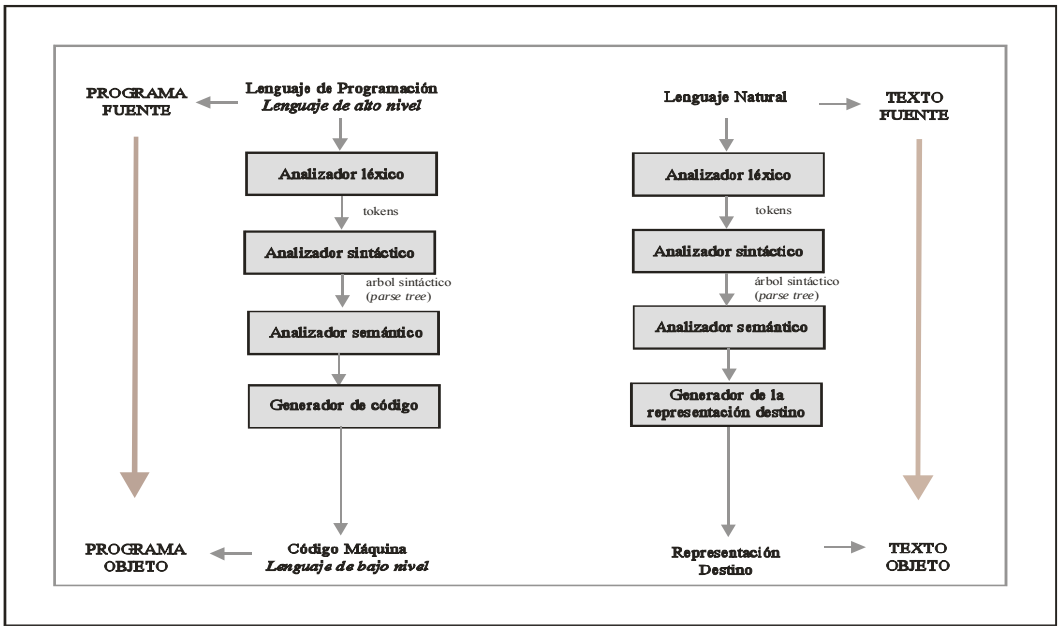


Fig. 3.25: Comparación entre el proceso de traducción del Lenguaje Formal y del Lenguaje Natural

El módulo de análisis léxico en PLN se puede desarrollar siguiendo distintos métodos, la aplicación que vamos a utilizar se basa en autómatas y transductores de estado-finito. Estos mecanismos son especialmente apropiados para representar diccionarios porque aceptan precisamente lenguajes finitos, que se corresponderían con la propia definición de un diccionario. Pero además, ampliando esta definición, se pueden construir diccionarios que contengan no sólo palabras sino formas canónicas y sus correspondientes categorías morfológicas constituyendo así diccionarios morfológicos que se ajustan adecuadamente a esta técnica.

Sin embargo, a pesar de la eficacia de este tipo de autómatas al proceso de análisis léxico, aún siendo particularmente adecuados en el módulo de analisis léxico, estos mecanismos no ofrecen una solución definitiva. Esto se debe a los distintos problemas que originan los lenguajes naturales como la propia segmentación de las palabras, el gran volumen de información léxica necesaria para procesar el lenguaje natural, así como los problemas intrínsecos del léxico como locuciones, homonimia, polisemia o cambios de significado. En definitiva, el gran problema de los lenguajes naturales que es el de la ambigüedad.

Aún teniendo en cuenta las consideraciones anteriores, estas diferencias no son tan significativas cuando las técnicas de PLN se aplican a los sistemas que pretenden reconocer *patrones lingüísticos* con un objetivo específico como la extracción, indización o clasificación por su vinculación al componente léxico y por consiguiente a los *patrones de dominio*. Por esta razón, nos centramos más en los aspectos en los que los lenguajes formales y naturales se aproximan que en los que los separan.

Con el presupuesto anterior, vamos a describir de forma muy general cómo actúa el módulo de un programa basado en diagramas y en una tablas de transiciones para reconocer el léxico y la sintaxis del programa fuente. En relación con esto, un analizador léxico se diseña a partir de diagramas y tablas de transiciones, el procedimiento de análisis, según esas dos representaciones, se utiliza para detectar si una cadena del programa fuente representa o no un nombre de variable aceptable y se sintetizan en los siguientes puntos (Brookshear 1995):

I. Procedimiento basado en diagramas de transiciones
   a. *Construir un diagrama de transiciones* que represente la estructura de un nombre de variable aceptable, teniendo en cuenta que esta estructura termina con distintos símbolos o marcas (espacio o punto y coma) que señalan el final de una cadena.
   b. *Escribir un segmento de programa* que reconozca estas estructuras *por medio de* determinados *algoritmos.*
   c. La composición de esos algoritmos se basa normalmente en *estructuras iterativas con condición inicial* (**mientras**...**hacer**), *estructuras alternativas* (**caso** variable **de**) dependiendo del estado actual y *estructuras condicionales* (**si** (condición) **entonces**) dependiendo de las opciones posibles de cada estado.

II. Procedimiento basado en tablas de transiciones
   a. *Asignar a una variable un valor inicial* correspondiente al estado inicial.
   b. *Actualizar de forma iterativa esta variable según la tabla* y de acuerdo a los símbolos de la cadena hasta llegar al final.
   c. *Escribir un segmento de programa* que reconozca nombres de variables aceptables.
   d. La estructura de los algoritmos básicamente es la misma que la que se usa en los diagramas de transiciones añadiendo la *estructura de repetición con condición final* (**repetir**...**hasta** (condición))

Para comprobar el procedimiento anterior partimos de la tabla de transiciones del AFD A (apartado 3.2):

f	a	b
→ * q_0	q_0	q_1
* q_1	q_0	q_2
* q_2	q_0	q_3
q_3	q_3	q_3

El segmento de programa del analizador léxico basado en esta tabla se podría especificar en el siguiente algoritmo, Tabla 3.1

TABLA 3.1: Adaptación del algoritmo de análisis léxico Brookshear (1993) , basado en AFD *A*

algoritmo Análisis léxico

Variable

a, *b* : Entrada;

q_0, q_1, q_2, q_3 : Estado;

q_0 : Estado Inicial;

q_0, q_1, q_2 : Estado Aceptar;

Fvariable

Estado := q_0 ;

Repetir

Leer el siguiente símbolo;

caso símbolo **de**

a, *b* : Entrada := símbolo;

Ninguno de los anteriores: salir a la rutina de error;

Estado := Tabla [Estado, Entrada];

hasta Estado = Estado Aceptar;

Frepetir

falgoritmo.

El analizador léxico basado en la tabla de transiciones del AFND A' (apartado 3.2.3.) se podría especificar en el siguiente algoritmo, Tabla 3.2

f	a	b	c	λ
→ *q_0	$\{q_0, q_1\}$	∅	∅	∅
*q_1	∅	$\{q_1, q_0\}$	$\{q_2\}$	$\{q_2\}$
*q_2	∅	∅	∅	∅

TABLA 3.2: Adaptación del algoritmo de análisis léxico Brookshear (1993) , basado en AFND A'

algoritmo Análisis léxico

Var

a, b, c, λ : Entrada;

q_0, q_1, q_2 : Estado;

q_0 : Estado Inicial;

q_0, q_1, q_2 : Estado Aceptar;

Fvar

Estado := q_0 ;

Repetir

Leer el siguiente símbolo;

caso símbolo **de**

a, b, c, λ : Entrada := símbolo;

Ninguno de los anteriores: salir a la rutina de error;

Estado := Tabla [Estado, Entrada];

si Estado = 'vacío' **entonces** salir a la rutina de error;

hasta Estado = Estado Aceptar;

Frepetir

falgoritmo.

El proceso anterior solo es una mera simplificación del funcionamiento del análisis léxico de los compiladores de lenguajes formales, que se podría adoptar al análisis del lenguaje natural, aunque siempre tenieno presente el problema de la ambigüedad. A pesar de lo anterior, nos interesa plantear el paralelismo de componentes, siguiendo con esta consideración si el módulo léxico se encarga de segmentar la cadena de símbolos en *tokens*, el módulo sintáctico del compilador se va a encargar de analizar aquellos *tokens* del programa fuente que representan construcciones algorítmicas del lenguaje de programación. De esta forma, el componente sintáctico tiene como objetivo analizar las estructuras básicas de estas construcciones como: declaración de variables, asignación, estructuras secuenciales, estructuras alternativas (**caso**...**de**), alternativas con condición (del tipo **si**...**entonces**...**sino**), estructuras iterativas con condición inicial (del tipo **mientras**...**hacer**) o iterativas con condición final (**repetir**...**hasta**). El componente sintáctico está dedicado, por tanto, a analizar el *patrón de componentes léxicos* (Brookshear 1995), es decir, el reconociendo de aquellos componentes léxicos que representan determinadas estructuras sintácticas.

Para que el proceso de traducción sintáctica del compilador se realice de forma eficaz es necesario emparejar las sentencias del programa fuente con determinadas reglas gramaticales propias de ese lenguaje de programación. Esto no supone normalmente ningún problema porque los lenguajes formales se definen por reglas prescritas que se cumplen con exactitud. La salida de esta etapa en la compilación consiste en generar un *árbol sintáctico* en el que las estructuras lingüísticas que forman el programa fuente se representan en función de la sintaxis propia del lenguaje de programación. Después de comprobar la sintaxis el compilador genera el programa en lenguaje máquina o programa objeto, tras otros procesos adicionales como puede ser *parser semántico*. Mediante todo estos procesos, el compilador se encargaría de transformar las Expresiones Regulares en código ejecutable.

Si el proceso de análisis sintáctico tiene como objetivo comparar sentencias de entrada y reglas gramaticales, en tanto en cuanto las sentencias estén sujetas a las reglas se podrá

realizar el análisis de forma eficaz. En los lenguajes de programación las sentencias se restringen a la rigidez de las reglas propias de ese lenguaje, de no ser así se obtendría un mensaje de error por parte del compilador. Por el contrario, las sentencias del lenguajes natural muchas veces no se corresponden con ninguna estructura, o se corresponden con varias estructuras posibles, volviendo a surgir el problema de la ambigüedad.

Para resolver este problema se puede recurrir a una modelización estadística de la lenguas naturales, en la que considerando la probabilidad de las distintas estructuras, se escoge la más probable. Si las probabilidades se representan en un AFD entonces estamos ante un modelo estocástico, o *Modelo de Markov*, en el que la probabilidad de una determinada transición está determinada por la transición precedente. En el caso de que una palabra pueda terner más de una categoría, es necesario utilizar algún tipo de mecanismo que se encargue de asignar la etiqueta correcta porque la entrada al analizador sintáctico no puede ser ambigua. Para la selección de la etiqueta más probable se pueden utilizar anotadores estocásticos, representados en un FST probabilísticos, o *Modelo Oculto de Markov.*

Hasta aquí hemos pretendido realizar una breve revisión muy general de las técnicas de estado-finito, fundamentalmente porque nos vamos a basar en esta tecnología para representar las variantes léxicas y sintácticas. Nuestro objetivo no es crear algoritmos para el reconocimiento de *patrones lingüísticos*, sino describir esos patrones en términos de Expresiones Regulares y desarrollar, en algunos casos los autómatas y transductores, que se podrían utilizar para reconocerlos. Pero los autómatas y transductores no son programas, por esa razón para el proceso de reconocimiento y agrupación vamos a recurrir a una aplicación lingüística que implemente esta tecnología.

Capítulo 4

METODOLOGÍA PARA LA REPRESENTACIÓN DE EXPRESIONES LÉXICAS Y SINTÁCTICAS CON TÉCNICAS DE ESTADO-FINITO

La identificación automática de información relevante en un texto a partir de determinadas construcciones léxicas o determinadas secuencias léxicas como son los sintagmas, antes que *unitérminos*, o *palabras-clave*, requiere el uso de métodos lingüísticos que a su vez necesitan el desarrollo de diversas herramientas lingüísticas. Con este objetivo es preciso desarrollar bases de información léxicas, o sintácticas, y aplicaciones informáticas que hagan uso de la información anterior. Las bases de información lingüísticas constituyen un componente necesario de los sistemas de indización y recuperación de información que manejan PLN. Pero no es el componente más importante, se trata sólo de una parte de dichos sistemas en los que se puede hacer uso de distintos recursos lingüísticos de carácter general, como diccionarios y gramáticas electrónicas, necesarios en las técnicas de *parsing*. Sin embargo, un *parser* de propósito general no es la herramienta más adecuada para los objetivos de la

indización, extracción o recuperación de información. Por esta razón, es preciso realizar un análisis morfológico y sintáctico orientado a objetivos concretos.

Para enfrentarnos al problema de la creación de las bases de información lingüísticas contamos con distintos modelos y algoritmos. En este trabajo vamos a adoptar dos modelos básicos: *Gramáticas Formales* y *Máquina de Estados*. Para operar con estos formalismos se utilizan los denominados algoritmos de «*búsqueda de espacio de estados*», que entrarían dentro de la categoría de autómatas y transductores capaces de decidir si una cadena de entrada pertenece a un lenguaje dado y, si es así, determinar, transformar o construir algún tipo de estructura para ella. En consecuencia, nuestro objetivo se centraría en plantear el problema del reconocimiento automático de expresiones léxicas y sintácticas en un *espacio de búsqueda de estados*.

La identificación automática de determinadas cadenas terminales, generadas por un tipo de Gramática Formal como son las Gramáticas Regulares, se realiza por medio de *Técnicas de Estado-Finito*. Y aunque las Gramáticas Regulares constituyan un formalismo bastante débil para generar y describir el lenguaje natural se imponen con bastante eficacia en aplicaciones prácticas específicas, sobre todo aquellas que requieren la identificación de subconjuntos del lenguaje natural, es decir, aquellos subconjuntos más fáciles de manipular debido esencialmente a que se pueden describir por medio de Expresiones y Relaciones Regulares.

Siguiendo con lo anterior, un componente clave de los sistemas que usan Técnicas de Estado-Finito en PLN es construir un conjunto de patrones, consistente en Expresiones Léxicas y Sintácticas, que se compilan en *Redes de Estado-Finito* y que a su vez pueden ser minimizadas y construidas de forma determinista para mejorar la velocidad de todas las operaciones. Las Expresiones Léxicas y Sintácticas se configuran como Expresiones Regulares, definidas como *una clase de lenguaje de programación de alto nivel para manipular cadenas, lenguajes y relaciones* que, además, permiten la realización de determinadas *operaciones basadas en el álgebra y en la teoría de conjuntos* (Karttunen 2000), como *clausura de Kleene*, *complementación* o *unión*. De alguna forma, las

Expresiones Regulares se pueden considerar como una especie de metalenguaje cuya eficacia se basa en su doble capacidad tanto para especificar cadenas de búsquedas textuales como para diseñar una clase concreta de máquina, constituyendo un método para representar de forma concisa un lenguaje y a su vez un método para describir el lenguaje aceptado por un determinado autómata.

A partir del planteamiento anterior, se van a establecer una serie de equivalencias entre Expresiones Regulares, Lenguajes Regulares y Autómatas Finitos, por una parte, y Gramáticas Regulares, Lenguajes Regulares y Autómatas Finitos. Estas correspondencias se establecen a partir de los *Teoremas de Análisis* y *Síntesis* de Kleene (Kleene 1956), en los que se demuestra que:

- Partiendo de un Autómata Finito se puede *extraer* el Lenguaje Regular que reconoce, y generar posteriormente la Expresión Regular que lo representa.

- Partiendo de una Expresión Regular, que define un Lenguaje Regular, se puede *construir* la Gramática Regular que genera dicho lenguaje, obteniéndose a continuación el Autómata Finito que reconoce el lenguaje generado por dicha Gramática.

De las anteriores equivalencias se deduce que las Expresiones Regulares sirven de nexo entre Lenguajes Regulares y Autómatas de Estado Finito. Además, existen determinadas operaciones simples con conjuntos de cadenas que se pueden describir con Expresiones Regulares y que por tanto son susceptibles de ser reconocidas por un Autómata Finito. Las citadas operaciones muy conocidas en la Teoría de Autómatas, algunas de las cuales ya se han tratado en otro capítulo, pero a las que nos vamos a volver a referir de forma más exhaustiva dada su importancia en la Teoría de Autómatas y por ser la base metodológica sobre la que se van a construir los analizadores léxicos y sintácticos. Sin embargo, antes de desarrollar este procedimiento es preciso plantear una cuestión previa: *cómo se obtienen los datos lingüísticos.*

4.1. Modelos lingüísticos de investigación y obtención de datos

Aunque no vamos a profundizar en complejas cuestiones sobre los modelos utilizados en la investigación lingüística, que requerirían un análisis muchos más extenso que excedería los límites de este trabajo, sí es necesario que nos detengamos en una cuestión metodológica previa: la relación entre *los métodos lingüísticos de investigación* y *la obtención de los datos*. En la investigación lingüística, la recopilación de los datos está vinculada a dos procedimientos generalmente aceptados:

a. *El método científico*, en el que los datos se obtienen de grandes *corpus textuales*.
b. *El método deductivo*, en el que los datos se obtienen de la *competencia lingüística*.

El método científico usa comúnmente un modelo empírico basado en lo que se denomina *lingüística de corpus*, a partir del cual se formulan hipótesis y se construyen teorías. El *corpus de datos* está constituido por un repertorio de información lingüística que se obtiene por la segmentación de una muestra representativa de una lengua. A su vez, los *corpora* pueden ser de muchos tipos: *monolingües*, *multilingües*, *etiquetados gramaticalmente*, o *etiquetados semánticamente*.

Los *corpora* se recopilan de forma bastante cuidadosa con el objetivo de que contengan textos de una gran variedad de fuentes y materias, a esto se añade que estos repertorios etiquetados son de gran utilidad para calcular la probabilidad asociada de cada término con una determinada categoría POS. De este modo, e independientemente de los distintos tipos de información asociada a la muestra, los *corpora* representan de forma exhaustiva todos los aspectos relevantes de una lengua, por esta razón se usan como base para estudios

estadísticos, tales como frecuencias de palabra, extracción de todas las co-ocurrencias de un término en particular, o localización y frecuencia de determinadas estructuras gramaticales

La mayoría de los *corpora* disponibles se han desarrollado para la lengua inglesa, de entre los primeros *corpora* recopilados se encuentran el *Brown Corpus* (BC) con un millón de palabras etiquetadas morfosintácticamente, el *British National Corpus* (BNC) que contiene más de 100 millones de palabras, el *Penn Treebank* con 3 millones de palabras anotadas automáticamente con categorías POS, el *IBM/Lancaster Treebank* con un *corpus* analizado de forma sintáctica, el *Birminghan Collection of English Text* con más de veinte millones de palabras, o el *Longman/Lancaster English Language Corpus* con treinta millones de palabras, entre otros. Para la lengua española contamos con el *Corpus de Referencia del Español Actual* (CREA), recopilado por la Real Academia Española (RAE), con unos 25 millones de palabras.

El prestigio de las técnicas de investigación empíricas está en que se limitan a realizar inferencias inducidas sólo a partir de la observación de los fenómenos lingüísticos en los *corpora de datos*. Estas inferencias tienen como objetivo *descubrir* el comportamiento de las unidades lingüísticas, por eso a este procedimiento también se le denomina *lingüística de corpus*, o *procedimientos lingüísticos de descubrimiento.* El método científico utiliza el *modelo inductivo-analítico*, que se adscribe básicamente a la investigación lingüística realizada dentro del paradigma estructuralista. En el estructuralismo, la recopilación de datos es una tarea fundamental del análisis lingüístico porque su objetivo es la *descripción* de las formas superficiales, a partir de las cuales se inducen generalizaciones de los fenómenos lingüísticos. En consecuencia, la relevancia científica del modelo empírico está en que utiliza métodos de descubrimiento mediante la observación, el análisis y la experimentación con objetivos *taxonómicos*, básicamente por medio de técnicas de segmentación y clasificación de las unidades existentes en los *corpora de datos*.

Sin embargo, el hecho de que el método científico se limite a los datos de los *corpora* hace que con esta premisa sólo se puedan formular teorías descriptivas a partir de los datos observables, y en consecuencia se considera un método incompleto para la explicación de

muchos fenómenos lingüísticos, o estructuras lingüísticas que no aparecen en los *corpora*. Frente al planteamiento anterior, la revolución que supuso la investigación llevada a cabo por Chomsky (Chomsky 1965) es que intentó desarrollar una *teoría explicativa* de los fenómenos lingüísticos. El cambio radical que originó la aportación chomskyana fue considerar una *gramática como una teoría explicativa de la estructura lingüística*, y a partir de esa teoría plantear hipótesis que se puedan comprobar de forma deductiva en los *corpora*. De alguna forma, la comprobación de las teorías hace este método utilice también *procedimientos empíricos*, aunque en un sentido diferente al método científico, como veremos.

Pero la cuestión que estamos planteando aquí es de dónde se obtienen los datos para la construcción de las gramáticas, si no lo hace de un *corpus* de datos. Aunque resulte sorprendente, dado el rigor matemático con el que Chomsky formula la *Teoría de la Gramática Generativa*, los datos se obtienen de la *introspección* que un hablante tiene de una lengua, o en otra palabras, del conocimiento implícito que todo hablante tiene de una lengua. Esos datos, según Chomsky, constituyen la *competencia lingüística*, definida formalmente como un *sistema finito de reglas* (Chomsky 1965). De esta forma, el conocimiento implícito de la lengua constituiría la competencia lingüística de un hablante ideal. A su vez un lingüista puede actuar como un hablante ideal, que tiene además conocimiento explícito de una lengua y capacitad para formalizar sistemáticamente el sistema de reglas.

A pesar de la importancia de la intuición y la introspección en el *modelo hipotético-deductivo*, las intuiciones puedan ser imprecisas o falsas por esta razón el lingüista debe especificar de forma explícita las estructuras lingüísticas y formalizar ese conocimiento en un conjunto de *reglas de producción*. Pero además, el lingüista debe formular hipótesis explicativas, sobre determinados fenómenos lingüísticos, que posteriormente se deben contrastar de forma empírica. Por tanto, sólo la exigencia de la formalización explícita de las hipótesis intuitivas y su comprobación con procedimiento empíricos pueden probar la *validez* de las teorías, que han planteado la formulación de esas hipótesis.

Por otra parte, partimos de que las gramáticas son *teorías científicas empíricas* que proporciona descripciones estructurales de las cadenas de una lengua y formalizaciones sistemáticas de cómo se generan por medio de un sistema de reglas. En relación con esto, hay sistemas matemáticos capaces de proporcionar una descripción de esas estructuras de forma automática, como son los *autómatas*. La vinculación entre la *Teoría científica* de una lengua y la *Teoría de autómatas* está en que se pueden considerar los dos extremos de la misma teoría: la primera representa un *modelo teórico* de la competencia lingüística para la generación de estructuras y la segunda un *modelo automático* de dicha competencia para el reconocimiento de tales estructuras. Dicho de otro modo, los *autómatas* proporcionan el procedimiento automático para el reconocimiento de las estructuras lingüísticas.

A su vez, como los autómatas son mecanismos que realizan operaciones matemáticas sobre las cadenas de entradas, las estructuras de las cadenas lingüísticas en los formalismos gramaticales se pueden representar también con modelos matemáticos. En consecuencia, si el modelo deductivo en investigación lingüística consiste en la derivación de hipótesis a partir de una supuesta *Teoría científica*, en este trabajo vamos a considerar la representación de esas hipótesis por medio de *Expresiones Regulares,* derivar a continuación las *Gramáticas Regulares* y obtener, por último, los *Autómatas de Estado-Finito* que son capaces de generarlas.

Una vez aclarado los supuestos metodológicos del *modelo hipotético-deductivo*, el planteamiento adoptado en este trabajo con este procedimiento se podría sintetizar de forma muy general en los siguientes pasos:

- Adoptar *la Teoría científica* de una lengua, o *Gramática*, y la *Teoría de autómatas* como marco general para las construcciones hipotéticas.
- Formular *hipótesis empíricas*, esto es, comprobables empíricamente, que *expliquen* el funcionamiento de los datos lingüísticos según nuestra *competencia lingüística* basándonos en las teorías anteriores. A su vez, las hipótesis empíricas pueden estar orientadas por los datos de un *corpus*, aunque no basadas en los datos de un *corpus.*

- Representar las *construcciones hipotéticas* en forma de *Expresiones Regulares*.
- *Obtener las Gramáticas Regulares* que expliquen el funcionamiento de los datos lingüísticos, por *derivación de las Expresiones Regulares*.
- *Establecer la equivalencia* entre las Gramáticas Regulares y los mecanismos automáticos de reconocimiento correspondientes: *Autómatas de Estado-Finito Gráficos.*
- Representar las *Gramáticas Regulares* por medio de *Transductores de Estado-Finito Gráficos*, cuyas entradas son *estructuras sintácticas etiquetadas* y cuyas salidas son *estructuras controladas*.
- *Verificar de forma empírica* las Gramáticas Regulares construidas, y compiladas en los distintos reconocedores de estado-finito, con el objetivo de comprobar *deductivamente las hipótesis.*

Aunque no siempre vamos a seguir de forma rigurosa los procesos anteriores, porque en algunos casos nos vamos a limitar a trasladar las expresiones directamente a los mecanismos de reconocimiento o autómatas. De cualquier forma, éste es el procedimiento que hemos adoptado y que se va a desarrollar a lo largo de todo este trabajo, y para ello vamos a comenzar con las operaciones básicas que se pueden realizar con las Expresiones Regulares, dada su relevancia en la formalización de las mencionadas *hipótesis empíricas*.

4.2. Operaciones con Expresiones Regulares

Las operaciones con Expresiones Regulares se desencadenan a partir de un Alfabeto Σ^*, definido por un conjunto de símbolos, y de un Lenguaje L, definido como conjuntos de cadenas sobre Σ^*. A partir de aquí, si L_1 y L_2 se consideran dos lenguajes, las operaciones que se pueden realizar sobre dichos lenguajes son:

- *Clausura de Kleene*, o clausura del lenguaje L, denotada por L^*, formada por el conjunto de concatenaciones de L incluyendo la palabra vacía. Así, definiendo $L^0 = \{\lambda\}$ y $L^i = LL^{i-1}$ para $i \geq 1$, la *cerradura, cierre o clausura de Kleene* del lenguaje L estaría formada por el conjunto de cero o más concatenaciones de L:

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

- *Clausura positiva de Kleene,* o clausura positiva del lenguaje L, denotada por L^+, formada por el conjunto de una o más concatenaciones de L:

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

- *Complemento* de un lenguaje, denotada por $\overline{L_1}$, formado por el conjunto de todas las cadenas que no están en el lenguaje L_1, es decir, $\overline{L_1} = \Sigma^* - L_1$
- *Concatenación* de L_1 y L_2, denotada por L_1L_2, formada por el conjunto $\{xy \mid x \in L_1 \quad \text{y} \quad y \in L_2\}$
- *Unión* de L_1 y L_2, denotada por $L_1 \cup L_2$, formada por el conjunto $\{x \mid x \in L_1 \quad \text{ó} \quad x \in L_2\}$
- *Intersección* de L_1 y L_2, denotada por $L_1 \cap L_2$, formada por el conjunto $\{x \mid x \in L_1 \quad \text{y} \quad x \in L_2\}$
- *Diferencia* de L_1 y L_2, denotada por $L_1 - L_2$, formada por el conjunto $\{x \mid x \in L_1 \quad \text{y} \quad x \notin L_2\}$

A un lenguaje de este tipo se le denomina *Lenguaje Regular* y puede estar representado por una Expresión Regular, a su vez una Expresión Regular r se define a partir del lenguaje $L(r)$ que genera. Asumiendo esto, los operadores que actúan sobre dichos lenguajes serían los mismos que actuarían sobre las expresiones, así dado un *Alfabeto*:

- Si $a \in \Sigma$, a es la Expresión Regular correspondiente al lenguaje $L(a) = \{a\}$ y λ es la Expresión Regular cuyo lenguaje es $L(\lambda) = \{\lambda\}$
- Si r y s son Expresiones Regulares definidas a partir de los Lenguajes $L(r)$ y $L(s)$:
  - (r) *es la Expresión Regular cuyo lenguaje es* $L(r)$
  - $(r)^*$ *es la Expresión Regular cuyo lenguaje es* $(L(r))^*$
  - $(\bar{r})$ *es la Expresión Regular cuyo lenguaje es* $\bar{L}(r) = \Sigma^* - L(r)$
  - $(r)(s)$ *es la Expresión Regular cuyo lenguaje es* $L(r)L(s)$
  - $(r) + (s)$, o $(r)/(s)$, *es la Expresión Regular cuyo lenguaje es* $L(r) \cup L(s)$
  - (r) & (s) es la Expresión Regular cuyo lenguaje es $L(r) \cap L(s)$
  - $(r)-(s)=(r)$ es la Expresión Regular cuyo lenguaje es $L(r) - L(s) = L(r)$

La prioridad de las operaciones dentro de una expresión se establece en el siguiente orden de preferencia: *1) paréntesis*, *2) clausura de Kleene*, *3) concatenación*, *4) unión*. Por otra parte, dos Expresiones Regulares r y s son equivalentes, es decir $r = s$, si describen el mismo Lenguaje Regular $L(r) = L(s)$. Existen, además, muchas equivalencias con respecto a Expresiones Regulares basadas en las *correspondientes igualdades de lenguajes* (Kelley 1995) pudiéndose demostrar diferentes propiedades o reglas asociadas a dichas expresiones como:

1. $\lambda^* = \lambda$
2. $r\,\lambda = \lambda\, r = r$
3. $\varnothing\, r = r\, \varnothing = \varnothing$
4. $r + \varnothing = r = \varnothing + r$
5. $r + r = r$
6. $\varnothing^* = \lambda$
7. $r^* r^* = r^*$
8. $r\, r^* = r^* r$

9. $r^* = r^{**} = r^* r^* = (\lambda + r)^* = r^*(r + \lambda) = (r + \lambda)\, r^* = \lambda + r\, r^*$
10. $r^* = \lambda + r^1 + r^2 + r^3 + \ldots + r^n + r^{n+1}\, r^*$
11. $r^* = (\lambda + r)^{n-1} + r^n\, r^*$
12. $(r^*)^* = r^*$
13. $r^+ = r\, r^*$
14. $r + s = s + r$
15. $(r + s) + t = r + (s + t)$
16. $(r\ s)\ t = r\ (s\ t)$
17. $r\ (s + t) = r\ s + r\,t$
18. $(r + s)\ t = r\ t + s\ t$
19. $(r + s)^* = (r^* + s^*)^* = (r^* s^*)^* = (r^* s)^* r^* = r^*(s\, r^*)^*$
20. $(r + s)^* = (r^*\ s^*)^* = (r^* + s^*)^*$
21. $(r\ s)^* r = r\ (s\ r)^*$
22. $(r^*\ s^*)\ r^* = (r + s)^*$
23. $(r^*\ s)^* = (r + s)^* + s + \lambda$
24. Regla de inferencia
    - Si $r = s^*\ t$ entonces $r = sr + t$
    - Si $\lambda \notin s$ entonces $r = s\ r + t \Rightarrow r = s^*\ t$

La unión tiene como elemento neutro $\varnothing$, además de ser asociativa y conmutativa, la concatenación tiene como elemento neutro λ y es asociativa y distributiva con respecto a la unión. A su vez, algunas reglas, como la 24, se pueden probar usando igualdades ya conocidas:

- $r = s^*\ t = (\lambda + s^+)\ t$ *puesto que* $s^* = \lambda + s^+$
- $= (\lambda + s\ s^*)\ t$ *por (13)*

- $= \lambda\ t + s\underline{s^*\ t}$ *por (17)*
- $= t + s\ r$ *por (2)*
- $= s\ r + t$ *por (14)*

4.3. Cálculo de Expresiones Regulares

Teniendo en cuenta las propiedades anteriores se puede demostrar la equivalencia entre Expresiones Regulares y Autómatas Finitos, a partir de su conexión con los Lenguajes Regulares. Con este objetivo, se tiene que determinar la relación entre lenguajes y autómatas por medio de las expresiones, que de alguna forma representan a ambos. Para el establecimiento de esta relación fue fundamental la aportación del denominado *Teorema de Kleene* (Kleene 1956) que, según Cohen en cuya versión del teorema nos vamos a basar, constituye el resultado más importante y fundamental de la Teoría de Autómatas Finitos. El teorema es el siguiente: *cualquier lenguaje que pueda ser definido por una Expresión Regular, o por un Autómata Finito, o por un gráfico de transiciones se puede definir a su vez por los tres métodos* (Cohen 1991).

El teorema anterior demuestra, en primer lugar, que todo *lenguaje que pueda ser definido por un Autómata Finito puede ser definido por un diagrama o gráfico de transiciones*, en segundo lugar, que todo *lenguaje que pueda ser definido por un diagrama de transiciones se puede definir también por una Expresión Regular* y, en tercer lugar, que todo l*enguaje que pueda ser definido por una Expresión Regular se puede definir también por un Autómata* (Cohen 1991), y con esto se demostraría que los tres métodos son equivalentes.

En el capítulo anterior ya se probó que cualquier lenguaje que pueda ser definido por un Autómata Finito se puede definir también por un diagrama de transiciones. En este capítulo nos vamos a centrar en la descripción de los dos supuestos restantes por su vinculación a las Expresiones Regulares, y por constituir la base metodológica de las técnicas de estado-finito.

A estas demostraciones se las denomina comúnmente *Teoremas de Análisis y Síntesis de Kleene*, su enunciado es el siguiente:

- *Teorema de Análisis*: todo lenguaje aceptado por un Autómata Finito es un Lenguaje Regular.
- *Teorema de Síntesis*: todo Lenguaje Regular es un lenguaje aceptado por un Autómata Finito.

Estos teoremas resuelven los denominados problemas de análisis y síntesis que están vinculados a las siguientes cuestiones:

- Probar que a partir de un Autómata Finito se puede obtener el Lenguaje Regular que acepta, y de ahí la Expresión Regular que lo representa.
- Probar que a partir de una Expresión Regular se puede generar el Autómata Finito que acepte el Lenguaje Regular descrito por dicha Expresión Regular.

4.3.1. El problema de análisis con Expresiones Regulares

Para dar solución al problema de análisis se aplican una seria de técnicas, denominadas *ecuaciones características o de variables*, que permitirán obtener una expresiones a partir de un autómata. Así, partiendo de un Autómata Finito, $\mathrm{A}' = (\Sigma, \mathrm{Q}, f, \mathrm{s}, F)$, representado en un *diagrama de transiciones* (Fig. 4.1), y con el objetivo de demostrar que a partir de este autómata se puede llegar a una Expresión Regular se van a seguir una serie de pasos que, según Kelley (Kelley 1995), se sintetizan en los siguientes:

- Para todo estado q_i de A', se define x_i como el conjunto de las cadenas sobre Σ que hace que A' pase desde el estado q_i hasta un estado de aceptación. Se dice que

x_i constituye *el conjunto de las cadenas aceptadas por el estado* q_i. Si desde q_i no se puede llegar a un estado final, $x_i = \varnothing$.

- Si $q_i \in F$, entonces $\lambda \in x_i$ y si $x_i \notin F$, entonces λ no está en x_i.
- Si $f(q_i, a) = q_j \in F$ entonces $a \in x_i$. Si $q_j \in f(q_i, n)$ entonces x_i contiene $n\,x_j$ (es decir, la concatenación del símbolo n y x_j, ó conjunto de cadenas que permiten pasar desde q_j a un estado final, debe estar en x_i, $n\,x_j \subseteq x_i$).
- De esta forma, x_i estaría formada por: $x_i = \cup \{ n\ x_j \mid q_j \in f(q_i, n) \}$.

Además, como se comprobará en la fase de resolución de ecuaciones, para poder realizar la descripción del autómata por medio de un sistema de ecuaciones se tienen que eliminar, cuando existan, las transiciones-λ de un estado hacia sí mismo así como los diferentes ciclos que se producen a través de transiciones-λ, o nulas, de un estado hacia sí mismos (esto se puede realizar transformado el autómata en AFND sin transiciones nulas). Teniendo en cuenta lo anterior, la formulación de *ecuaciones de variables* para cada estado, se obtiene asociando a cada estado una variable, de tal forma que el estado q_i tenga asociada la variable x_i, esto se realiza aplicando el siguiente proceso:

- La parte izquierda de cada ecuación debe aparecer la variable asociada al estado: si el estado es q_i se escribe en la parte izquierda de la ecuación x_i.
- La parte derecha debe aparecer los sumandos que se corresponden con los arcos de salida del estado (siempre teniendo en cuenta que *si desde* q_i *no se puede llegar a un estado final* $x_i = \varnothing$, es decir, no se obtendría ninguna palabra del lenguaje y de esto se deduce que *sólo se tendrán en cuenta aquellas transiciones que llevan a estados finales*) de tal forma que:

  a) Si el arco del estado q_i al estado q_j está etiquetado con el símbolo n, se suma $n\ x_j$ y si además el estado q_j es final se suma n.

b) Si el estado q_i es final se suma λ.

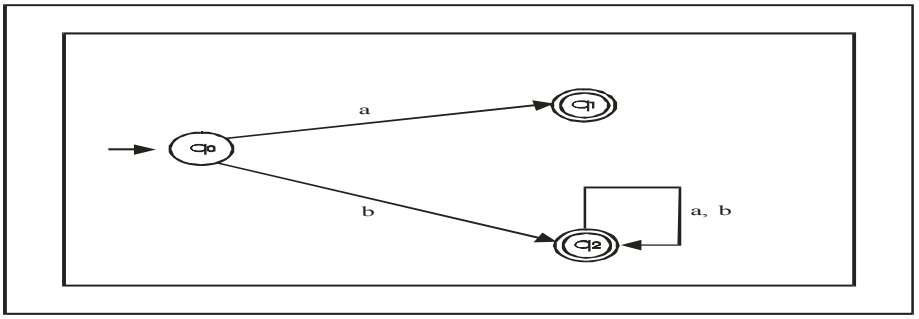


Fig. 4.1: Diagrama de transiciones del AFD A'

Basándonos en lo anterior, las ecuaciones características o fundamentales del Autómata Finito A' serían las siguientes:

$$x_0 = ax_1 + bx_2 + a + b$$

$$x_1 = \lambda$$

$$x_2 = ax_2 + bx_2 + a + b + \lambda$$

Para resolver las denominadas *ecuaciones fundamentales* de cada estado que tengan la forma $x_i = sx_i + r$ donde $\lambda \notin s$ se aplica la regla 24: si $\lambda \notin s$ entonces $r = sr + t \Rightarrow r = s^* t$ –con el objetivo de que se cumpla que $\lambda \notin s$ se tendrían que eliminar o no tener en cuenta las transiciones nulas de un estado hacia sí mismo o transformar el autómata en AFND sin transiciones nulas, pero en este caso no es necesario porque en el autómata A' no existen este tipo de transiciones– . Una vez calculado el valor de x_0, o ecuación del estado inicial, obtendríamos la Expresión Regular que representaría el conjunto de cadenas que hacen que el autómata pase de un estado inicial a un estado final, es decir, $x_0 = L(A')$. Iniciando el proceso a partir de los estados finales por las razones que ya se han expuesto, la resolución matemática de las ecuaciones anteriores sería la siguiente:

- Solución a la ecuación $x_2 = ax_2 + bx_2 + a + b + \lambda$

$$x_2 = (a + b)\,x_2 + (a + b + \lambda) \qquad \text{por (18)}$$

$$x_2 = (a + b)^* (a + b + \lambda) \qquad \text{por (24)}$$

$$x_2 = (a + b)^* (a + b) + \lambda \qquad \text{por (15)}$$

$$x_2 = (a + b)^* \qquad \text{por (9)}$$

- Solución a la ecuación $x_0 = ax_1 + bx_2 + a + b$

$$x_0 = a\lambda + b(a + b)^* + a + b$$

$$x_0 = a + b\left((a + b)^* + (a + b)\right) \qquad \text{por (2)(15)}$$

$$x_0 = a + b(a + b)^* \qquad \text{por (10)}$$

$$x_0 = b(a + b)^* + a \qquad \text{por (15)}$$

La solución a la ecuación característica del estado inicial, $x_0 = b(a + b)^* + a$, corresponde a la Expresión Regular, $b(a + b)^* + a$, que describe el Lenguaje Regular, $L = \{\{b(a + b)^* + a\}\}$, aceptado por el autómata A'.

4.3.2. El problema de síntesis con Expresiones Regulares

La solución al problema de síntesis se plantea de una forma opuesta al problema de análisis, en este caso se tiene que probar que partiendo de una Expresión Regular se puede construir el Autómata Finito que es capaz de reconocer el lenguaje representado por dicha Expresión Regular. Hay dos métodos para dar solución a este problema que son muy conocidos en la Teoría de Autómatas:

1. Asociar a cada posible Expresión Regular el Autómata Finito que reconoce el Lenguaje Regular que describe dicha Expresión Regular

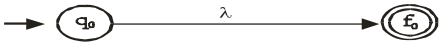
2. Calcular las *derivadas* de la Expresión Regular y obtener a continuación la *Gramática Tipo 3*, que genera el lenguaje descrito por dicha Expresión Regular. Posteriormente, realizar la equivalencia de dicha Gramática Regular con el Autómata Finito que es capaz de reconocer las cadenas generadas por ésta.

Si se asume que para cada Expresión Regular r hay un autómata que reconoce el lenguaje descrito por dicha expresión, $L(r) = L(A)$, la primera solución consistiría en transformar cada expresión en un autómata, de tal forma que si:

- $r = \varnothing$, el Autómata Finito (Fig. 4.2) que reconoce el lenguaje descrito por la Expresión Regular $\varnothing$ es:

Fig. 4.2: AFD que reconoce la expresión $\varnothing$

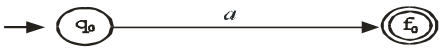
- $r = \lambda$, el Autómata Finito (Fig. 4.3) que reconoce el lenguaje descrito por la Expresión Regular λ es:



$r = \lambda$
$L(r) = L(A)$

Fig. 4.3: AFD que reconoce la expresión λ

- $r = a$, el Autómata Finito (Fig. 4.4) que reconoce el lenguaje descrito por la Expresión Regular a es:



$r = a$
$L(r) = L(A)$

Fig. 4.4: ADF que reconoce la expresión a

- $r = s^*$, el Autómata Finito (Fig. 4.5) que reconoce el lenguaje descrito por la Expresión Regular s^* es:

$r = s^*$
$L(r) = L(A)$

Fig. 4.5: ADF que reconoce la expresión s^*

- $r = s + t$, el Autómata Finito (Fig. 4.6) que reconoce el lenguaje descrito por la Expresión Regular $s + t$ es:

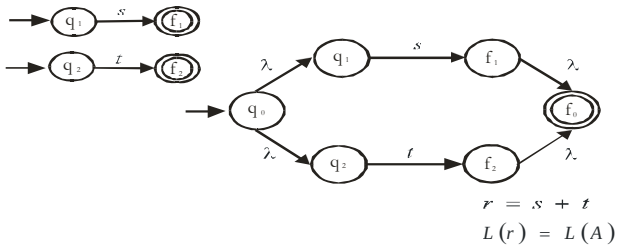


Fig. 4.6: AFD que reconoce la expresión $s + t$

- $r = st$ el Autómata Finito (Fig. 4.7) que reconoce el lenguaje descrito por la Expresión Regular st es:

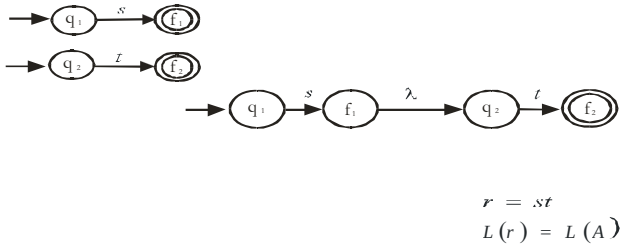


Fig. 4.7: AFD que reconoce la expresión st

La aplicación de este método genera autómatas demasiado grandes y complejos que posteriormente habría que minimizar, en consecuencia se trata de una técnica eficaz pero demasiado extensa para llevarla a la práctica. El segundo método es más eficaz para resolver el problema de síntesis, consistiría según Kelley (Kelley 1995) en la realización de tres etapas básicas:

1. *Cálculo de derivadas*.

2. *Obtención de la Gramática Tipo 3* generadora de la Expresión Regular.
3. *Obtención del Autómata*, que reconoce las cadenas generadas por la Gramática Tipo 3.

Según este segundo método, las derivadas de una Expresión Regular, r , respecto a un símbolo $a \in \Sigma$ están formadas por el conjunto de palabras, X, representadas por r que comienzan por el símbolo del que se deriva, es decir, el conjunto de cadenas cuya cabecera sea a:

$$D_a(r) = \{X \mid \quad a \cdot X \in r\}$$

Para calcular la derivada de una Expresión Regular asociada a un símbolo, o en otra palabras, el cálculo del conjunto de *restos o colas* de cadenas que comienzan por el símbolo respecto del que se deriva viene dado por las siguientes *reglas de derivación*:

$$D_a(\varnothing) = \varnothing$$
$$D_a(\lambda) = \lambda$$
$$D_a(a) = \lambda$$
$$D_a(b) = \varnothing$$
$$D_a(r^*) = D_a(r) \cdot r^*$$
$$D_a(r + s) = D_a(r) + D_a(s)$$
$$D_a(r \cdot s) = D_a(r) \cdot s + \alpha(r) \cdot D_a(s)$$

$$\text{donde} \quad \alpha(r) = \begin{cases} \lambda & \text{si } \lambda \in r \\ \varnothing & \text{si } \lambda \notin r \end{cases}$$

$$\alpha(r^*) = \lambda$$
$$\alpha(r) = \varnothing$$

$$D_a(r^+) = D_a(r\ r^*) = D_a(r) \cdot r^* + \alpha(r) \cdot D_a(r^*)$$
$$D_{ab}(r) = D_b(D_a(r))$$

Con el objetivo de calcular las derivadas de una expresión se aplicarían no sólo las reglas anteriores sino también las propiedades asociadas a dichas expresiones. Así, si tenemos la

Expresión Regular $b(a+b)^* + cb^*$ la forma de calcular sus derivadas con respecto a los símbolos a, b y c sería la siguiente:

$$r_0 = b(a+b)^* + cb^*$$

$$\begin{aligned} D_a(r_0) &= \left[D_a(b)\ (a+b)^* + \alpha(b)\ D_a(a+b)^* \right] + \left[D_a(c)\ b^* + \alpha(c)\ D_a\left(b^*\right) \right] \\ &= \left[\varnothing\ (a+b)^* + \varnothing\ D_a(a+b)\ (a+b)^* \right] + \left[\varnothing\ b^* + \varnothing\ D_a(b)\ b^* \right] \\ &= \varnothing \end{aligned}$$

$$\begin{aligned} D_b(r_0) &= \left[D_b(b)\ (a+b)^* + \alpha(b)\ D_b(a+b)^* \right] + \left[D_b(c)\ b^* + \alpha(c)\ D_b\left(b^*\right) \right] \\ &= \left[\lambda(a+b)^* + \varnothing\ D_b(a+b)\ (a+b)^* \right] + \left[\varnothing\ b^* + \varnothing D_b(b)\ b^* \right] \\ &= \left[\lambda(a+b)^* \right] + \left[\varnothing + \varnothing \right] \\ &= \lambda\ (a+b)^* = (a+b)^* = r_1 \end{aligned}$$

$$\begin{aligned} D_c(r_0) &= \left[D_c(b)\ (a+b)^* + \alpha(b)\ D_c(a+b)^* \right] + \left[D_c(c)\ b^* + \alpha(c)\ D_c\left(b^*\right) \right] \\ &= \left[\varnothing\ (a+b)^* + \varnothing\ D_c(a+b)\ (a+b)^* \right] + \left[\lambda\ b^* + \varnothing\ D_c(b)\ b^* \right] \\ &= \left[\varnothing + \varnothing \right] + \left[\lambda\ b^* + \varnothing \right] \\ &= \lambda\ b^* = b^* = r_2 \end{aligned}$$

A partir de las derivadas calculadas, $D_a(r_0) = \varnothing$, $D_b(r_0) = r_1$ y $D_c(r_0) = r_2$, se siguen derivando otras expresiones a partir de las Expresiones Regulares r_1 y r_2 con respecto a los símbolos a, b y c:

$$r_1 = (a+b)^*$$

$$\begin{aligned} D_a(r_1) &= D_a(a+b)\ (a+b)^* \\ &= \left[\ D_a(a) + D_a(b)\ \right] (a+b)^* \\ &= \left[\lambda + \varnothing \right] (a+b)^* \\ &= \lambda(a+b)^* \\ &= (a+b)^* = r_1 \end{aligned}$$

$$
\begin{aligned}
D_b(r_1) &= D_b(a+b)\,(a+b)^* \\
&= [\,D_b(a)+D_b(b)\,]\,(a+b)^* \\
&= [\varnothing+\lambda]\,(a+b)^* \\
&= \lambda(a+b)^* \\
&= (a+b)^* = r_1
\end{aligned}
$$

$$
\begin{aligned}
D_c(r_1) &= D_c(a+b)\,(a+b)^* \\
&= [\,D_c(a)+D_c(b)\,]\,(a+b)^* \\
&= [\varnothing+\varnothing]\,(a+b)^* \\
&= \varnothing\,(a+b)^* \\
&= \varnothing
\end{aligned}
$$

$$r_2 = b^*$$

$$
\begin{aligned}
D_a(r_2) &= D_a(b)\,b^* \\
&= \varnothing\,b^* \\
&= \varnothing
\end{aligned}
$$

$$
\begin{aligned}
D_b(r_2) &= D_b(b)\,b^* \\
&= \lambda\,b^* \\
&= b^* = r_2
\end{aligned}
$$

$$
\begin{aligned}
D_c(r_2) &= D_c(b)\,b^* \\
&= \varnothing\,b^* \\
&= \varnothing
\end{aligned}
$$

Las derivadas de r_1 y r_2 serían respectivamente:

$$D_a(r_1) = r_1 \qquad D_b(r_1) = r_1 \qquad D_c(r_1) = \varnothing$$

$$D_a(r_2) = \varnothing \qquad D_b(r_2) = r_2 \qquad D_c(r_2) = \varnothing$$

Después de calcular todas las derivadas de una expresión, se construye la Gramática Regular que genera el Lenguaje Regular representado por dicha expresión. Esta gramática estará compuesta por: el conjunto de símbolos del alfabeto, el conjunto de derivadas obtenidas de la Expresión Regular de la que se parte, r_0, y las reglas de producción P que tienen la forma siguiente:

- Si $D_a(R) = S$, se crea la regla $R ::= aS$ ó $R \rightarrow aS$
- Si $D_a(R) = \varnothing$, no se crea ninguna regla
- Si $\lambda \in D_a(R)$, se crea la regla $R ::= a$ ó $R \rightarrow a$
- Si $\lambda \in R_0$, se crea la regla $R_0 ::= \lambda$ ó $R \rightarrow \lambda$

De forma que la Gramática Tipo 3 que genera el lenguaje definido por la Expresión Regular $r_0 = b(a+b)^* + cb^*$ se define como:

$$G = (\{a,b,c\}\{r_0, r_1, r_2\}, r_0, P)$$

donde P es:

$r_0 ::= b\ r_1 \mid c\ r_2 \mid b \mid c$
$r_1 ::= a\ r_1 \mid b\ r_1 \mid a \mid b$
$r_2 ::= b\ r_2 \mid b$

Una vez obtenida la gramática, el paso siguiente es calcular el AF equivalente que reconozca el lenguaje generado por dicha gramática. Este autómata estará compuesto por los siguientes elementos: el conjunto de símbolos del alfabeto, el conjunto de derivadas obtenidas de la Expresión Regular de la que se parte, la Expresión Regular de la que se parte, r_0, y el conjunto de derivadas, de modo que:

- Si $D_a(r_i) = r_j$ y $r_j \neq \lambda, r_j \neq \varnothing$, entonces $r_j \in f(r_i, a)$
- Si $D_a(r_i) = \varnothing$, entonces $f(r_i, a) = \varnothing$
- Si $\lambda \in D_a(r_i)$, entonces $F \in f(r_i, a)$
- Si $\lambda \in r_0$, entonces $F \in f(r_0, \lambda)$

Según esto, el AF que reconoce el lenguaje generado por la Expresión Regular $r_0 = b(a+b)^* + cb^*$ sería el siguiente:

$$AF = (\{a, b, c\}, \{r_0, r_1, r_2, F\}, f, r_0, \{F\})$$

donde f se define como:

$f\ (r_0, a) = \varnothing$
$f\ (r_0, b) = \{r_1, F\}$
$f\ (r_0, c) = \{r_2, F\}$
$f\ (r_1, a) = \{r_1, F\}$
$f\ (r_1, b) = \{r_1, F\}$
$f\ (r_1, c) = \varnothing$
$f\ (r_2, a) = \varnothing$
$f\ (r_2, b) = \{r_2, F\}$
$f\ (r_2, c) = \varnothing$
$f\ (F, a) = \varnothing$
$f\ (F, b) = \varnothing$
$f\ (F, c) = \varnothing$
$f\ (\varnothing, a) = \varnothing$
$f\ (\varnothing, b) = \varnothing$
$f\ (\varnothing, c) = \varnothing$

A continuación, el AF se puede representar en una *tabla de transiciones* (Fig. 4.8):

f	a	b	c
→ r_0	∅	r_1, *F*	r_2, *F*
r_1	r_1, *F*	r_1, *F*	∅
r_2	∅	r_2, *F*	∅
* *F*	∅	∅	∅
∅	∅	∅	∅

Fig. 4.8: Tabla de transiciones del AF equivalente a la Gramática Tipo 3

Por último, se construye el Autómata Finito Determinista Mínimo equivalente y se renombran los estados. El AF resultante se puede representar a su vez en una *tabla de transiciones* (Fig.4.9), o en un *gráfico de transiciones* (Fig. 4.10).

f	a	b	c
→ q_0	∅	q_1	q_2
* q_1	q_1	q_1	∅
* q_2	∅	q_2	∅
∅	∅	∅	∅

Fig. 4.9 : Tabla de transiciones del AFD Mínimo equivalente a la Gramática Tipo 3

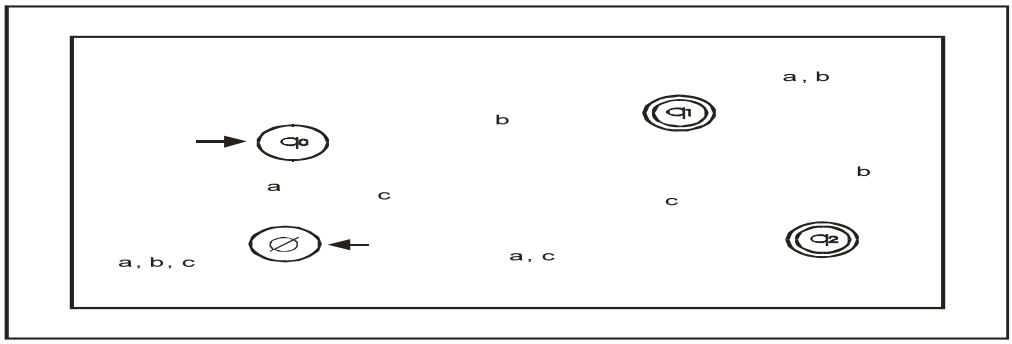


Fig. 4.10 : Diagrama de transiciones del AFD Mínimo equivalente a la Gramática Tipo 3

Además sobre las Expresiones Regulares intervienen determinados operadores, algunos de los cuales ya se han comentado, que dan lugar a Expresiones Regulares Complejas. Estos operadores actúan sobre expresiones simples como hemos visto, que a su vez pueden estar constituidas por conjuntos de cadenas, representadas por *Autómatas Finitos*, o por pares de cadenas, representadas por *Transductores Finitos*.

Una vez delimitadas las técnicas básicas utilizadas por los formalismos de estado-finito y cómo se derivan las gramáticas a partir de las Expresiones Regulares, el paso siguiente es aplicar dichas técnicas a la identificación y agrupación de expresiones sintácticas, aunque previamente se tienen que desarrollar las herramientas de análisis léxico. En los próximos apartados, se va a exponer el procedimiento para la construcción de los analizadores lingüísticos, haciendo uso de una aplicación desarrollada por Silberztein (Silberztein 1999).

4.4. Metodología para la Representación de Expresiones Léxicas con Técnicas de Estado-Finito

El análisis de expresiones léxicas y sintácticas se realiza a través de distintos niveles de descripción lingüística básicamente: *parsing* morfológico, o léxico, y *parsing* sintáctico. El primero tiene como objetivo obtener una segmentación de la palabra en una cadena de morfemas, la fuente fundamental de conocimiento implicado en este análisis es el *lexicón o diccionario electrónico*; el segundo tiene como finalidad básica proporcionar una estructura de los sintagmas que manifieste las relaciones sintácticas, la fuente fundamental de conocimiento en este análisis es una *gramática electrónica*. El componente léxico es esencial porque en último extremo tanto los sintagmas como las frases están formadas por palabras –este planteamiento tiene su origen en las reglas de producción de las gramáticas en general, debido a que dichas reglas se pueden clasificar, como ya se señaló en otro capítulo, en dos tipos: las denominadas *reglas categoriales o sintagmáticas*, formadas por símbolos no terminales, y las denominadas *reglas léxicas*, formadas por símbolos terminales o cadenas en las que se deposita, en última instancia, la información morfológica, sintáctica y semántica–.

Para que se pueda analizar la estructura de una sentencia es necesario que ésta haya sido previamente divida en *tokens*, o palabras separadas por espacios en blanco. A su vez, para analizar dichas palabras es preciso que éstas se sub-dividan en determinados morfemas. Con el objetivo de distinguir los morfemas que forman un palabra, los analizadores léxicos cuentan habitualmente con el lexicón y con un conjunto de reglas. Con la intervención de estos dos componentes una palabra se analiza en una secuencia de morfemas, a la que se añade las característica *part-of-speech* (POS). El resultado del análisis léxico consistirá en la asignación de las distintas categorías léxico-gramaticales a las unidades léxicas, incorporación de este modo información morfológica y propiedades sintácticas de concordancia. De forma simplificada, a esta información se la denomina habitualmente *base*

de información léxica y es la que se almacena en forma de *diccionarios electrónicos*. El resultado de este procesamiento se transmitirá al *parser* sintáctico.

Uno de los problemas que se le plantean a los analizadores léxicos cuando se aplican al PLN es la determinación de la unidad léxica que se va a tomar como unidad de reconocimiento. Aunque parezca un proceso sencillo, sólo lo es en apariencia. La determinación de las unidades que se van a seleccionar como medida de reconocimiento léxico presenta bastantes dificultades. La distinción entre palabras, *unidad lingüística mínima con significado independiente*, y morfema, *unidad lingüística mínima con significado propio*, se plantea aquí como una cuestión relevante.

La palabra tiene una estructura compleja que se constituye básicamente a partir de estructuras elementales como: desinencia, radical o *stem*, raíz, prefijos y sufijos. La delimitación de cada una de estas subunidades la expuso de forma bastante precisa Saussure (Saussure 1945) y, a pesar del paso del tiempo, no ha perdido su vigencia:

a) *Desinencia* es la característica flexional o elemento variable del *fin de palabra* que distingue las formas de un paradigma nominal o verbal. Por otra parte, el fin de palabra puede estar representado por la *desinencia cero*.
b) *Tema de flexión*, radical o *stem*, se obtiene por la eliminación de la *desinencia*, se trata del elemento separado espontáneamente por la comparación con una serie de palabras emparentadas y que lleva la idea común de todas ellas. Se considera por tanto una unidad básica que se encuentra en un lugar intermedio entre la *raíz* y la *forma flexiva,* nominal o verbal, considerada globalmente.
c) *Raíz* es el elemento irreducible y común a todas las palabras de una misma familia. La raíz es el componente o elemento *significativo* de la palabra, no analizable morfológicamente y común a todas las palabras emparentadas. Se trata de la parte que alcanza el máximo grado de *abstracción* y de *generalidad*. Teniendo en cuenta que una palabra representa siempre una idea relativamente determinada desde el punto de vista gramatical, esta característica se contradice con la abstracción propia

de la raíz y, por tanto, una raíz no puede *constituir una palabra ni recibir la adjunción directa de una desinencia*.

d) El prefijo precede a la parte de la palabra reconocida como radical y el sufijo constituye el elemento que se añade a la raíz para hacer de ella un radical. De alguna forma, el *análisis de prefijos y sufijos representa otra versión del reconocimiento del radical o stem.*

Nuestro interés se centra en el análisis de los radicales o *stems* que son los que permiten la inmediata inserción de elementos de flexión. Muchas veces, el *stem* y la raíz pueden coincidir si no aparecen afijos derivativos (como `libro-` en `libro-s`), en cuyo caso se habla de *stem simple*, o puede estar formado por una raíz y un afijo (como *`organiz-ador`* en *`organiz-ador-es`*, *`organiz-ador-a`*, *`organiz-ador-as`*), en este último caso se habla de *stem derivado*. A su vez, una palabra puede estar compuesta por dos o más *stem*s, en cuyo caso se habla de *stem compuesto* (como *`neurotransmisor`*, *`fotodegradable`*, *`hispanohablante`*,...). En lo que respecta a los radicales verbales, éstos están constituidos por una raíz y una vocal (*`infrorm-a`*, *`comprend-e`*,*`comprim-i`*). La distinción entre *radicales vocálicos*, o acabados en vocal, y *no vocálicos*, o acabados en consonante, es importante porque determinará, como veremos en capítulo siguiente, la adscripción del *stem* a las distintas clases flexivas y, en el caso de los radicales verbales, la vocal nos indicará la pertenencia del verbo a la primera, segunda o tercera conjugación (*`infrorm-a-r`*, *`comprend-e-r`*,*`comprim-i-r`*).

Por otra parte, tenemos que existen las denominadas palabras de *clase abierta*, en las que se incluyen unidades léxicas portadoras de significado (como nombres, verbos, adjetivos y algunos adverbios), y las palabras de *clase cerrada*, en las que se incluyen elementos sub-léxicos dotados sólo de información estructuralmente relevante (como artículos, pronombres, preposiciones, conjunciones, ...).

El análisis de las unidades anteriores es propio del estudio descriptivo de las palabras, esto es, del estudio de las formas y de las alteraciones formales de las palabras. Precisamente estas cuestiones entran dentro del objeto de estudio de la *morfología* que tiene como finalidad básica el análisis, función y distribución de las unidades mínimas obtenidas del análisis de las estructuras formales de las palabras o, lo que es lo mismo, de los *morfemas*. Los morfemas son unidades mínimas no analizables en otras unidades, que se pueden clasificar según diversos criterios como *autonomía*, *significado léxico*, o *función*. De forma general, se asume la siguiente ordenación:

- *morfemas libres* o *afijos* y *morfemas ligados*, según el criterio de autonomía;
- *morfemas léxicos* y *morfemas gramaticales*, según el criterio de significado;
- *morfemas fundamentales o de base*, *morfemas flexivos* y *morfemas derivativos*, según el criterio de función.

Por otra parte, los morfemas ligados o trabados, es decir, aquellos que sólo se pueden usar en combinación con morfemas libres, se clasifican habitualmente según su posición respecto del morfema libre en: *prefijos*, *infijos* y *sufijos*. Los morfemas gramaticales son casi siempre ligados, pero los morfemas léxicos pueden ser *libres*, cuando la raíz no se puede subdividir en partes más pequeñas, o *trabados*, cuando la raíz no se considera una palabra independiente, esto es, cuando se trata de una *raíz trabada*. Por esta razón, es necesario aclarar que la clasificación anterior no es excluyente, en el sentido de que un morfema trabado puede tener significado léxico (como la raíz trabadas *informa–*), o sólo significado gramatical (como algunas terminación, *–r*). De la misma forma un morfema libre puede tener significado léxico (como *visual, local,...*) o sólo significado gramatical (como el morfema de futuro o condicional del inglés *shall* o *will*).

En relación con el proceso de formación de palabras, tenemos que se puede realizar por *derivación* o por *composición*. En el proceso de formación de palabras por derivación uno de los componentes es un morfema libre o léxico y el otro es un morfema trabado, que puede cambiar la categoría del morfema léxico –como *recuper-a-r (Verbo)* + –able =

recuperable(Adverbio) o matriz(Nombre) + –cial = matricial(Adjetivo)–. Frente a esto, el proceso de formación de palabras por composición se basa en las relaciones sintagmáticas, o relaciones de sucesión, de las unidades léxicas. Así, la formación de compuestos se realiza mediante la combinación de varias unidades léxicas que se puede descomponer a su vez en otras unidades libres. Muchas veces las unidades pertenecen a distintas clases de palabras –como Nombre + Adjetivo = Adjetivo– en estos casos suele prevalecer la categoría del segundo término; otras veces, la palabra compuesta tiene significado propio y no es reducible a sus partes. Por otra parte, hay que tener en cuenta que el proceso de composición está fuertemente vinculado con la categoría de palabras de clase cerrada mientras que el proceso de derivación está relacionado con la categoría de palabras de clase abierta.

A la vista de las dificultades que plantea la elección de los elementos de análisis nos vamos a limitar, por los objetivos prácticos de este trabajo, básicamente al análisis del *radical* y, en consecuencia, el tratamiento de la morfología se va a restringir a dos estructuras básicas:

a) *Forma canónica*, radical o *stem*, compuesta por el morfema de base que no es analizable morfológicamente y es común a todas las palabras de la misma clase. En un análisis descriptivo, sería lo que queda después de la eliminación de todos los morfemas, o *sufijos flexivos / derivativos*.

b) *Morfemas*, o unidades que se combinan con el radical modificando su función y significado gramatical. A su vez, estos morfemas se pueden clasificar en:

   - *Flexivos* que proporcionan información gramatical acerca de una forma canónica, o de una serie de formas canónicas, de este tipo serían los morfemas de género y número, o los morfemas de la conjugación verbal.

   - *Derivativos* que permiten la formación de nuevas palabras proporcionando información de la clase de palabra que se constituyen por composición con

determinados *stems* –a este tipo pertenecen los *afijos*, tanto *prefijos* como *sufijos*, que forman parte de las palabras–.

El problemas que surge a continuación es la determinación de la información que se va a almacenar en el *lexicón*, o qué distinciones tanto gramaticales como morfológica son relevantes para el procesamiento y el reconocimiento de las unidades léxicas. En muchos casos el lexicón se desarrolla para la aplicación concreta en la que se va a utilizar y suele adoptar dos tipos de construcción:

- Como una lista exhaustiva de todas las entradas léxicas de la lengua que se vaya a analizar.

- Como una lista parcial de las entradas léxicas en la que se distinguen, por un lado, los radicales o *stems* de las palabras así como los afijos que se agregan a los *stems* en los procesos morfológicos de flexión y derivación, y, por otro, un conjunto de reglas léxicas y morfológicas, que se configura como una auténtica *gramática de la palabra* (*word grammar*) para llevar a cabo dichos procesos e implantándose como un componente analítico de cadenas.

Una simple lista de todas las formas léxicas que pueden aparecer en una lengua daría lugar a una construcción demasiado extensa a la que habría que poner necesariamente algún tipo de limitación. Por el contrario, el diseño de una lista parcial es más adecuado, como se demostrará a continuación, y el papel de las reglas morfológicas aunque no sea esencial sí puede resolver el problema de determinadas irregularidades en las combinaciones entre *stems* y afijos.

Siguiendo con lo anterior, el análisis morfológico tiene como objetivo la identificación de patrones léxicos, definidos por cadenas de morfemas, según el modelo de representación léxica que se incluya en el lexicón, o diccionario electrónico. De esta forma, la morfología, y por extensión las reglas morfológicas, se implanta como una cuestión relevante en el modelo

de representación del lexicón. Básicamente, la morfología se define como el estudio de la formación de palabras y de la clase de palabra, o clase de constituyente, al que pertenece cada palabra. En esencia, la morfología se ocupa de las alteraciones formales de las palabras, que están en relación con el grupo o la clase a la que pertenezcan dichas palabras, comúnmente se divide en dos categorías:

1. *Morfología flexiva*: se circunscribe a la combinación de un *stem* con un morfema gramatical, tiene como representantes más claros el género, el número, o las formas verbales. La morfología flexiva establece la concordancia y las relaciones gramaticales entre las palabras dentro de una frase, en consecuencia el contenido de la flexión tiene propiedades, como género y número, que afectan a las construcciones sintácticas y es, por tanto, *obligatorio*. A su vez, la forma en la que los *stems* se combinan con los morfemas gramaticales responde a determinadas reglas morfo-sintácticas que hacen que las flexiones se realicen de forma regular, aunque presenten algunas irregularidades.

2. *Morfología derivativa*: se ocupa de cómo se derivan unas palabras de otras dando lugar a transformaciones en el significado y en ocasiones a un cambio en la clase de palabra. La morfología derivativa afecta fundamentalmente al significado de las palabras pero no a las relaciones gramaticales de las construcciones sintácticas y , en consecuencia, su aplicación es *opcional*. Además, aunque la construcción de derivados se realice por la aplicación de distintas reglas, éstas presentan muchas irregularidades produciendo derivados alejados semánticamente de la raíz origen.

En un sentido muy general, la morfología flexiva se ocuparía de la flexión del *stem* de las palabras por medio de *sufijos flexivos* y la morfología derivativa de la formación de nuevos *stems* de palabras por medio de *afijos derivativos*. En este trabajo, se van a tener en cuenta los afijos derivativos cuando formen parte del *stem*, pero se va a excluir una representación exhaustiva de los mismos, por las razones ya expuestas –que están en relación con la irregularidad y la no obligatoriedad con la que se comporta el proceso de derivación, con la

dificultad añadida de que la derivación puede cambiar la categoría de la palabra y esto complica su representación y almacenamiento–.

En relación con las formas flexionadas, que sí se van a representar de forma exhaustiva, partimos de que se componen de determinadas partículas que se asocian a los radicales y no cambian la categoría gramatical de dichas palabras. El radical, o *stem*, de la palabra se establece como un elemento común de una serie de formas flexionada que se encuentran en mutua relación paradigmática. El modelo de flexión de las clases de palabras como sustantivo, adjetivo, artículo, pronombre o numeral es el *género* y el *número*, además en el adjetivo también se puede diferenciar entre comparativo y superlativo, frente a esto, el modelo de flexión verbal básicamente es la *persona*, el *número*, el *modo*, el *tiempo* y el *aspecto*.

Por otra parte, la representación de la morfología flexiva en el lexicón, formado por `stem + afijos`, implica tener en cuenta los problemas e irregularidades que presentan los sufijos flexivos. El análisis léxico realizado por Autómatas de Estado-Finito permite integrar en el lexicón las combinaciones de morfemas que se consideran válidas así como las alteraciones que se puedan producir en dichas combinaciones. De forma simplificada, Jurafsky y Martin (Jurafsky y Martin 2000) sintetizan un método sencillo para integrar la morfología en el lexicón, representado por un AFD, también denominado *analizador de 1-nivel*:

- Primero se capturan las combinaciones posibles entre morfemas, lo que se denomina *morfotáctica* (Fig. 4.11).

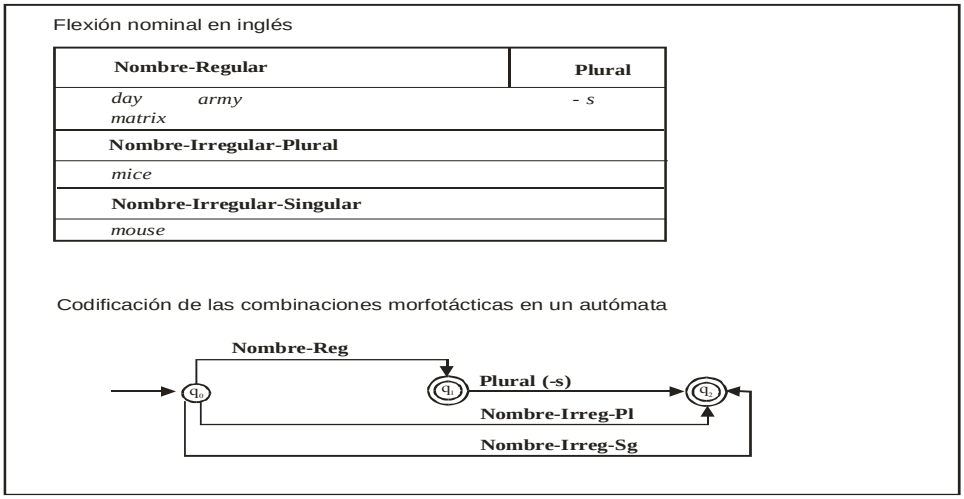


Fig. 4.11: Representación de las combinaciones de morfemas en AFD

- Después, se integran las palabras y las combinaciones de morfemas válidas en el lexicón (Fig. 4.12).

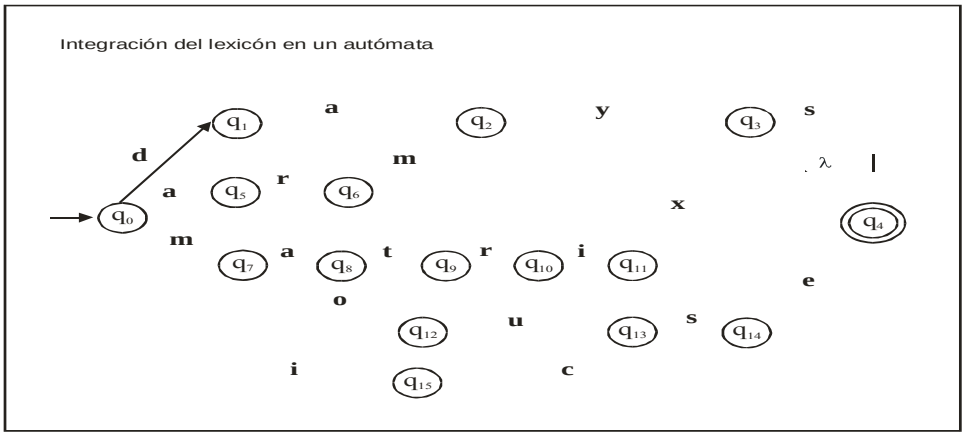


Fig. 4.12: Representación del lexicón en un analizador de *1-nivel*

Además, la función de los analizadores léxicos no es sólo reconocer las combinaciones válidas sino asignar etiquetas léxico-gramaticales a las unidades léxicas y señalar el mecanismo que ha dado lugar a su formación, así como indicar distintas propiedades de concordancia sintáctica con otras unidades. El resultado del análisis léxico radica en la incorporación de toda esta información a las diferentes unidades léxicas. Este proceso se

realiza también a partir de la información que está contenida en diccionarios electrónicos de formas canónicas, o lexicones computacionales, y de un conjunto de reglas implementadas en transductores, configurados como *analizadores de dos-niveles*. El proceso general para crear los analizadores de *dos-niveles*, representados en Transductores de Estado-Finito, *Finite-State Transducers* (FST), se sintetiza en:

1. *Ampliar un AFD con una cinta, tape, extra.*
2. *Agregar símbolos extra a las transiciones de un AFD*

En la medida en que el *parsing* morfológico consiste precisamente en tomar una palabra de entrada y crear una estructura para ella, un analizador de dos-niveles se puede considerar una herramienta adecuada para llevar a cabo este proceso. Esta idea tiene su origen en un acertado planteamiento: entre la palabra de entrada, o *Forma Superficial*, y su estructura, *o Forma Léxica*, se establece una *Relación Regular* que se puede compilar en un Transductor de Estado-Finito (Karttunen, Kaplan y Zaenen 1992) (Fig. 4.13). Con el objetivo de introducir la Relación Regular, el transductor procede de la siguiente forma:

- Lee de una cinta, usando el *Segundo Símbolo* de cada transición.
- Escribe en la segunda cinta, usando el *Primer Símbolo* de cada transición.

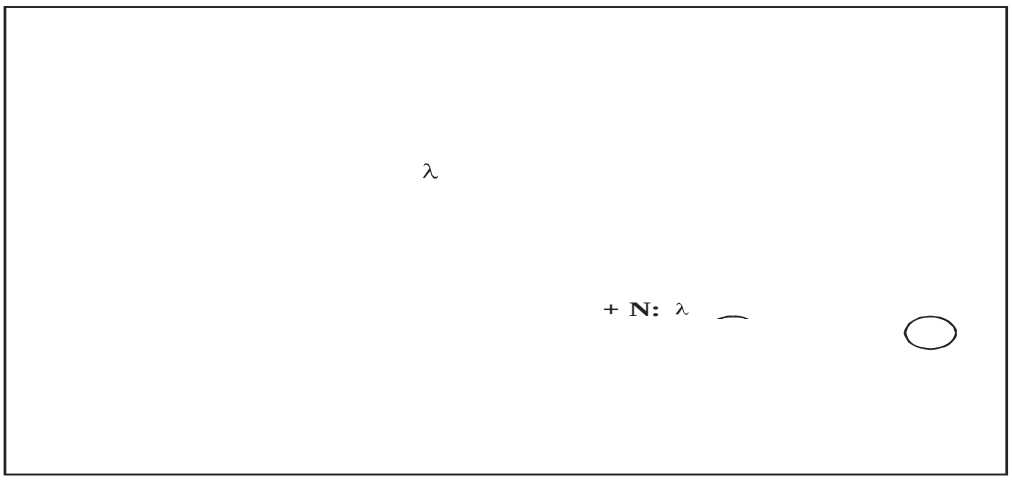


Fig. 4.13: Representación de Relaciones Regulares en un FST

Los *analizadores de dos-niveles* equiparan *Formas Superficiales* a *Formas Léxicas* y viceversa, porque son bidireccionales lo que implica que se pueden emplear tanto en el reconocimiento como en la generación de cadenas. Para realizar este proceso las Formas Léxicas se encuentran en un diccionario, o lexicón, en el que se representan las *formas canónicas* seguidas por una *secuencia de etiquetas* que muestran características morfológicas y categorías sintácticas o gramaticales. El proceso para integrar la información anterior en el lexicón, representado en este caso por un FST, se puede sintetizar según Jurafsky y Martin (Jurafsky y Martin 2000) como sigue:

- Primero, se capturan las secuencias de morfemas en su representación canónica y las secuencias de etiquetas con las características morfológicas y las categorías gramaticales (Fig. 4.14)

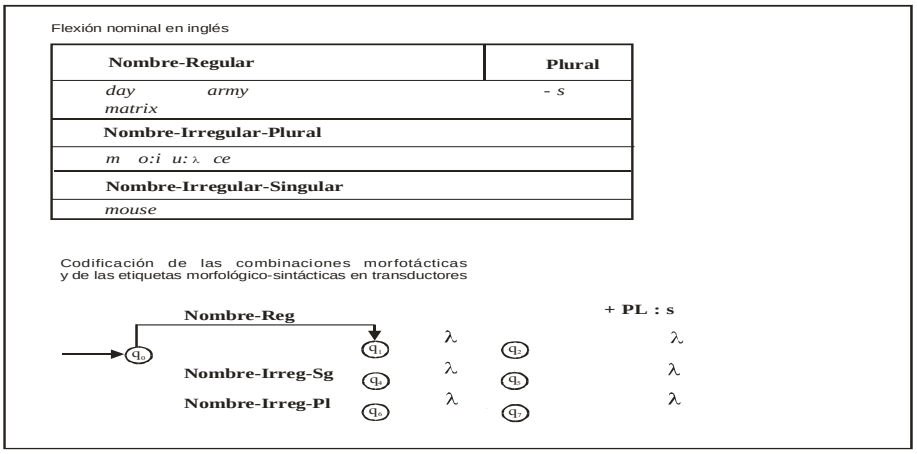


Fig. 4.14 : Representación de rasgos morfológicos y etiquetas sintácticas en FST

- Después, se integran las formas canónicas, los rasgos morfológicos y las categorías gramaticales o sintácticas en el lexicón representado en este caso por un transductor (Fig. 4.15)

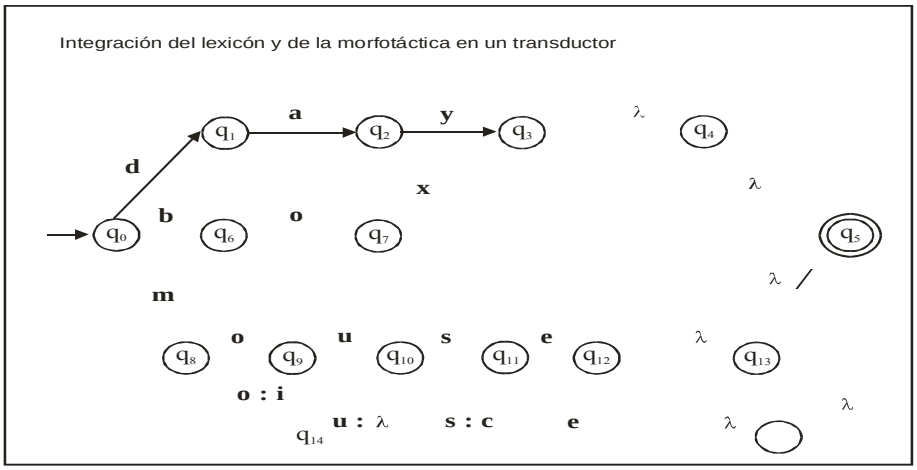


Fig. 4.15: Representación del lexicón en un analizador de *2-niveles*

Un *analizador de dos-niveles* recibe una cadena superficial y genera su transducción léxica, de ahí la denominación de *dos-niveles* –nivel superficial y nivel léxico–, además de poder reconocer cadenas, cuando se utiliza para establecer si la equiparación entre la cadena superficial y la léxica es una correspondencia válida. Las *Formas Léxicas* se configuran como Expresiones Regulares, en este caso expresiones léxicas almacenadas en forma de listas de morfemas en un diccionario de formas canónicas compuesto por `stem + afijos + categorías pos`, y un conjunto de reglas morfológicas aplicadas de forma paralela entre los dos niveles y compiladas en FST. Las reglas representan las combinaciones válidas de morfemas y se encargan de relacionar cadenas superficiales a cadenas léxicas, de modo que las distintas formas flexionadas y las formas irregulares se equiparen a la misma forma léxica (Fig. 4.16):

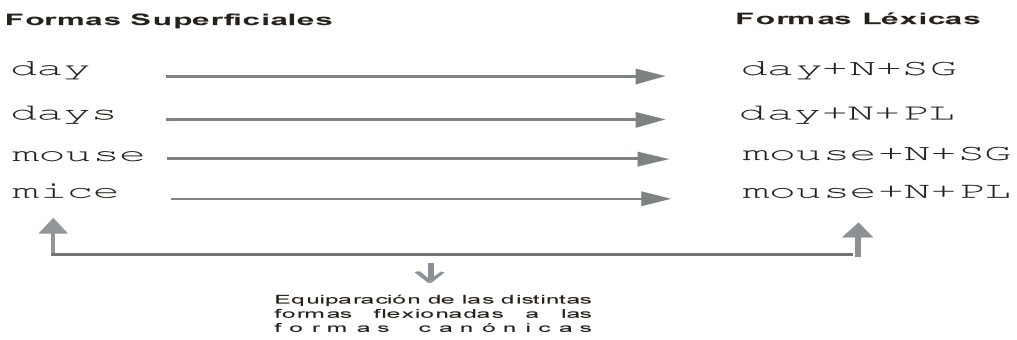


Fig. 4.16: Correspondencia entre *Formas Superficiales* y *Formas Léxicas*

Un *analizador de dos-niveles* constituiría un modelo general con dos elementos básicos: `Sistema Léxico + Reglas morfológicas.` A primera vista, tendría una implementación simple por medio de FST. Sin embargo, hay una variedad de reglas ortográficas o de deletreo, *spelling rules*, y alteraciones fonológicas producidas en el encadenamiento de morfemas que originan que muchas veces no se pueda realizar la correspondencia entre cadenas superficiales y léxicas con este sencillo modelo, como en el caso de `armies` a `army+N+PL`, o `matrices` a `matrix+N+PL`. Además, hay casos en los que la distancia entre ambas formas aumenta y la simple aplicación de reglas paralelas de *dos-niveles* no es suficiente, como en el caso de `mice` a `mouse+N+PL`.

Como se ha puesto de manifiesto, los *analizadores de dos-niveles* no resuelven el problema de las alteraciones fonológicas, o de la aplicación de determinadas reglas ortográficas, entre las formas léxicas y superficiales. Esta limitación es importante porque hay lenguas en las que las formas flexionadas son similares a las formas canónicas, como es el caso del inglés, con un sistema de flexión que presenta pocas irregularidades, frente a un sistema de derivación más complejo; sin embargo hay otras, como el español, en las que ocurre exactamente lo contrario. Para afrontar estos problemas y evitar las ambigüedades en la equiparación sería necesario contar con al menos *tres-niveles* de representación: *léxico*, *intermedio* y *superficial* (Jurafsky y Martín 2000). Pero para ello sería preciso utilizar también más de un transductor: *uno que se interpondría entre los niveles superficial e intermedio y otro entre los niveles intermedio y léxico* (Fig. 4.17).

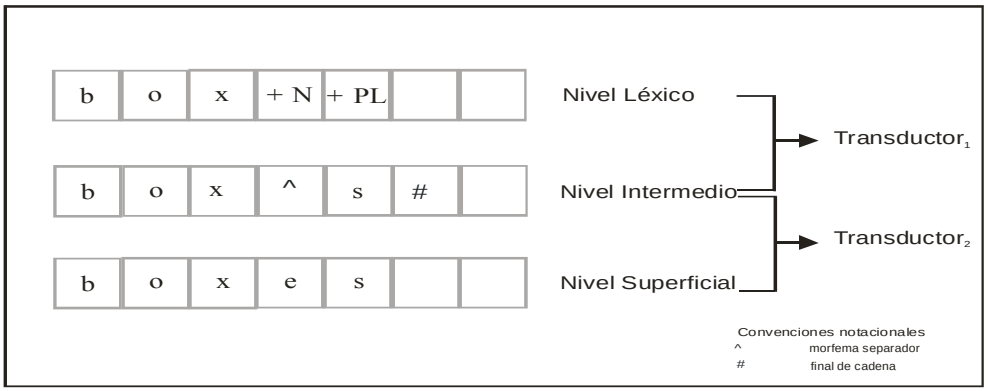


Fig. 4.17: Representación de alteraciones morfológicas en más de dos niveles

Partimos del siguiente planteamiento: *a)* los mecanismos de estado-finito aportan al análisis morfológico el principio general de que la relación entre la *Forma Superficial* de una palabra y su análisis, o *Forma Léxica*, se realiza a través de reglas morfológicas que se codifican en Relaciones Regulares; y *b)* se presenta el problema de que muchas reglas morfológicas y ortográficas necesitan implementarse en analizadores de más de un nivel de representación, lo que conlleva que la equiparación entre cadenas léxicas y cadenas canónicas no se realice por medio de un formalismo tan sencillo.

A continuación, vamos a considerar los distintos procedimientos que se han formulado para solucionar, o atenuar, este problema. En esta línea, aparecen distintas contribuciones metodológicas enfocadas al análisis morfológico con mecanismos de estado-finito, entre las que destacan:

- Composición de series, o *cascadas*, de transductores que representan reglas de re-escritura secuenciales (Kaplan y Kay 1981).
- Formalismos paralelos que representan reglas morfológicas de *dos-niveles* (Koskenniemi 1983; Karttunen 1983; Antworth 1990; Ritchie et al. 1991).
- Modelos en los que las entradas léxicas-canónicas se vinculan a FST, que representan directamente las irregularidades de las formas flexionadas (Silberztein 1999).

Incorporando las aportaciones anteriores, el procedimiento para representar la información en el lexicón que vamos a emplear se basa en el diseño de diccionarios a los que se vinculan FST gráficos. La *hipótesis explicativa* de la que vamos a partir para equiparar *Formas flexionadas* a *Formas canónica* va a consistir en establecer una *Relación Regular* entre *lemas* y *descripción flexional*, que se va a representar directamente en transductores gráfico. El desarrollo de este proceso se realizará en el capítulo siguiente, pero antes es necesario resolver el problema de las *irregularidades* entre *Formas flexionadas* y *Formas canónicas*, para ello vamos a describir cómo se resuelve este problema con otras técnicas de estado-finito, fundamentalmente con el importante modelo conocido como *Morfología de dos-niveles*, y cómo lo vamos a resolver en este trabajo con la aplicación propuesta por Silberztein (Silberztein 1999).

4.4.1. El problema del reconocimiento de Expresiones Léxicas con Técnicas de Estado-Finito

Es necesario aclarar que la morfología de *dos-niveles* se basa en determinar sub-cadenas permitidas de un lenguaje y no representar la totalidad de la *gramática léxica* de ese lenguaje, el desarrollo de los distintos métodos para codificar reglas morfológicas en mecanismos de *dos-niveles* tiene su origen en la importante aportación que supuso el *modelo de fonología de dos-niveles* de Johnson (Johnson 1972). Por esta razón es necesario que hagamos referencia a dicho modelo, como ya hicimos en el Capítulo 2. Este formalismo surgió por la reducción del poder de las reglas transformacionales –aplicadas de forma secuencial creando niveles intermedios entre las formas superficiales y las forma subyacentes, o léxicas–, a simples reglas de *dos-niveles* descriptivas –aplicadas, por el contrario, de forma simultánea o en paralelo entre las formas superficiales y las formas léxicas sin la intervención de niveles intermedios–. Las ventajas de esta importante restricción propició que dichas reglas se pudieran configurar como Relaciones Regulares y , en consecuencia, que fueran susceptibles

de implementarse en FST. En otras palabras, las reglas se aplican de forma *simultánea* o en paralelo, y no de forma *secuencial,* y esto precisamente fue lo que hizo posible que se pudieran describir como Relaciones Regulares.

Sin embargo, aún cuando no se realice la reducción anterior, las reglas de re-escritura, o transformacionales, se podrían representar también por medio de transductores, aunque no sean exactamente reglas de re-escritura generativas en las que los símbolos se transforman en otros y dejan de estar disponibles para la aplicación de la siguiente regla. Esta idea tuvo su principal exponente en Kaplan y Kay (Kaplan y Kay 1981), los cuales proponen una formalismo, extrapolable a distintos niveles de análisis, que ha tenido una gran repercusión en determinadas aplicaciones prácticas: *representar reglas de re-escritura fonológicas por medio de FST que se introducen unos a otros*, lo que habitualmente se conoce como *cascada de transductores*. El interés de este método se basa en la gran avance que supone poder reducir la complejidad de los métodos de reconocimiento y análisis de cadenas cuando es necesario el uso de más de *dos-niveles*.

El método para poder representar reglas morfológicas y alteraciones morfológicas en analizadores léxicos parte de un presupuesto clave que proporciona el formalismo matemático en el que se basan los analizadores de *dos-niveles*: *las Relaciones Regulares se cierran bajo composición* (Kaplan y Kay 1981). Esto es, si tenemos dos reglas que se aplican de forma secuencial por medio de dos transductores que se *alimentan* entre sí, de forma que el *output* del primer transductor sea el *input* del segundo transductor, se puede diseñar un nuevo transductor equivalente por medio de la operación de **composición** (Fig. 4.18). Este transductor simple haría corresponder el *input* del primer transductor con el *output* del segundo sin generar ningún nivel intermedio, y de esta forma se dispone de un mecanismo más simple que maneja sólo dos niveles. Para desarrollar la operación de composición se construye un nuevo transductor, FST_3, con nuevos estados (x, y), de forma que:

- $x \in Q_1$, donde Q_1 es el conjunto de estados del FST_1
- $y \in Q_2$, donde Q_2 es el conjunto de estados del FST_2

- y donde la función de transición de FST_3 se define como:

$$f_3((x_a, y_a),\ o{:}i) = (x_b, y_b) \text{ si}$$

$$\exists\ u \text{ en la función de transición de } FST_1,\ f_1(x_a, o{:}u) = x_b$$

$$\text{y en la función de transición de } FST_2,\ f_2(y_a, u{:}i) = y_b$$

El procedimiento anterior se basa en que entre el nivel subyacente, o léxico, y el nivel superficial, hay niveles intermedios en los que están implicados un sistema de reglas aplicadas de forma secuencial y representadas en *cascada*, o series de transductores conectados verticalmente, que representan cada uno de ellos una regla de forma individual. Los niveles intermedios, así como los símbolos que actúan en ellos, se pueden eliminar por medio de la composición de los distintos transductores en uno nuevo, que operaría sólo en *dos-niveles*. Todo esto conduce a una simplificación en la que la equiparación entre los niveles superficiales y léxicos se contempla como una Relación Regular representada en un único transductor, mucho menos complejo que si se describiera a lo largo de todos los niveles intermedios, y que va a tener una repercusión práctica en la eficacia de los métodos de reconocimiento de patrones. Sin embargo, la composición de sistemas de reglas en un único transductor no es el método más adecuado cuando se aplica a los casos particulares del lenguaje natural debido a que la profundidad de la cascada puede ser de cuatro o cinco niveles y su composición resulta poco práctica para los objetivos del reconocimiento léxico.

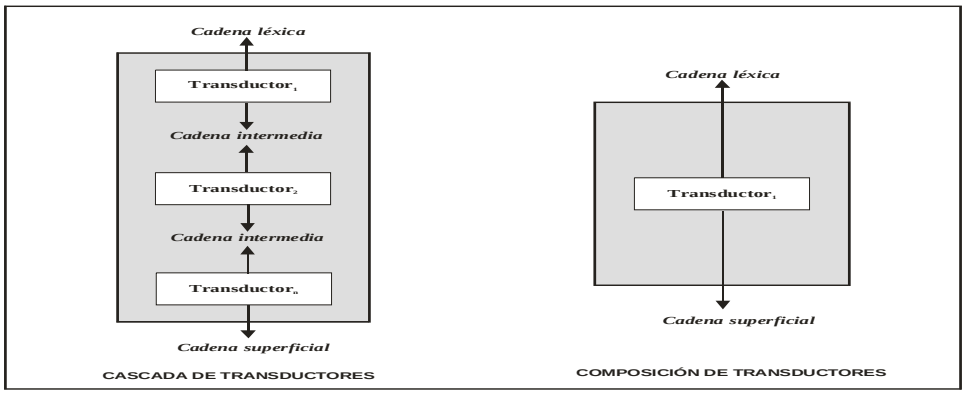


Fig. 4.18: Composición de secuencias de FST (Kaplan y Kay 1981)

Una alternativa a la solución anterior la aporta la morfología de *dos-niveles*, o el modelo de Kimmo Koskenniemi (Koskenniemi 1983), que desde un punto de vista metodológico recibe también la herencia de la *fonología de dos-niveles* de Johnson (Johnson 1972). La propuesta de Koskenniemi se basa en las dos formas tradicionales en las que se concibe el estudio de los morfemas que componen el lenguaje natural:

a) Morfología propiamente dicha que estudia la formación de palabras según la combinación de morfemas, *stems* y afijos.

b) Alteraciones morfológicas –o fonológicas, en el sentido de morfofonémica, o estudio de las alteraciones fonológico/morfológica–, es decir, según las alteraciones de forma que afectan a los morfemas conforme al contexto fonológico en el que ocurre.

Como ya se ha indicado, el modelo de *dos-niveles* de Koskenniemi propone que toda unidad léxica se puede representar como una equiparación entre una cadena léxica y otra superficial:

Forma Léxica: m a t r i x + λ + s
Forma Superficial: m a t r i c e s

y que la mencionada correspondencia se produce por la acción de conjuntos de reglas ordenadas que se definen en términos de Relaciones Regulares. Cada regla se vincula a un transductor que codifican alguna limitación en la equiparación –una regla vinculada a un transductor se encarga de transformar $c:x$ ó $c \rightarrow x$ en determinado contexto fonológico y otra regla se dedica a añadir o eliminar una e cuando ésta aparece detrás de c y delante de s como es el caso del ejemplo anterior–. De esta forma, un conjunto de transductores (Fig. 4.19) se encarga de unir cadenas superficiales a cadenas léxicas, aceptando cada uno de ellos al mismo tiempo el par *Forma Superficial-Forma Léxica*. Por otra parte, como se puede comprobar, ni en este modelo ni en el que proponen Kaplan y Kay se incluyen las categorías morfológicas como parte de la *Forma Léxica*.

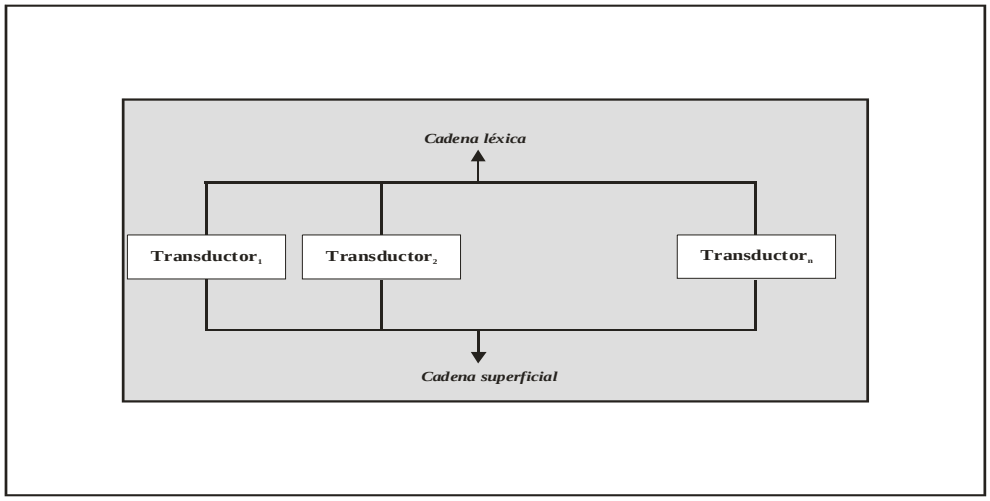


Fig. 4.19: Construcción de FST en paralelo (Koskenniemi 1983)

Lo que distingue a este formalismo del modelo de Kaplan y Kay es que la forma de descomponer estas reglas no se representa como una serie de transductores, que operan de forma secuencial –en el que el *output* de uno es el *input* de otro–, sino por medio de reglas, que limitan o restringen la equiparación entre formas superficiales y léxicas. En otra palabras, las reglas contienen información parcial sobre algún aspecto de la equiparación y se codifican cada una de ellas en un FST. A su vez, el conjunto de transductores, en los que se representan las distintas limitaciones o restricciones en la equiparación, actúan juntos de forma **paralela** (Karttunen 1991). La conexión de los transductores hace que todos acepten a la vez cualquier par de caracteres y, en consecuencia, actúan simultáneamente en el proceso de reconocimiento

Teniendo en cuenta los modelos anteriores, Karttunen, Kaplan y Zaenen (Kattunen *et al.* 1992) desarrollan un transductor léxico que equipara directamente *Formas Superficiales* a *Formas Léxicas*, y viceversa. Sin embargo, tanto las alteraciones regulares e irregulares que se pueden producir entre ambas formas, así como las modificaciones que puedan desencadenar los afijos flexivos y derivativos hacen que el establecimiento de la equiparación sea una operación compleja. Para afrontar este problema el transductor léxico se basa en la

composición de reglas morfológicas de *dos-niveles*. En este formalismo se tiene en cuenta los mecanismos de formación de palabras y retoma dos de las propuestas anteriores: el analizador morfológico de *dos-niveles* de Koskenniemi y la composición de *cascadas* de transductores de Kaplan y Kay. De forma sistemática, el transductor léxico se construye con dos componentes analíticos básicos, tal y como se expone en el trabajo de Karttunen, Kaplan y Zaenen (Karttunen *et al*. 1992):

- Un *lexicón fuente, de estado-finito*, o componente léxico compilado en un simple autómata o un transductor, que define el conjunto de *Formas Léxicas* válidas del lenguaje.

- Un conjunto de reglas de estado-finito, o componente de reglas compiladas en transductores, que se encargan de asignar *Formas Léxicas* a todas las realizaciones de superficie, y viceversa. Por tanto, las reglas establecen las condiciones para que la información depositada en el diccionario de formas canónicas, y asociada a la entrada léxica se aplique con éxito. Las reglas se compilan en FST y se unen con el lexicón por medio de las operaciones matemáticas de *intersección* y *composición* (Fig. 4.20).

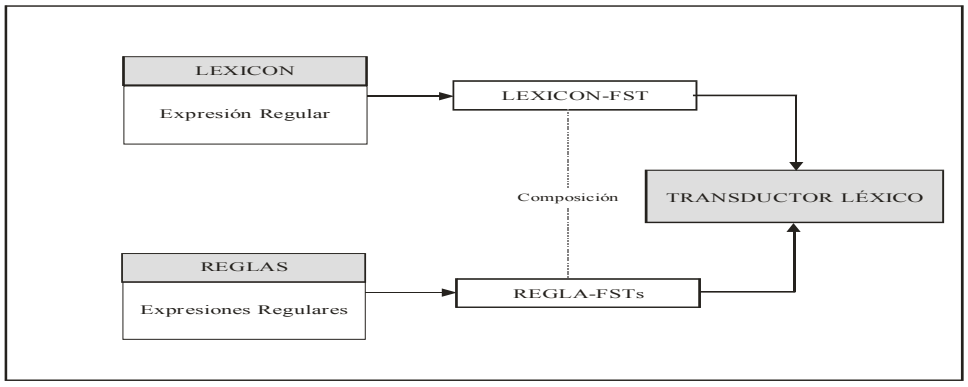


Fig. 4.20: Composición del lexicón-y de las regla-FST (Karttunen *et al.* 1992)

Para realizar el proceso anterior, el transductor léxico se apoya en el modelo de *dos-niveles* en el sentido de que cada palabra se representa como una equiparación entre cadenas léxicas y

cadenas superficiales a través de los *paths* que contenga el transductor. Las diferencias entre ambas formas se describen por medio de reglas de *dos-niveles*: la entrada correspondería a la palabra que se pretende analizar, *Forma Superficial*, y la salida correspondería a la representación léxica canónica, *Forma Léxica*, según la información aportada por el lexicón. La representación del nivel intermedio, *Forma Intermedia*, desaparece en un proceso en el que se combina la intersección entre los FST que representan alteraciones fonológicas, o reglas ortográficas, y la composición del lexicón con los transductores de reglas. La metodología que formaliza el proceso de construcción del transductor léxico es la siguiente:

- Se desarrolla un lexicón y se compila en un FST (Fig. 4.21) en el que se incluye: secuencias de morfemas en su representación canónica y categorías morfológicas.

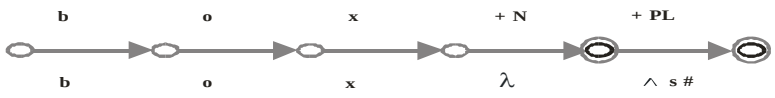


Fig. 4. 21: Representación canónica de morfemas

La función de transición del lexicón-FST sería la siguiente:

$f(q_0, b{:}b) = q_1$
$f(q_1, o{:}o) = q_2$
$f(q_2, x{:}x) = q_3$
$f(q_3, +N{:}\lambda) = q_4$
$f(q_4, +PL{:}\wedge\ s\ \#) = q_5$

- Se averiguan las reglas y se compilan cada una de ellas en un FST (Fig. 4.22) en el que se incluye conjuntos de reglas de dos tipos morfológicas y léxicas. Las reglas morfológicas tienen como función determinar cómo se combinan los morfemas; las reglas léxicas se encargan de establecer cómo se derivan unos *stems* de otros. En este caso, la regla se define como: «*añadir, o eliminar, una* `e` *cuando se encuentra en el contexto: detrás de* `x` *y delante de* `s`*, cuando sea final de palabra*».

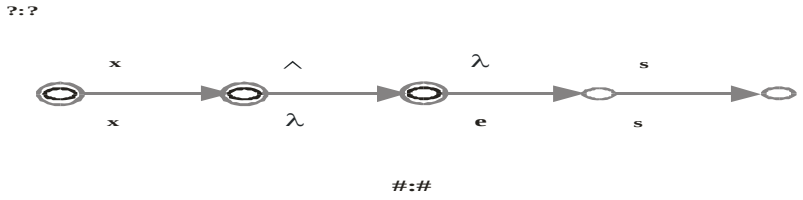


Fig. 4.22:Representación de reglas morfológicas

La función de transición de la Regla-FST es la siguiente:

$f(q_0,\ x{:}x) = q_1$
$f(q_1,\ \wedge{:}\lambda) = q_2$
$f(q_2,\ \lambda : e) = q_3$
$f(q_3,\ s{:}s) = q_4$
$f(q_4,\ \#{:}\#) = q_0$

- Se realiza la composición del lexicón-FST y de la Regla-FST de la forma siguiente:

$f(q_4, +\text{PL}{:}\ \wedge\ s\ \#) = q_5$ .o. $f(q_1,\ \wedge :\ \lambda) = q_2$

$f((q_4,q_1), +\text{PL}{:}\lambda\ \ s\ \#) = (q_5,q_2)$

$f((q_4,q_1), +\text{PL} : \lambda\ \ s\ \#) = (q_5,q_2)$ .o. $f(q_2,\ \lambda :\ e) = q_3$

$f((q_4,q_1), +\text{PL:e}\ s\ \#\ ) = ((q_5,q_2),q_3)$

/ ... /

y de la que se obtiene un FST simple (Fig. 4.23):

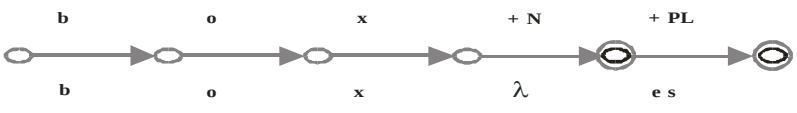


Fig. 4.23: Composición del lexicón y las reglas morfológicas

La función de transición del nuevo FST es:

$$f(q_0,\ b{:}b) = q_1$$
$$f(q_1,\ o{:}o) = q_2$$
$$f(q_2,\ x{:}x) = q_3$$
$$f(q_3,\ +N{:}\lambda) = q_4$$
$$f(q_4,\ +PL{:}es) = q_5$$

Las fases de análisis del transductor léxico se sintetizan en:

- En la primera etapa, se ordenan los sistemas de reglas paralelas de *dos-niveles* en una *cascada*.
- En la segunda etapa, se efectúa la *intersección* de las reglas de cada nivel que dan lugar a un transductor simple FST_A.
- En la tercera etapa, se práctica la *composición* del lexicón-FST y del transductor de reglas FST_A.
- En la cuarta etapa, se presenta el resultado final consistente en un único FST que se encarga de equiparar formas canónica y categorías sintácticas, o gramaticales, con las correspondientes formas superficiales, y viceversa (Fig. 4.24).

Fig. 4.24: Operaciones de intersección y composición en un FST (Karttunen *et al*. 1992)

El transductor léxico propuesto por Karttunen, Kaplan y Zaenen, basado en las propiedades matemáticas de las reglas de re-escritura, *intersección* y *composición*, aporta un valioso

formalismo fundamentalmente porque algunas de sus propuestas se pueden extrapolar a cualquier proceso reconocimiento de expresiones. Sus autores sintetizan las importantes contribuciones metodológicas que aporta su modelo (Karttunen *et al*. 1992):

1. Permite componer cualquier descripción de `n-niveles` en una simple descripción de `2-niveles`.
2. La separación inicial entre las entradas léxicas del lexicón y las reglas resulta un planteamiento eficaz en la construcción de los sistemas de reconocimiento léxico porque las reglas descomponen una equiparación muy compleja entre *Formas Léxicas* y *Formas Superficiales* en un *conjunto de relaciones más simples* que se pueden manipular computacionalmente.
3. La función de las reglas es ampliar el lexicón de `2-niveles`, en el sentido de que se construyen junto a él pero no como parte de él. En consecuencia, las reglas no intervienen en todos los proceso de reconocimiento sino *sólo cuando sean necesarias* porque la equiparación entre las formas canónicas y las formas superficiales sea opaca, o no se muestre de forma clara.
4. Las operaciones de intersección y composición reducen la complejidad de la interacción entre el lexicón y las reglas, el resultado final es la construcción de un FST simple donde la separación entre ambos componentes desaparece.

Este modelo confirmaría la necesidad de elaborar listas parciales en los procesos implicados en el reconocimiento léxico. Una lista general se encargaría de organizar el lexicón –construido a partir de distintas operaciones, como *concatenación*, *unión*, o *clausura de Kleene*, semejante a los Lenguajes Regulares y Relaciones Regulares– y un conjunto de reglas formalizarían las alteraciones morfológicas cuando la correspondencia entre *Formas Léxicas* y *Formas Superficiales* no se establezca de forma clara en un modelo simple de *dos-niveles*. Con este planteamiento, el análisis léxico con técnicas de Estado-Finito nos proporcionará el patrón léxico –el *stem* y todos los afijos junto con las etiquetas que indican las características *part-of-speech*, como la persona, el número, el modo o el aspecto– de todas

las palabras que puedan aparecer en los textos, esto es, la información necesaria para el procesamiento sintáctico posterior.

El analizador *Xerox Finite-State Morphological Tools* se basa en el formalismo propuesto por Karttunen, Kaplan y Zaenen (Karttunen *et al.* 1992) y actualmente se aplica con éxito al inglés, francés, alemán, español, portugués, holandés e italiano. El resultado del análisis morfológico con esta herramienta ofrecería un resultado parecido al siguiente:

```
day     day+Noun+Sg
days    day+Noun+Pl

army    army+Noun+Sg
armies  army+Noun+Pl

leaf    leave+Verb+Pres+Non3sg
leaf    leaf+Verb+Pres+Non3sg
leaf    leaf+Noun+Sg

leaves  leave+Verb+Pres+3sg
leaves  leave+Noun+Pl
leaves  leaf+Verb+Pres+3sg
leaves  leaf+Noun+Pl

matrix  matrix+Noun+Sg
matrices        matrix+Noun+Pl

foot    foot+Verb+Pres+Non3sg
foot    foot+Noun+Sg
feet    foot+Noun+Pl

basis   basis+Noun+Sg
bases   basis+Noun+Pl
bases   base+Noun+Pl
bases   base+Verb+Pres+3sg
/../
```

Teniendo en cuenta los modelos anteriores, nuestro objetivo se centrará en el desarrollo de herramientas de análisis léxico según la aplicación propuesta por Silberztein (Silberztein 1999), que ofrece una solución más sencilla, porque no interviene el componente de reglas morfológicas sino que las irregularidades se representan directamente en transductores gráficos. La aplicación desarrollada por Silberztein para el análisis léxico consiste esencialmente en un diccionario compuesto de f*ormas canónicas* y *códigos morfo-sintácticos* que indican la categoría POS a la que pertenece cada entrada del diccionario. A su vez, cada código se vincula a un FST gráfico compuesto de un *nodo inicial* y un *nodo final* que describe el trayecto que debe seguir el analizador morfológico (Fig. 4.25). Mediante el modelo anterior las irregularidades se representan directamente en transductores.

Para poder obtener la flexión de las formas canónicas cuando existen irregularidades se utiliza un sencillo mecanismo de eliminación de caracteres, denominado `operador de borrado L`, de este modo no sería preciso la intervención de las *reglas morfológicas*. Es preciso anotar que el uso del operador `L` da lugar a que se elimine un solo carácter, pero si fuera necesario eliminar más de un carácter se indicaría simplemente con el número de caracteres a eliminar.

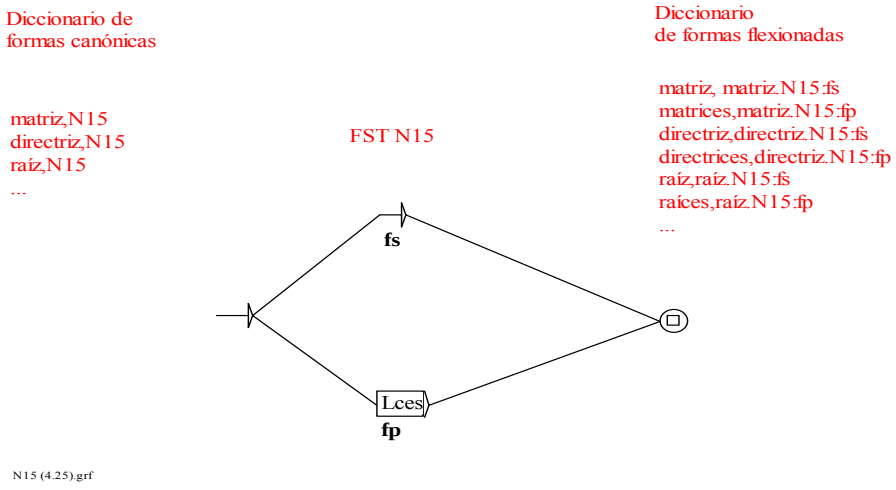


Fig. 4.25: Representación de irregularidades morfológicas en FST

Consideramos que esta representación es más eficaz para los objetivos prácticos de la generación de índices, porque no sólo nos permitirá crear de forma natural diccionarios especializados de formas canónicas a los que se vinculan FST, sino que nos permitirá identificar las irregularidades entre *Formas Superficiales* y *Formas Léxicas* con procedimientos más sencillos y eficaces sin la mediación y compleja representación de las reglas morfológicas. El desarrollo de este procedimiento para la construcción de los analizadores léxicos se realizará en el Capítulo 5.

4.5. Metodología para la Representación de Expresiones Sintácticas con Técnicas de Estado-Finito

Los analizadores sintácticos son *aceptadores* o *reconocedores* de las estructuras sintácticas de un lenguaje. Para realizar este proceso los analizadores, o *parser*, infieren la estructura de las cadenas de palabras, a partir del conocimiento almacenado en *lexicones computacionales* y *gramáticas electrónica*s, y resuelven si las mencionadas cadenas se pueden derivar de las gramáticas, es decir: *si son gramaticales*, o *no gramaticales.* Es necesario aclarar que, en este proceso, las entradas de los lexicones son los elementos terminales de las gramáticas, y por lo tanto dependen de ellas. El resultado del análisis léxico únicamente sería el apoyo de las gramáticas, y las unidades analizadas se insertaría a las reglas de producción. Teniendo en cuenta la aclaración anterior, en el análisis de estructuras lingüísticas es preciso definir una *gramática electrónica*, o conjunto de reglas de producción que sean capaces de representar las estructuras sintácticas de las sentencias de ese lenguaje. Una vez que se haya obtenido la gramática, ésta se trasladará a los distintos tipos de analizadores o mecanismos de

reconocimiento de estructuras sintácticas, en este caso el *parser* es un Autómata de Estado-Finito (AFD).

Dependiendo del lenguaje que el formalismo sintáctico pueda generar, las gramáticas se clasifican como más o menos *expresivas*, y aunque las *Gramáticas Regulares* constituyan un formalismo poco expresivo para la equiparación o identificación de estructuras sintácticas, son adecuadas para describir las estructuras de los *Noun Phrases* (NP), o *Sintagmas Nominales* (SN). Un NP es una estructura sintáctica con diversos grados de complejidad que se compone básicamente de un *núcleo* formado por un *nombre* junto a una serie de elementos opcionales –como determinantes o cuantificadores– y *modificadores* –como adjetivos, adverbios, participios, sintagmas u oraciones– que se vinculan al nombre para especificar diversas propiedades.

Para que pueda funcionar el componente sintáctico es necesario que se haya efectuado previamente un análisis morfológico, en el que se hayan identificado las categorías gramaticales, etiquetas POS, de las unidades léxicas. Esta exigencia se debe a que las entradas al analizador sintáctico son etiquetas léxico-gramaticales no ambiguas. Una vez etiquetadas las unidades léxicas, si se quisiera construir un analizador sintáctico, o AFD, que reconociera NP aplicaríamos la metodología clásica utilizada en la *Teoría de Estado-Finito*, expuesta en un epígrafe precedente, consistente en el desarrollo de los siguientes procesos:

1. *Describir las estructuras de los NNPP* por medio de Expresiones Regulares.
2. *Derivar las Expresiones Regulares*.
3. *Construir las Gramática Regular* a partir de las derivaciones de las Expresiones Regulares.
4. *Trasladar las Gramáticas Regulares* a Autómatas de Estado-Finito Gráficos.
5. *Transformar los Autómatas* en sus equivalentes deterministas.
6. *Minimizar los Autómatas*.
7. *Obtener los Autómatas* que se encarguen de reconocer los SSNN especificados.

En el caso de que se quisieran reconocer los siguientes NNPP:

a) Nombres, N, seguidos opcionalmente de Adjetivos y Nombres, $(A + N)^*$, en este caso se considera además que un Participio, P, pueda hacer la función de Adjetivo, $(P + N)^*$.

b) Determinante, DET, seguido de un número indeterminado de Nombres, N.

El procedimiento para reconocer las estructuras de los NNPP anteriores, $NP \rightarrow N\ (A + N)^* + DET\ N^*$, partiría de su *especificación* en términos de Expresiones Regulares:

$$ER_0 = N\ (A + N)^* + DET\ N^*$$

equivalente a :

$$ER_0 = N\ (P + N)^* + DET\ N^*$$

La fase siguiente consistiría en calcular las *derivadas* de la Expresión Regular (ER_0), esto es, el cálculo del conjunto de cadenas que comienzan por el símbolo respecto del que se deriva. Así, partiendo de la expresión:

$$ER_0 = N\ (A + N)^* + DET\ N^*$$

se obtendrían las siguientes derivadas:

$$D_N(ER_0)$$

$$D_N\left(N\ (A + N)^* + DET\ N^*\right) =$$

$$\left[D_N(N)\ (A + N)^* + \alpha(N)\ D_N(A + N)^*\right] + \left[D_N(DET)\ N^* + \alpha(DET)\ D_N\left(N^*\right)\right] =$$

$$\left[\lambda(A + N)^* + \varnothing\ D_N(A + N)^*\right] + \left[\varnothing\ N^* + \varnothing\ D_N\left(N^*\right)\right] =$$

$$\lambda\ (A + N)^* = ER_1$$

$$D_A(ER_0)$$
$$D_A\left(N\ (A + N)^* + DET\ N^*\right) =$$
$$\left[D_A(N)\ (A + N)^* + \alpha(N)\ D_A(A + N)^*\right] + \left[D_A(DET)\ N^* + \alpha(DET)\ D_A\left(N^*\right)\right] =$$
$$\left[\varnothing(A + N)^* + \varnothing\ D_A(A + N)^*\right] + \left[\varnothing\ N^* + \varnothing\ D_A\left(N^*\right)\right] = \varnothing$$

$$D_{DET}(ER_0)$$
$$D_{DET}\left(N\ (A + N)^* + DET\ N^*\right) =$$
$$\left[D_{DET}(N)\ (A + N)^* + \alpha(N)\ D_{DET}(A + N)^*\right] + \left[D_{DET}(DET)\ N^* + \alpha(DET)\ D_{DET}\left(N^*\right)\right] =$$
$$\left[\varnothing(A + N)^* + \varnothing\ D_{DET}(A + N)^*\right] + \left[\lambda\ N^* + \varnothing\ D_{DET}\left(N^*\right)\right] =$$
$$\lambda\ N^* = N^* = ER_2$$

$$D_N(ER_1)$$
$$D_N\left((A + N)^*\right) =$$
$$D_N(A + N)\ (A + N)^* =$$
$$\left[\ D_N(A) + D_N(N)\ \right](A + N)^* =$$
$$\left[\varnothing + \lambda\right](A + N)^* =$$
$$\lambda\ (A + N)^* = ER_1$$

$$D_A(ER_1)$$
$$D_A\left((A + N)^*\right) =$$
$$D_A(A + N)\ (A + N)^* =$$
$$\left[\ D_A(A) + D_A(N)\ \right](A + N)^* =$$
$$\left[\lambda + \varnothing\right](A + N)^* =$$
$$\lambda\ (A + N)^* = ER_1$$

$$D_{DET}(ER_1)$$
$$D_{DET}\left((A + N)^*\right) =$$
$$D_{DET}(A + N)\ (A + N)^* =$$
$$\left[\ D_{DET}(A) + D_{DET}(N)\ \right](A + N)^* =$$
$$\left[\varnothing + \varnothing\right](A + N)^* = \varnothing$$

$$D_N(ER_2)$$
$$D_N\left(N^*\right) =$$
$$D_N(N)\ N^* =$$
$$\lambda\ N^* = ER_2$$

$$D_A(ER_2)$$
$$D_A\left(N^*\right) =$$
$$D_A(N)\ N^* =$$
$$\varnothing\ N^* = \varnothing$$

$D_{DET}(ER_2)$
$D_{DET}(N^*) =$
$D_{DET}(N)\ N^* =$
$\varnothing\ N^* = \varnothing$

La *Gramática Regular Lineal por la Derecha* que reconoce la expresión anterior se define como:

$G = (\{N, A, DET\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

en la que las reglas de producción, P, se obtienen aplicando las siguientes reglas:

- Si $D_a(ER_i) = ER_j$ y $ER_j \neq \lambda$, $ER_j \neq \varnothing$, se crea una regla $ER_i ::= aER_j$
- Si $D_a(ER_i) = \varnothing$, no se crea ninguna regla
- Si $\lambda \in D_a(ER_i)$, se crea una regla $ER_i ::= a$
- Si $\lambda \in ER_0$, se crea una regla $ER_0 ::= \lambda$

De este modo, las reglas de producción, P, correspondientes a las derivadas anteriores serían:

$ER_0 ::= N\ ER_1 \mid DET\ ER_2 \mid N \mid DET$
$ER_1 ::= N\ ER_1 \mid A\ ER_1 \mid N \mid A$
$ER_2 ::= N\ ER_2 \mid N$

A partir de la Gramática Regular anterior se obtendría el AFD que reconoce el lenguaje que dicha gramática genera, definido según el procedimiento anteriormente descrito como:

$AF = (\{N, A, DET\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, \{F\})$

donde la función de transición, f, se obtiene aplicando las siguientes reglas:

- Si $D_a(ER_i) = ER_j$ y $ER_j \neq \lambda$, $ER_j \neq \varnothing$, entonces $ER_j \in f(ER_i, a)$

- Si $D_a(ER_i) = \varnothing$, entonces $f(ER_i, a) = \varnothing$
- Si $\lambda \in D_a(ER_i)$, entonces $F \in f(ER_i, a)$
- Si $\lambda \in ER_0$, entonces $F \in f(ER_0, \lambda)$

El Autómata Finito se puede representar en la siguiente tabla de transiciones (Fig. 4.26):

f	N	A	D E T
→ E R 0	E R 1, F	∅	E R 2, F
E R 1	E R 1, F	E R 1, F	∅
E R 2	E R 2, F	∅	∅
* F	∅	∅	∅
∅	∅	∅	∅

Fig. 4.26: Tabla de transiciones del AFD que reconoce ER_0

A continuación, se puede transformar en un *Autómata Finito Determinista Mínimo* para que el reconocimiento de la expresión sea más eficaz y, por último, se renombran los estados (Fig. 4.27). Del mismo modo, se puede obtener su representación en un diagrama de transiciones (Fig. 4.28)

f	N	A	D E T
→ q 0	q 1	∅	q 2
* q 1	q 1	q 1	∅
* q 2	q 2	∅	∅
∅	∅	∅	∅

Fig. 4.27: AFD Minimizado que reconoce la expresión ER_0

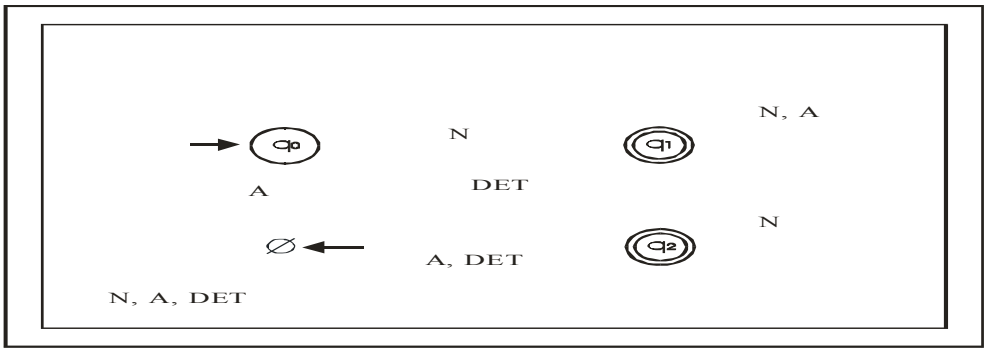


Fig. 4.28: Diagrama de transiciones que reconoce la expresión ER_0

Básicamente, el proceso que se sigue para la equiparación de patrones sintácticos con técnicas de estado-finito consistiría en: calcular las derivadas de una Expresión Regular, y generar a continuación el autómata que reconoce el lenguaje que representa dicha expresión. El AFD resultante de la expresión $ER_0 = N\ (A + N)^* + DET\ N^*$ sería capaz de detectar construcciones nominales como:

```
Computer assisted indexing
Value added networks
Value added tax
Library
Collection development policies
/.../
```

A su vez, incorporando la aplicación informática desarrollada por Silberztein podemos crear además AFD gráficos (Fig. 4.29), equivalentes al anterior, por medio de un editor gráfico *FSGraph* (Silberztein 1996). En la representación obtenida con *FSGraph* los estados del autómatas permanecen ocultos y sólo se muestra la función de transición entre estados por medio de las etiquetas correspondientes a las distintas categorías sintácticas, así como la etiqueta < E > que representa la *cadena vacía*.

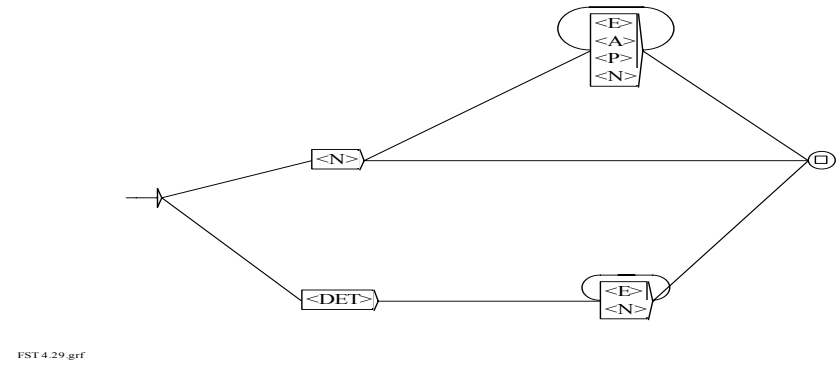
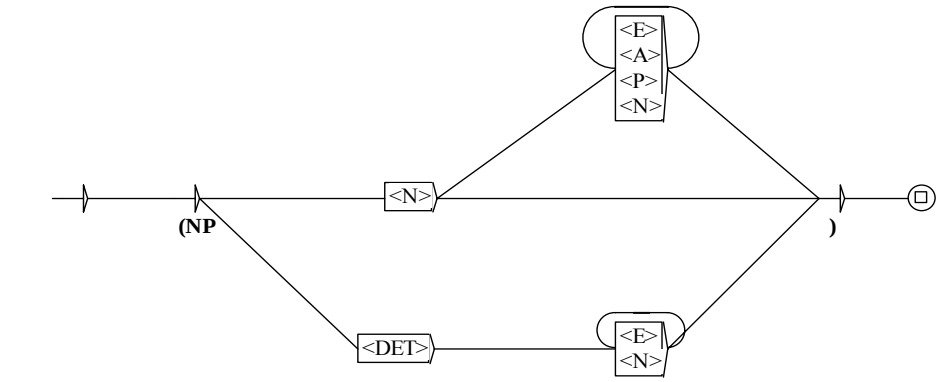


Fig. 4.29: Gramática representada en un AFD gráfico

Por otra parte, al igual que en los analizadores léxicos, la función de los analizadores sintácticos no es sólo identificar determinada construcciones sintácticas sino ofrecer una *representación estructurada* de las construcciones identificadas. Con este objetivo, las gramáticas electrónicas se pueden configurar como *transductores* cuyas entradas son etiquetas sintácticas y cuya salidas son marcas, en forma de *paréntesis etiquetados* (Fig. 4.30). El análisis sintáctico con este mecanismo ofrecería como resultado la etiquetación en el texto de las secuencias lingüísticas reconocidas, o la representación de las estructuras de esas secuencias en forma de *diagramas* o *árboles ramificados* (Fig. 4.31).



Representación de las estructuras sintácticas de los NPs reconocidos

(NP computer assisted indexing)
(NP value added tax)
(NP value added networks)
(NP collection development policies)

FST 4.30.grf

Fig. 4.30: Gramática representada en un FST gráfico

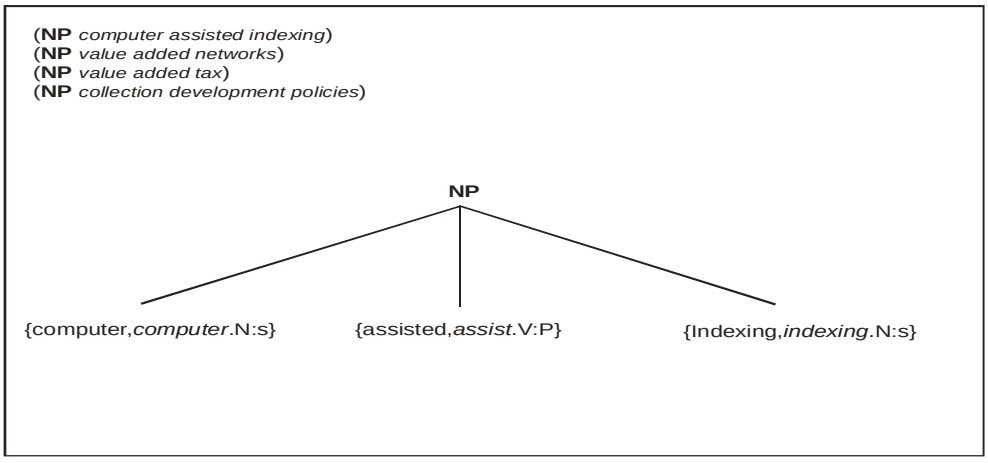


Fig. 4.31: Representación de la estructura sintáctica de un NP en forma de *árbol de derivación*

Con este método se podrían reconocer otro tipo de NNPP que se especifican a modo de ejemplo en las siguientes Expresiones Regulares:

- NNPP que estén encabezados por un número indefinido de A (*Adjetivos*), o la cadena vacía, y que acaben en un número indefinido de N (*Nombres*). La Expresión Regular que define este tipo de NP sería la siguiente:

$$NP1 \rightarrow A^{*}\ N^{+}$$

$$ER_0 = A^{*}\ N^{+}$$

- NNPP que estén introducidos por un N (*Nombre*), seguidos por una PREP (*Preposición*) y que acaben en un número indefinido de N (*Nombres*), o la cadena vacía. Además, la PREP puede estar seguida opcionalmente por un número indefinido de A (*Adjetivos*), o la cadena vacía. La Expresión Regular que describe este tipo de NP sería la siguiente:

$$NP2 \rightarrow N\ PREP\ (A)^{*}\ N^{*}$$

$$ER_0 = N\ PREP\ (A)^{*}\ N^{*}$$

- NNPP que comiencen por un DET (*Determinante*) seguidos por un N (*Nombre*) y que acaben en un A (*Adjetivo*). Además, los sintagmas nominales introducidos por DET pueden estar seguidos opcionalmente por la combinación $(N\ A)^{*}$, repetida un número indefinido de veces, o ninguna. La Expresión Regular que representa este tipo de NP sería la siguiente:

$$NP3 \rightarrow DET\ (N\ A)^{*}\ N\ A$$

$$ER_0 = DET\ (N\ A)^{*}\ N\ A$$

- NNPP que estén introducidos por un N (*Nombre*), repetido un número indefinido de veces, y que acaben en un P (*Participio*). Además, pueden estar seguidos opcionalmente por la combinación $\left(PREP^{*}\ N\right)^{*}$, repetida un número indefinido de

veces, o ninguna. La Expresión Regular que describe este tipo de NP sería la siguiente:

$$NP4 \rightarrow N^{*}\ PA\ \left(PREP^{*}\ N\right)^{*}$$

$$ER_{0} = N^{*}\ PA\ \left(PREP^{*}\ N\right)^{*}$$

- NNPP que estén encabezados por un DET (*Determinante*) seguidos por un número indeterminado de N (*Nombres*), o la cadena vacía, y acaben en un A (*Adjetivo*). La Expresión Regular que representa este tipo de NP sería la siguiente:

$$NP5 \rightarrow DET\ N^{*}\ A$$

$$ER_{0} = DET\ N^{*}\ A$$

A partir de las Expresiones Regulares anteriores se podrían obtener los Autómatas de Estado-Finito que serían capaces de reconocer NNPP como:

```
Automatic text analysis
Access to information
College of advanced education libraries
International Pharmaceutical Abstracts
Integrated Academic Information Management Systems
Association of Assistant Librarians UK
Association of American Library Schools Indexing
Association of Reseach Libraries USA
/.../
```

Sin embargo, el proceso anterior fracasa si una misma unidad léxica tiene asignada más de una categoría gramatical, debido a que las entradas al analizador sintáctico no pueden ser ambiguas. Este problema tiene su origen en el hecho de que en el *diccionario*, o *lexicón*, es frecuente que una misma unidad léxica tenga asignada más de una categoría gramatical, y

precisamente esta información lingüística es la que se traslada, en el proceso de asignación de etiquetas, a las secuencias léxicas. La solución a este problema está en el desarrollo de herramientas de *desambiguación de etiquetas POS* y, aunque pueda parecer que se trata de un problema léxico, afecta fundamentalmente al análisis sintáctico, por esta razón se va a plantear en el siguiente apartado. En consecuencia, antes de realizar el análisis, o *parsing*, sintáctico es preciso resolver el problema de la ambigüedad en la asignación de etiquetas gramaticales.

4.5.1. El problema del reconocimiento de Expresiones Sintácticas con Técnicas de Estado-Finito

La mayor complicación del reconocimiento de las estructuras sintácticas es el de la ambigüedad en la etiquetación, para solucionar este problema existen diversos métodos que difieren en la técnica utilizada, aunque la mayoría tienen en cuenta el *contexto* de aparición de las unidades lingüísticas. Básicamente existen dos procedimientos para solucionar la ambigüedad con técnicas de estado-finito:

1. *Métodos estadísticos*, utilizados por los *etiquetadores estocásticos*
2. *Métodos simbólicos*, utilizados por los *etiquetadores sintácticos.*

Dentro de los métodos estadísticos, los formalismos de estado-finito que se pueden utilizar para resolver el problema de la ambigüedad en la asignación de etiquetas encontramos los etiquetadores estocásticos, que se basan en el cálculo de:

- La probabilidad de que se asigne una etiqueta a un determinado término.
- La probabilidad de que aparezca una determinada secuencia de categorías.

Para la obtención de las probabilidades anteriores, los etiquetadores estocásticos necesitan partir de un *corpus de entrenamiento anotado con etiquetas POS* –por esta razón la mayoría de los etiquetadores que utilizan métodos estadísticos necesitan previamente la información aportada por un diccionario en formato electrónico–. A partir de esta información, los etiquetadores estocásticos se pueden *entrenar* para calcular la probabilidad de frecuencia de un término dada una etiqueta. Con este objetivo, los etiquetadores obtienen la denominada *matriz de probabilidad de observación*, representada con la fórmula siguiente:

$$P(w_i/t_i) = \frac{f(w_i, t_i)}{f(t_i)}$$

donde

- $f(w_i, t_i)$ es la probabilidad de frecuencia de la palabra w_i con la etiqueta t_i
- $f(t_i)$ es la probabilidad de frecuencia de palabras con la etiqueta t_i

Pero el problema está en cómo se puede adquirir la probabilidad de que dos unidades co-ocurran, es decir, cómo se podría obtener (w_i/t_i) , cuando a un término se le puede asignar más de una etiqueta. En relación con esta cuestión, a la probabilidad de que un término y una etiqueta, o dos etiquetas, *co-ocurran* se le denomina comúnmente *probabilidad condicionada*.

Así, dados dos sucesos M y N, la probabilidad del suceso M está condicionada por N y se denota por $P(M/N)$. Si la probabilidad del suceso N es distinta de *cero*, la *probabilidad condicionada* se expresa con la fórmula siguiente:

$$P(M/N) = \frac{P(M \cap N)}{P(N)}$$

donde

- $P(M \cap N)$ es la probabilidad conjunta de que los dos sucesos ocurran simultáneamente.
- $P(N)$ es la probabilidad de que el suceso (N) ocurra.

- y en el caso de que los sucesos (M) y (N) sean independientes, se cumple $P(M/N) = P(M)$ porque $P(M \cap N) = P(M)P(N)$

Una de las formas de calcular la *probabilidad condicionada* es por medio de los denominados *modelos n-gramas*, donde la `n` representa el número de unidades o diagramas que se tienen en cuenta –siendo `n = 2` un *bigrama*, `n = 3` un *trigrama*, o `n = 4` un *tetragrama*-. Para calcular la probabilidad condicionada de una unidad lingüística dada otra unidad anterior se utiliza, en el caso de un *bigrama*, la siguiente fórmula:

$$P(U_i/U_{i-1}) = \frac{f(U_{i-1}, U_i)}{f(U_{i-1})}$$

o, la siguiente fórmula si se tratara de un *trigrama*:

$$P(U_i/U_{i-2}, U_{i-1}) = \frac{f(U_{i-2}, U_{i-1}, U_i)}{f(U_{i-2}, U_{i-1})}$$

Una vez calculada la probabilidad condicionada se *entrenaría* al etiquetador para que pudiera determinar la probabilidad de la unidad U_i condicionada por la unidad precedente U_{i-1}. A su vez, para realizar el cálculo estadístico de los *bigramas* (U_{i-1}, U_i), o *trigamas* (U_{i-2}, U_{i-1}, U_i), sería necesario reconocer y contar en el *corpus de entrenamiento* cada vez que co-ocurren las dos, o las tres unidades.

Si incorporamos la probabilidad condicionada anterior a un Autómata Probabilístico obtendríamos un *Modelo de Markov*, en el que la *matriz de probabilidad de transición* se definiría por la aplicando de la siguiente fórmula:

$$P(t_i/t_{i-1}) = \frac{f(t_{i-1}, t_i)}{f(t_{i-1})}$$

donde

- $f(t_{i-1}, t_i)$ es la frecuencia de ocurrencia de las etiquetas t_{i-1} y t_i
- $f(t_{i-1})$ es la frecuencia de ocurrencia de la etiqueta t_{i-1}

A su vez, si incorporamos la probabilidad de observación a un *Transductor Probabilístico*, obtendríamos un etiquetador estocástico, o *Modelo Oculto de Markov* (HMM). La técnica en la que se basan los estiquetadores estocásticos para la desambiguación de etiquetas es tomar la probabilidad condicionada de que dos unidades co-ocurran, $P(w_i/t_i)$, y la probabilidad condicionada por el *contexto* de que una unidad siga a otra, $P(t_i/t_{i-1})$. El resultado de los etiquetadores estocásticos se configuraría como la multiplican de los datos de la *matriz de probabilidad de observación* por los datos de la *matriz de probabilidad de transición*, según la fórmula siguiente:

$$\prod_{i}^{n} P(w_i/t_i) P(t_i/t_{i-1})$$

Sin embargo, en un HMM no se puede conocer la secuencia de estados, etiquetas, por las que el modelo transita porque permanece *oculta*. Para conocer la secuencia de etiquetas correctas, en los casos de ambigüedad, el etiquetador estocástico escoge la secuencia de estados que *maximiza* la probabilidad emisión de observaciones, esto es, escoge la secuencia de observaciones que tenga la probabilidad más alta, aplicando el mencionado *algoritmo de Viterbi*:

$$\mathbf{max} \prod_{i}^{n} P(w_i/t_i) P(t_i/t_{i-1})$$

Por otra parte, con esta técnica se podría estimar no sólo la probabilidad con las que aparecen determinadas unidades –como morfemas, palabras, o etiquetas POS– sino también otro tipo de contenidos en un sistema de RI –etiquetas semánticas, o cualquier tipo de marcas–. Para ello bastaría con anotar un *corpus* con la información pertinente por medio de un lexicón,

contar la frecuencia con la que aparecen determinadas unidades, registrar y contar los *n-gramas* y entrenar al etiquetador para que identifique el contexto en que aparecen determinadas unidades, por eso es fundamental obtener los datos de la matriz de probabilidad condicionada por el contexto.

Si se utilizara un ADF Probabilístico para la etiquetación de un sencillo sintagma nominal, como `NP1 = free text searching`, se tendrían que realizar los respectivos cálculos estadísticos de los *n-gramas* y obtener a continuación las matrices de probabilidades. Con este procedimiento el etiquetador estocástico conseguiría eliminar la ambigüedad en la asignación de etiquetas a las cadenas `free` y `searching`, la primera cadena por la posibilidad de asignación de tres etiquetas —`(V) Verbo, (ADV) Adverbio, (A) Adjetivo`— y la segunda por la posibilidad de asignación de otras tres etiquetas —`(N) Nombre, (A) Adjetivo, (V) Verbo`— . Siguiendo con este supuesto, se registrarían y se contarían en el *corpus* la frecuencia conjunta de *palabras / etiquetas*, además de contar la frecuencia con la que aparece una determinada etiqueta:

```
(free, V) = 78
(free, ADV) = 84
(free, A) = 152
(text, N) = 314
...
```

y para obtener la probabilidad de ocurrencia de *palabra / etiqueta* se aplicaría la fórmula:

$$P(w_i/t_i) = \frac{f(w_i, t_i)}{f(t_i)}$$

con la cual obtendríamos la siguiente matriz de probabilidad de observación:

$$P(\text{free} \mid \text{V}) = 0.1$$
$$P(\text{free} \mid \text{ADV}) = 0.2$$
$$P(\text{free} \mid \text{A}) = 0.5$$
$$P(\text{text} \mid \text{N}) = 1$$
$$P(\text{searching} \mid \text{N}) = 0.6$$
$$P(\text{searching} \mid \text{A}) = 0.3$$
$$P(\text{searching} \mid \text{V}) = 0.4$$

De la misma forma, se contaría en este caso la frecuencia de cada *pareja*, o trío, de etiquetas que aparecen en el *corpus de entrenamiento*,

$$(DET, N) = 780$$
$$(N, N) = 903$$
$$(N, A) = 800$$
$$\ldots$$
$$(\text{DET, N, N}) = 768$$
$$(\text{DET, N, A}) = 854$$
$$(\text{DET, N, V}) = 890$$
$$\ldots$$

Además, sería necesario obtener la frecuencia con la que aparece una etiqueta. Con todos estos datos, se crearían los *bigrama* con los que obtendríamos la probabilidad condicionada de aparición de las etiquetas, mediante la fórmula:

$$P(t_i / t_{i-1}) = \frac{f(t_{i-1}, t_i)}{f(t_{i-1})}$$

que nos permitiría obtener la siguiente *matriz de probabilidad condicionada*, teniendo en cuenta que el *contexto de aparición de etiquetas*, representado en la siguiente *tabla de transiciones* (Fig. 4.32):

$P(N \mid N) = 0.5$
$P(N \mid V) = 0.4$
$P(N \mid A) = 0.7$
$P(N \mid ADV) = 0.3$
...

	N	V	A	ADV
N	0.5	0.6	0.3	0.2
V	0.4	0.1	0.3	0.7
A	0.7	0.4	0.5	0.3
ADV	0.3	0.4	0.2	0.1

Fig. 4.32: Matriz de probabilidades de transición entre etiquetas

En la matriz o tabla de probabilidades de transición se indicarían las probabilidades estimadas de que una etiqueta siga a otra en una secuencia. Así, según la matriz de probabilidades, que en este caso son inventadas, dada la etiqueta N la probabilidad de que la siguiente etiqueta sea N sería $P(N \mid N)=0.5$, de que sea V sería $P(V \mid N)=0.6$, o de que sea A sería $P(A \mid N)=0.3$. Además de estos datos, se podría calcular la probabilidad de aparición de una unidad al principio del texto considerando el '*espacio en blanco*' como una unidad del *bigrama*, así la estimación de la probabilidad de que una secuencia se inicie con una determinada categoría podría ser:

$P(N) = 0.5$
$P(V) = 0.4$
$P(A) = 0.6$
$P(ADV) = 0.4$

Si añadimos estos datos a un *Autómata de Estado-Finito Probabilístico* obtendremos una *cadena de Markov*. Una forma de representación de los *Autómatas Probabilísticos* es aquella en la que los estados están representados por categorías sintácticas y cada transición se asocia con una probabilidad que indica el *path* que explora el autómata para reconocer la combinación de etiquetas más probable (Fig. 4.33).

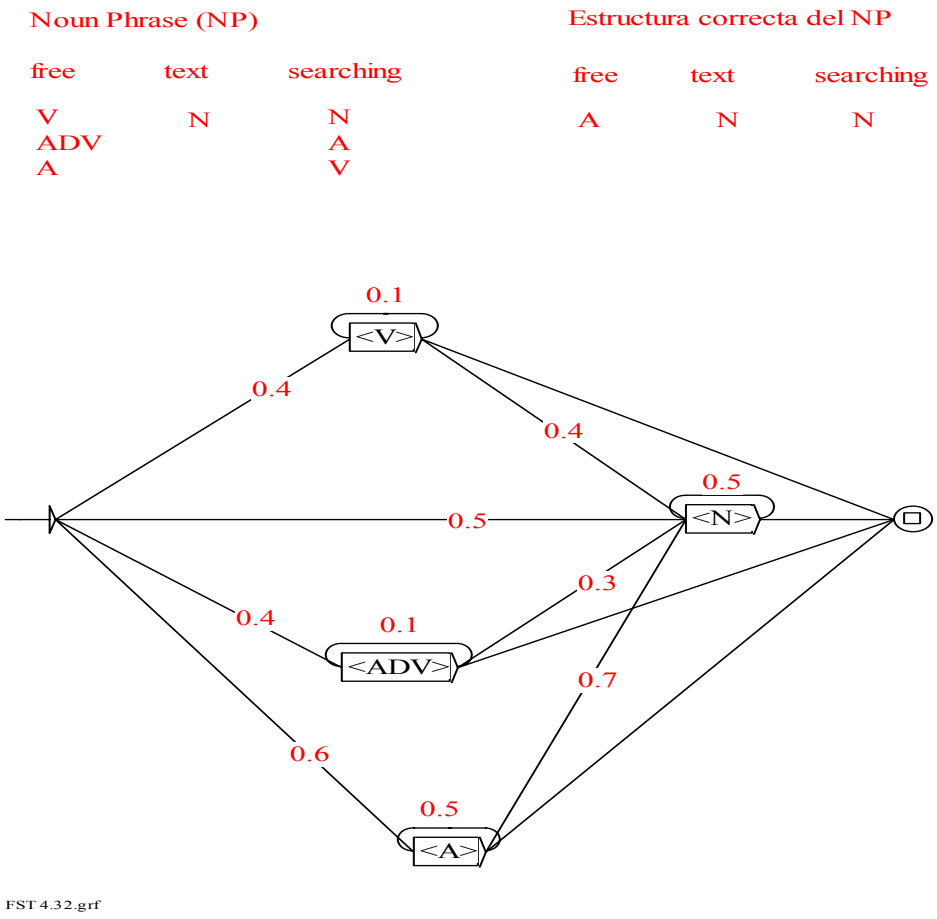


Fig. 4.33: Representación simplificada de un Autómata Probabilístico

A su vez, para resolver el problema de la ambigüedad cuando una cadena puede recibir la asignación de más de una categoría POS el etiquetador estocástico se representa como un *Transductor Probabilístico*, o *Modelo Oculto de Markov o Hidden Markov Model (HMM)*, en el que se combina *la matriz de probabilidad de observación* con la *matriz de probabilidad de transición*, o probabilidad condicionada por el contexto, mediante la fórmula siguiente:

$$\prod_{i}^{n} P\left(w_i/t_i\right) P\left(t_i/t_{i-1}\right)$$

Como en un HMM cada estado genera *observaciones*, en este caso *palabras / etiquetas*, pero sólo percibimos las *observaciones*, es necesario inferir los *estados ocultos*. Con este objetivo se aplica el *algoritmo de Viterbi*, que proporciona las secuencias de *estados ocultos*, esto es, las secuencias de etiquetas correctas con las que se va a eliminar la ambigüedad en la etiquetación.

El procedimiento de desambiguación, del caso concreto que estamos planteando, consistiría en el producto de la multiplicación de *observaciones de palabras* con una probabilidad mayor, $(0.5 * 1 * 0.6)$, por la multiplicación de *secuencias de categorías* con una probabilidad mayor, $\mathtt{A} \rightarrow \mathtt{N} \rightarrow \mathtt{N}$ $(0.4 * 0.7 * 0.5)$. El resultado final sería la etiquetación del sintagma nominal con la probabilidad mayor, obtenida del producto de las *matrices de transición* y de las *matrices de observación*. Por último, con este resultado se conseguiría inferir los *estados ocultos*, *o secuencia de etiquetas correctas*: `A N N` (**Adjetivo Nombre Nombre**):

$$\mathbf{max} \prod_{i}^{n} P\left(w_i / t_i\right) P\left(t_i / t_{i-1}\right)$$

$$(0.5 * 1 * 0.6) \ (0.6 * 0.7 * 0.5) = 0.0063$$

Expresado formalmente, los etiquetadores que utilizan métodos estadísticos para la eliminación de la ambigüedad calculan la secuencia de etiquetas más probables haciendo uso de la siguiente fórmula, adaptada de la *estadística Bayesiana*:

$$P\left(C \mid e\right) = \left[P\left(c_1\right) \prod_{i=2}^{n} P\left(c_i \mid c_{i-1}\right)\right] \times \left[\prod_{i=1}^{n} P\left(T_i \mid c_i\right)\right]$$

donde

- C es la secuencia de etiquetas que representa las distintas estructuras del NP

- e es el NP de entrada, $e = T_1\ T_2\ T_3 \ldots T_n$, formado por las unidades léxicas o términos T_i
- c_i son las etiquetas gramaticales
- n es el número de términos del NP

Con la aplicación de la fórmula anterior se puede determinar cuál es la secuencia de etiquetas correcta para representar la estructura de un NP. En el caso de que partiéramos del sintagma `NP1 = free text searching`, las combinaciones posibles serían nueve:

```
V N N
ADV N N
A N N

V N A
ADV N A
A N A

V N V
ADV N V
A N V
```

De estas nueve secuencias se tendría que seleccionar la más probable teniendo en cuenta:

- *La probabilidad de transición entre etiquetas*.
- *La probabilidad de que un término pertenezca a una determinada categoría*.

El modo de calcular las probabilidades de las secuencias aplicando la fórmula anterior sería el siguiente:

- `V N N` (**Verbo Nombre Nombre**)

  $P(C \mid T) = P(V) \times P(\text{free} \mid V) \times P(N \mid V) \times P(\text{text} \mid N) \times P(N \mid N)$
  $\times P(\text{searching} \mid N) =$
  $0.4 \times 0.1 \times 0.4 \times 1 \times 0.5 \times 0.6 = 0.0048$

- `ADV N N` (**Verbo Nombre Nombre**)

$$P(C \mid T) = P(ADV) \times P(free \mid ADV) \times P(N \mid ADV) \times P(text \mid N) \times P(N \mid N) \times P(searching \mid N) = 0.4 \times 0.2 \times 0.3 \times 1 \times 0.5 \times 0.6 = 0.0072$$

- A N N (**Adjetivo Nombre Nombre**)

$$P(C \mid T) = P(A) \times P(free \mid A) \times P(N \mid A) \times P(text \mid N) \times P(N \mid N) \times P(searching \mid N) = 0.6 \times 0.5 \times 0.7 \times 1 \times 0.5 \times 0.6 = 0.063$$

- V N A (**Verbo Nombre Adjetivo**)

$$P(C \mid T) = P(V) \times P(free \mid V) \times P(N \mid V) \times P(text \mid N) \times P(A \mid N) \times P(searching \mid A) = 0.4 \times 0.1 \times 0.4 \times 1 \times 0.3 \times 0.3 = 0.00144$$

- ADV N A (**Adverbio Nombre Adjetivo**)

$$P(C \mid T) = P(ADV) \times P(free \mid ADV) \times P(N \mid ADV) \times P(text \mid N) \times P(A \mid N) \times P(searching \mid A) = 0.4 \times 0.2 \times 0.3 \times 1 \times 0.3 \times 0.3 = 0.00216$$

- A N A (**Adjetivo Nombre Adjetivo**)

$$P(C \mid T) = P(A) \times P(free \mid A) \times P(N \mid A) \times P(text \mid N) \times P(A \mid N) \times P(searching \mid A) = 0.6 \times 0.5 \times 0.7 \times 1 \times 0.3 \times 0.3 = 0.0189$$

- V N V (**Verbo Nombre Verbo**)

$$P(C \mid T) = P(V) \times P(free \mid V) \times P(N \mid V) \times P(text \mid N) \times P(V \mid N) \times P(searching \mid V) = 0.4 \times 0.1 \times 0.4 \times 1 \times 0.6 \times 0.4 = 0.00384$$

- ADV N V (**Adverbio Nombre Verbo**)

$$P(C \mid T) = P(ADV) \times P(free \mid ADV) \times P(N \mid ADV) \times P(text \mid N) \times P(V \mid N) \times P(searching \mid V) = 0.4 \times 0.2 \times 0.3 \times 1 \times 0.6 \times 0.4 = 0.00576$$

- A N V (**Adjetivo Nombre Verbo**)

$$P(C \mid T) = P(A) \times P(free \mid A) \times P(N \mid A) \times P(text \mid N) \times P(V \mid N) \times P(searching \mid V) = 0.6 \times 0.5 \times 0.7 \times 1 \times 0.6 \times 0.4 = 0.0504$$

Una vez calculada las probabilidades de las secuencias, el procedimiento para la eliminación de la ambigüedad consistiría en escoger la secuencia con una probabilidad mayor. En este caso, la secuencia con mayor probabilidad es `A N N` (**Adjetivo Nombre Nombre**) y el resultado será la asignación de esta secuencia de etiqueta a la construcción nominal `NP1=free text searching .`

Para resolver el problema de la ambigüedad entre varias etiquetas candidatas contamos con métodos estadísticos, como los expuestos anteriormente, pero también con métodos basados en reglas. A pesar de la complejidad de métodos estadísticos, los etiquetadores estocásticos ofrecen mejores resultados y son más fáciles de entrenar que los etiquetadores los basados en reglas, mucho más costosos de construir, según algunos estudios comparativos (Chanod y Tapanainen 1995).

Sin embargo, el procedimiento para la eliminación de la ambigüedad propuesto por Silberztein (Silberztein 1999) es bastante sencillo en comparación con otros métodos. El modelo de Silberztein emplea un procedimiento basado en reglas por medio de la aplicación de *Gramáticas Locales* (GL) definidas como *reglas de dos-partes* representadas en FST gráficos. Las GL actuarían después de la aplicación de las herramientas de análisis léxicos, en los casos en los que fuera preciso desambiguar las cadenas de entrada que estén vinculadas a varias etiquetas. La desambiguación léxica y sintáctica es esencial porque las entradas al *parsing* sintáctico nunca pueden ser ambiguas.

Después de aplicar los diccionarios electrónicos, el texto de entrada quedaría representado, con la aplicación desarrollada por Silberztein (Silberztein 1999), de tres modos distintos:

1. Etiquetado *lineal*.
2. Etiquetado en forma de *Expresión Regular*.
3. Etiquetado en forma de *Transductor gráfico*.

Partiendo de que el texto de entrada es el sintagma `NP1=free text searching`, las tres representaciones se establecerían del modo siguiente:

- Etiquetado lineal:

```
free {text,.N} searching
```

- Etiquetado en forma de Expresión Regular:

```
({free,.A} + {free,.ADV} + {free,.V:W:P1s:P2s:P1p:P2p:P3p})
{text,.N}
({searching,search.V:G} + {searching,.A} + {searching,.N:s})
```

- Etiquetado en forma de FST gráfico (Fig. 4.34):

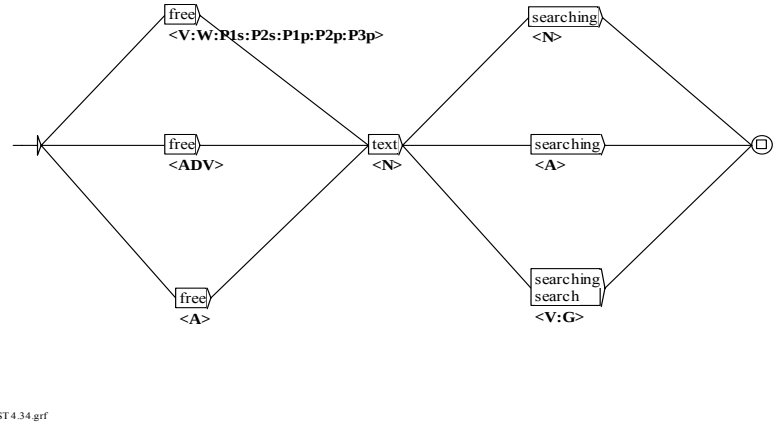


Fig. 4.34: Representación del etiquetado con ambigüedad en un FST gráfico

Sobre las tres representaciones pueden operar las GL, con el objetivo de la desambiguación de las etiquetas correspondientes a las unidades léxicas. Sin embargo, el texto representado linealmente no pueden formalizar todas las ambigüedades del etiquetado, porque sólo permite la etiquetación de las unidades no ambiguas. Por lo tanto, las GL únicamente actuarían en las representaciones del texto en forma de Expresiones Regulares, o en forma de FST gráficos.

Para la desambiguación del sintagma `NP1=free text searching` se debería construir una GL cuyo objetivo sea eliminar las transiciones consideradas incorrectas. Dicha gramática se representaría a su vez por medio de un FST gráfico (Fig. 4.35), cuya función sería vincular a cada unidad reconocida determinadas restricciones que se utilizarían para *destruir* transiciones, pero únicamente en ese contexto de aparición de etiquetas.

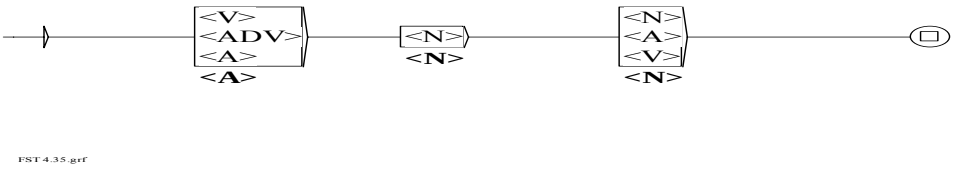


Fig. 4.35: *Gramática Local* representada en un FST gráfico

Después de la aplicación de la GL anterior al sintagma `NP1=free text searching`, se obtendría una asignación de etiquetas sin ambigüedad, en cualquiera de las representaciones mencionadas:

- Etiquetado lineal:

```
{free,.A} {text,.N} {searching,.N}
```

- Etiquetado en forma de Expresión Regular:

```
{free,.A}
{text,.N}
{searching,.N}
```

- Etiquetado en forma de FST gráfico (Fig. 4.36):

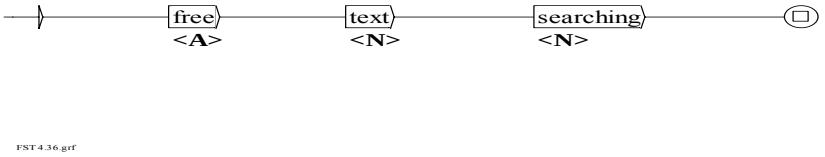


Fig. 4.36: Representación del etiquetado sin ambigüedad en un FST gráfico

Según el procedimiento anterior, para la desambiguación en la asignación de etiquetas es preciso que el texto se represente bien por medio de *Expresiones Regulares*, o bien por medio de *Transductores Gráficos*. La desambiguación sobre Expresiones Regulares llega a ser demasiado extensa cuando se trata de eliminar la ambigüedad de series de palabras compuestas porque precisa de muchas copias de las propias expresiones que la componen, llegando a provocar que esta notación se vuelva ilegible. Sin embargo, el mismo proceso sobre FST constituyen una notación adecuada porque las cadenas se asocian de un modo perceptible a distintas informaciones léxicas, o *hipótesis léxicas* (Silberztein 2000). De esta forma, el proceso de desambiguación con GL operaría sobre la representación del texto en FST gráficos. A su vez, las GL se representarían también en FST, cuya función principal será, como ya se ha mencionado, *suprimir* determinadas transiciones.

A pesar de esto, la eliminación de la ambigüedad por medio de GL puede introducir errores porque en el proceso de reconocimiento no siempre a la unidad léxica `free` se le debe asignar la etiqueta < **A** >, o a la unidad `searching` la etiqueta < **N** >, aunque estén en ese mismo contexto de aparición. Por esta razón, es preciso tener en cuenta la distinción entre las denominadas *gramáticas perfectas* e *imperfectas* (Silberztein 2000). Una *gramática «perfecta»* sería aquella que se aplica a textos en un lenguaje general, y que no dan lugar a errores (Fig. 4.37). Una *gramática «imperfecta»* sería aquella que se aplica a textos

especializados en un dominio de conocimiento en un lenguaje específico, y que provocaría errores si se intentara aplicar a textos en un lenguaje general. En consecuencia, el uso estas gramáticas se debe limitar a casos concretos para evitar errores en la etiquetación que, más que solucionar, agraven el problema de la ambigüedad.

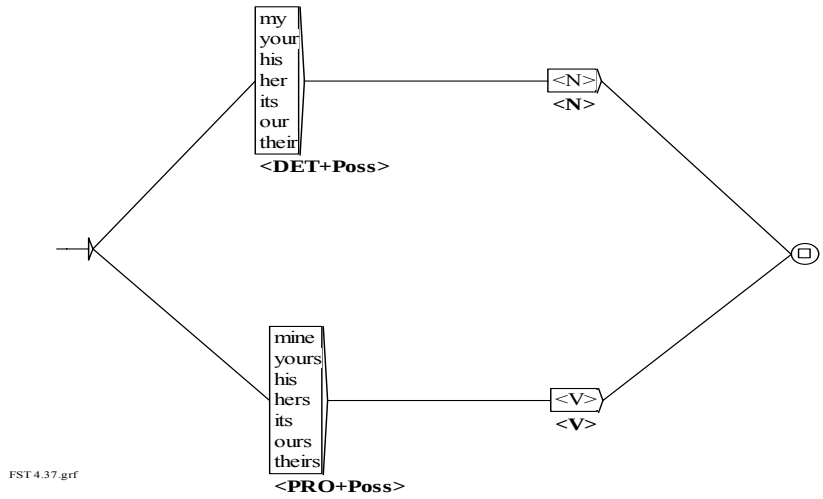


Fig. 4.37: Gramática Local que no produce errores

En el proceso de desambiguación sintáctica también se pueden emplear GL, sin embargo presenta algunas diferencias con respecto a la desambiguación de etiquetas. En la eliminación de la ambigüedad de las etiquetas, las entradas a las GL son ambiguas, en el sentido de que a una unidad se le pueden asignar más de una etiqueta, frente a esto en la eliminación de la ambigüedad sintáctica, las entradas a las GL no lo son, y en este caso su uso se limita a eliminar determinadas transiciones. Por ejemplo, un tipo de *Gramática Local* cuya función específica fuera eliminar la ambigüedad sintáctica se encargaría de etiquetar los determinantes demostrativos, `<DET+Ddem>`, como tales cuando preceden a un nombre, `<N>`, o como pronombres, `<PRO+Pdem>`, cuando preceden a un verbo, `<V>`. En este caso, la GL se encargaría únicamente de *destruir* determinadas transiciones (Fig. 4.38).

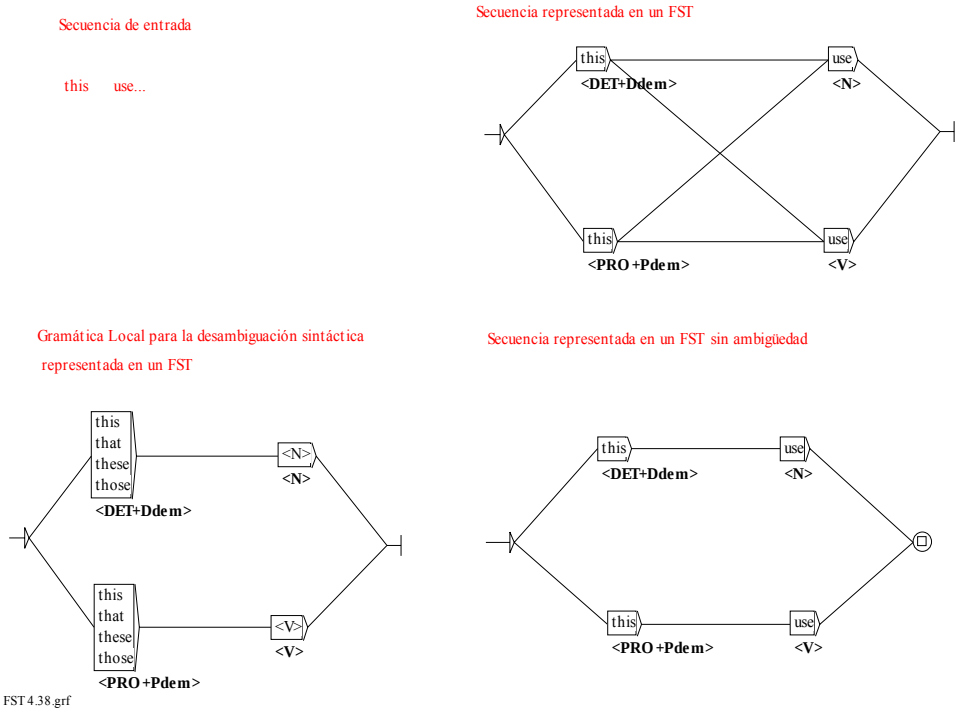


Fig. 4.38: Desambiguación sintáctica por medio de Gramáticas Locales

El formalismo que vamos a utilizar cuando sea preciso eliminar la ambigüedad de determinadas unidades lingüísticas se va a basar en reglas, descritas en términos de GL representadas en FST gráficos. Sin embargo, es muy difícil eliminar la ambigüedad por completo, tanto con éste como con otros métodos. Aún así, la solución para el reconocimiento de las estructuras sintácticas canónicas de los NNPP sobre etiquetas sin ambigüedad se va a desarrollar con un nuevo y original método consistente en realizar *parsing* sintáctico, no directamente sobre las *cadenas*, sino sobre el texto representado en forma de *Transductores gráficos*.

Siguiendo este procedimiento, el reconocimiento de las estructuras sintácticas de los NNPP, se va a realizar por medio de gramáticas, representadas en FST gráficos, que se confrontan, o *intersectan*, al texto representado también en FST gráficos (Fig. 4.39), de esta forma las gramáticas se equiparan siempre a etiquetas no ambiguas. Mediante esta operación las

proyecciones del FST, que representa al texto, se configuran como las entradas al FST, que representa la gramática. Se trata de una aplicación en *cascada* de los transductores gráficos, según la cual el *output* de un transductor se considera el *input* de otro transductor.

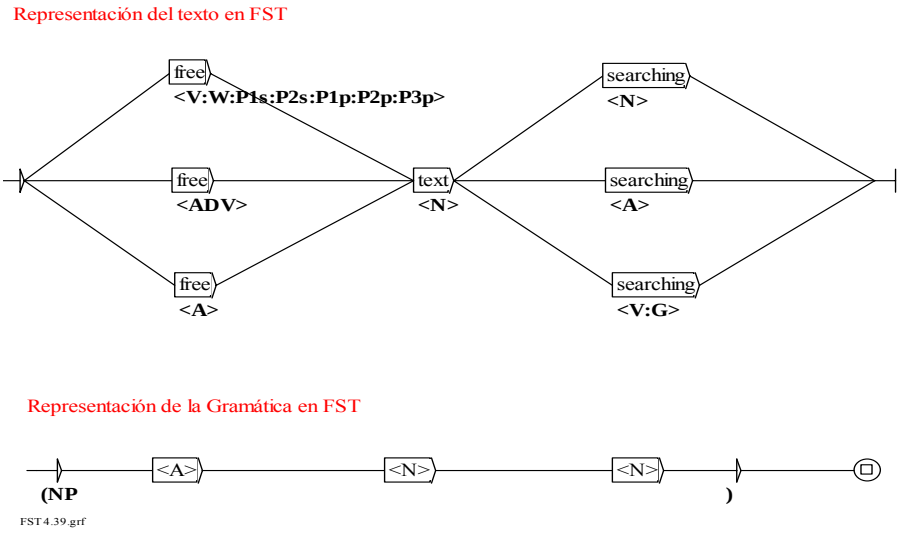


Fig. 4.39:Intersección de las gramáticas, representadas en FST, con el texto, representado en FST

En un capítulo posterior se va a realizar la equiparación de las *estructuras sintácticas canónicas* de los NNPP, que consistirá básicamente en la confrontación de las gramáticas que describen las estructuras de los NNPP con los transductores que representan al texto. Las gramáticas tendrán como función asociar determinadas marcas, o *paréntesis etiquetados*, a los NNPP reconocidos. Posteriormente, las estructuras sintácticas canónicas de tales NNPP se podrán utilizar para llevar a cabo otro tipo de operaciones como entradas a un nuevo *parsing* sintáctico, o entradas a un *índice*.

En el proceso anterior, sólo después de haber aceptado las etiquetas el transductor, que representa la gramática, es capaz de transformarla en otra. Sin embargo, una secuencia de unidades léxicas, o etiquetas, que no pertenezca al lenguaje reconocido por el transductor puede contener otras unidades, o etiquetas, que sí formen parte del lenguaje aceptado por dicho transductor. Con el objetivo de reconocer estas estructuras se introducirán las

mencionadas *Extensiones Locales* de los transductores, esto es, ampliaciones con `transiciones-?` y `transiciones-λ` según la propuesta de Roche y Schabes (Roche y Schabes 1995). Las `transiciones-?`, o transiciones de un símbolo por sí mismo, van a permitir aceptar cualquier símbolo de la cadena de entrada que no sea reconocido por el transductor y transformarlo por sí mismo. Las `transiciones-λ`, o transiciones vacías representadas por el símbolo `<E>`, situadas muchas veces en los estados finales van a permitir que el transductor vuelva a situarse en el estado inicial. De cualquier forma, todo este proceso se comprobará mucho mejor en otro capítulo cuando se confronten las gramáticas, que se van a desarrollar, con las secuencias textuales del *corpus de verificación*.

Capítulo 5

CONSTRUCCIÓN DE ANALIZADORES LÉXICOS CON TÉCNICAS DE ESTADO-FINITO

La unidad básica objeto del análisis léxico es la *palabra flexionada* y la unidad básica objeto del análisis sintáctico es el *sintagma*. De esta forma, las palabras, compuestas por morfemas, constituirían las unidades propias del análisis morfológico y los sintagmas, compuestos por palabras etiquetadas, constituirían las unidades propias del análisis sintáctico. Las cuestiones relativas a los tipos de relaciones que mantienen estas unidades lingüísticas, su clasificación, las distintas formas de flexión, así como sus alteraciones formales, se plantean aquí como propiedades relevantes porque todas estas características se han de tener en cuenta en el modelo que se adopte para su representación.

Las palabras se someten a un análisis flexional –que tiene como representante más claro el *género* y el *número* en las formas nominales, o la *conjugación* en las formas verbales– cuya función es el establecimiento de la concordancia dentro de los componentes de la oración. El análisis flexional se rige por reglas que aportan los mecanismos para poder relacionar los

distintos elementos en el contexto de la oración. Además, el análisis de número, género o el de la conjugación verbal está gobernado por reglas que determinan el grado de variación en la flexión. Frente a esto, otro tipo de análisis, como es el derivacional, no se somete tan claramente a la regularidad de las reglas y muchas palabras compuestas, o derivadas de otras, llegan a transformarse totalmente y se alejan de la palabra origen. La variabilidad en la derivación, y el hecho de que en las palabras derivadas los afijos formen parte del *stem*, hace que sea muy difícil fijar cualquier tipo de regularidad en su representación, por esta razón este análisis queda excluido de este trabajo.

El desarrollo de los analizadores léxicos nos permitirá distinguir las formas flexivas y las irregularidades que se producen en los distintos tipos de flexión y asignar las distintas categorías gramaticales a las unidades lingüísticas flexionadas. El objetivo fundamental que se va a tratar en este capítulo es describir la metodología que se ha seguido para la construcción de los recursos de análisis léxico –*diccionarios electrónicos* y *transductores léxicos*– a partir de la aplicación informática basada en la tecnología de estado-finito que ha sido diseñada por Silberztein (Silberztein 1996; Silberztein 2000). El desarrollo de los analizadores léxicos con esta aplicación nos permitirá representar las formas flexivas y las irregularidades que se producen en los distintos tipos de flexión, y asignar las distintas categorías gramaticales a las unidades lingüísticas flexionadas.

En un principio, en el proceso de flexión se va a tomar como base una unidad genérica denominada *lema*, definida como un conjunto de palabras con el mismo *stem* y la misma categoría léxico-gramatical general –por ejemplo, el *lema* de la cadena `usuario` estaría compuesto por el conjunto {`usuario,usuarios,usuaria,usuarias`} formado por todas las cadenas con el mismo *stem* y la misma categoría general de N (*Nombre*) –. Según la complejidad de la estructura del *stem*, las unidades que integran cada una de estas clases se van a agrupar según los distintos tipos de relación que se establecen en el *eje sintagmático*, o de combinación con las unidades presentes, y en el *eje paradigmático*, o de selección con las unidades de la misma clase formal.

Los analizadores léxicos mantienen una relación directa con el vocabulario o las palabras de clase abierta, fundamentalmente *nombres*, *verbos* y *adjetivos*, en torno a las cuales operan los morfemas flexivos proporcionando información sobre la función que tienen dichas palabras en el uso lingüístico. Para la representación y reconocimiento de las clases de palabras anteriores, vamos a adoptar, como *hipótesis explicativa*, el *lema* como base del proceso de descripción y el *stem* como unidad básica del proceso de flexión. Como ya se ha dicho, una palabra es una unidad compleja compuesta básicamente por dos elementos, *stem* y *afijos flexivos*, que tiene por tanto dos clases de elementos constituyentes –una parte constante y una parte variable que aporta significado gramatical– que se pueden materializar en una *estructura binaria* susceptible de ser representada con técnicas de estado-finito.

Después de vincular el *stem* a determinados afijos flexivos se puede hablar realmente de la unidad lingüística denominada *palabra*, pero el análisis de la flexión no es suficiente para su clasificación y reconocimiento. Para que pueda funcionar el componente sintáctico es necesario que se hayan distinguido previamente las categorías léxico-gramaticales a las que pertenecen dichas palabras. Por esta razón, el análisis morfológico tiene como objetivo no sólo identificar y organizar las formas flexivas de las palabras sino asignarles determinadas categorías gramaticales. En consecuencia, sólo a partir de la complementación de un análisis morfológico y gramatical podemos llegar a una auténtica distinción de esta unidad lingüística. Con este propósito es preciso desarrollar herramientas que se ocupen tanto del análisis flexional de las palabras, como del etiquetado de categorías léxico-gramaticales, o etiquetado *part-of-speech* (POS).

El desarrollo de las herramientas de análisis léxicos se inicia con la representación de la flexión de las palabras. En relación con esto, las formas flexivas se integran en conjuntos limitados de miembros, como son los denominados *paradigmas de flexión nominal, verbal y adjetival*. Aquí el término paradigma se utiliza en la acepción de *conjunto de formas relacionadas por procesos flexionales generales* y las relaciones paradigmáticas serían las que se constituyen entre la formas de un mismo paradigma, esto es, las *formas que pueden ocupar la misma posición en una cadena de palabras consecutivas* (Chomsky 1957). Según

la definición anterior, las formas flexivas de una palabra se agrupan en un *sistema cerrado* o *paradigma flexivo* (Matthews 1965; Matthews 1972) que encierra una enumeración ordenada de todas y cada una de las formas que puede presentar un *stem*, tomado en este caso como base común de cualquier forma flexiva dentro del mismo paradigma. En este contexto, es necesario resaltar la relevancia de la vocal del *stem*, que se manifiesta en la flexión nominal asignando los nombres a una determinada clase flexiva, o en la flexión verbal indicando la pertenencia del verbo a una determinada clase flexiva.

El conjunto de elementos que forman el paradigma tiene un valor constante representado por el *stem* y distintos códigos intracategoriales representados por la flexión. Los elementos dentro del mismo paradigma mantienen una relación de oposición, en la que la presencia de un código excluye a otro. Las oposiciones en el interior del paradigmática flexivo se organizan en torno a las distintas categorías léxico-gramaticales, o categorías *part-of-speech* (POS). Dentro de la categoría número se establece la oposición `singular/plural` y dentro de la categoría género `masculino/femenino`. A su vez, las oposiciones intracategoriales entre las unidades del mismo paradigma tienen la propiedad de ser recurrentes y regulares: así, la oposición masculino singular o plural, `ms/mp`, se repite en femenino singular o plural, `fs/fp`.

Otra noción fundamental es la *estructura del paradigma*, que hace referencia al número de categorías que puedan aparecer en el interior de los paradigmas. Dependiendo de la variación de ese número se puede hablar de *estructuras simples*, cuando intervenga sólo una dimensión o categoría, y *estructuras complejas*, cuando intervengan varias dimensiones o categorías (Coseriu 1981). El paradigma nominal se considera una estructura simple, en el que operan las categorías de número y género. El paradigma verbal tiene una estructura más compleja, pudiéndose distinguir el paradigma de la *primera*, *segunda* y *tercera conjugación*, que a su vez está compuesto por los paradigmas de *tiempo*, *modo*, *aspecto*, *número* y *persona*.

Las unidades léxicas que se van a considerar son básicamente: *palabras simple*s, *palabras compuestas*, *expresiones fijas* y *expresiones de dominio*. Para la construcción de los analizadores léxicos es preciso desarrollar previamente dos herramientas:

1. *Diccionarios, o lexicones, electrónicos de palabras simples y compuestas*
2. *Transductores de Estado-Finito Léxicos.*

Los diccionarios y los transductores léxicos nos permitirán reconocer las variantes de forma de una palabra, simple o compuesta, y cómo esas variantes muestran los distintos significados gramaticales a través de la flexión. Los transductores léxicos, además, nos permitirán agrupar formas derivadas o expresiones sinónimas mediante una representación gráfica. Posteriormente, se comprobará que, para que se pueda realizar el proceso de reconocimiento, las unidades léxicas, que componen los diccionarios, se tienen que compilar y compilar en FST. También se demostrará cómo se pueden localizar en los textos determinadas expresiones a través de la representación gráfica de los FST léxicos.

5.1. Construcción de Diccionarios electrónicos

Las unidades léxicas simples se representan en diccionarios electrónicos cuyas entradas son *lemas* vinculados a información morfológica flexional. La *hipótesis explicativa* de la que partimos para equiparar *Formas flexionadas* a *Formas léxicas* es adjuntar al *lema* un código morfo-sintáctico, que corresponde al nombre de la categoría a la que pertenece, y vincularlo con un FST gráfico que contiene la descripción de la morfología flexional. La *hipótesis* para la identificación y agrupación de variantes flexionales se basa en el establecimiento de una *Relación Regular* entre el *lema* y la *descripción flexional*. De este modo, todos los *lemas*, que pertenezcan al mismo paradigma flexivo, se relacionarán con el mismo FST gráfico. A su vez, los nombres de los FST se corresponderán con los códigos morfo-sintácticos de los lemas. Las entradas del diccionario de *lemas* tienen una distribución parecida a la siguiente:

científico,**N1**

reportero,**N1**

experimental,**A2**

documental,**A2**

éste,**PRODE1**

iniciar,**V1**

solucionar,**V1**

alguno,**CUANT3**

/.../

El código morfo-sintáctico **N1** se correspondería con el **FST N1** (Fig. 5.1), el código **A2** con el **FST A2** (Fig. 5.2), el código **PRODE1** con el **FST PRODE1** (Fig. 5.3), el código **CUANT3** con el **FST CUANT3** (Fig. 5.4) y el código **V1** con el **FST V1** (Fig. 5.5). Por otra parte, es necesario aclarar que en los FST gráficos aparece el `operador de borrado L` que se utiliza para poder obtener la flexión del *lema* encargándose de eliminar un carácter de la forma canónica, y en los casos en que sea preciso borrar dos o más caracteres se indicaría simplemente con el número correspondiente.

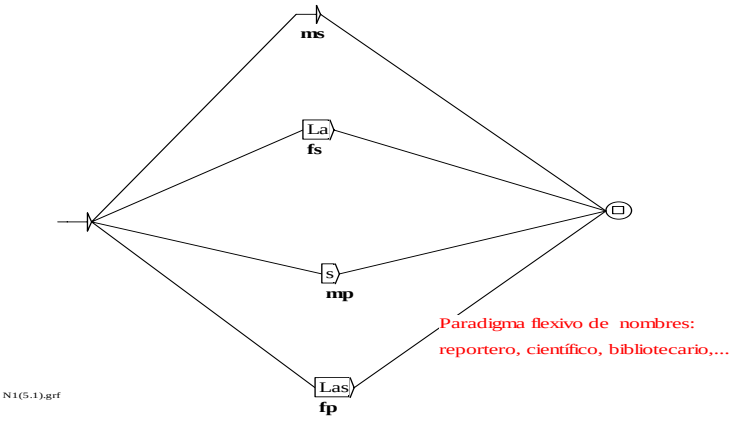
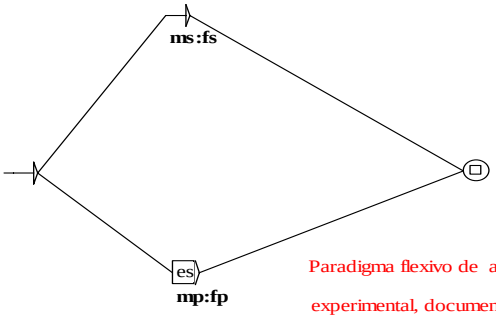
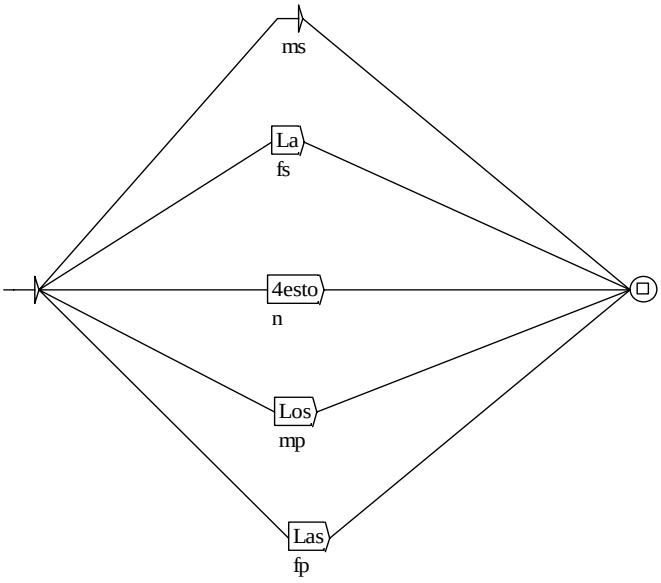


Fig. 5.1: FST N1



A2(5.2).grf

Fig. 5.2: FST A2



PRODE1(5.3).grf

Paradigma flexivo de los pronombres demostrativos: éste, ésta, esto,...

Fig. 5.3: FST PRODE1

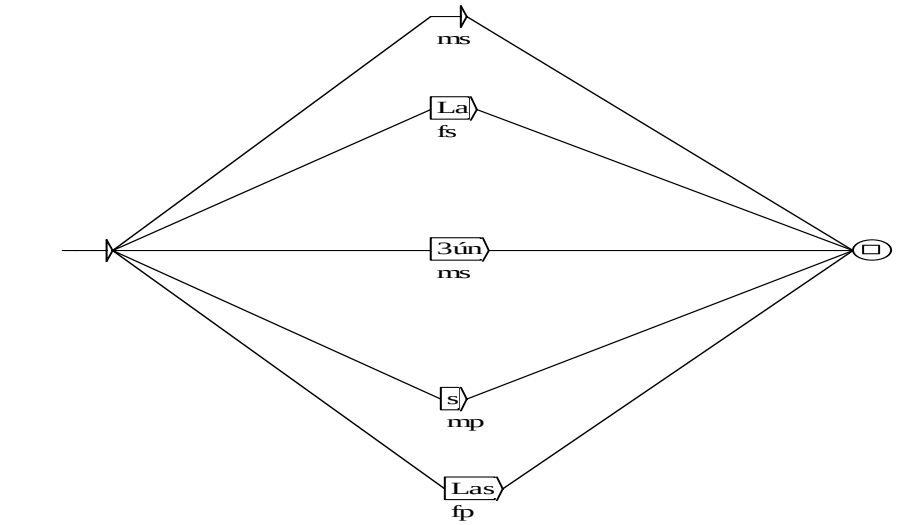


Fig.5.4: FST CUANT3

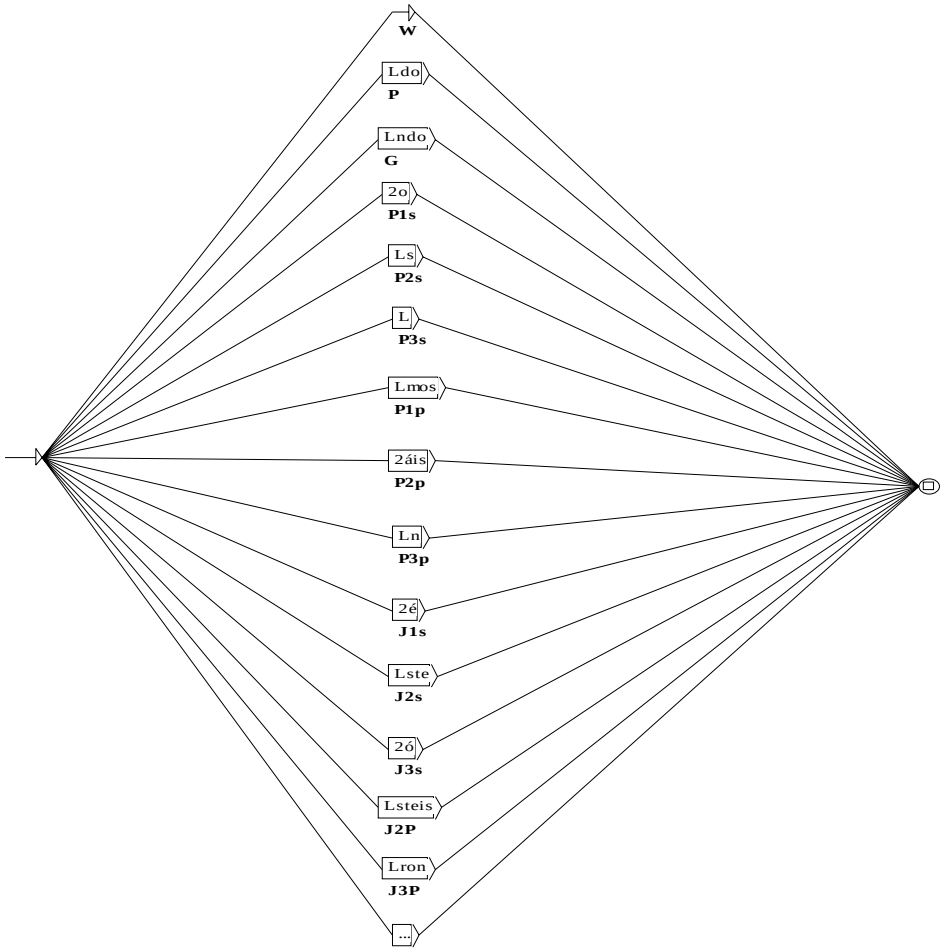


Fig. 5.5: FST V1

De la proyección de los FST gráficos sobre el diccionario de formas canónicas se obtiene automáticamente el *diccionario expandido de formas flexionadas* que se presenta del modo siguiente:

alguna,*alguno*.**CUANT3:fs**
alguno,*alguno*.**CUANT3:ms**
algunas,*alguno*.**CUANT3:fp**
algunos,*alguno*.**CUANT3:mp**
algún,*alguno*.**CUANT3:ms**
científica,*científico*.**N1:fs**

científico,*científico*.**N1:ms**
científicas,*científico*.**N1:fp**
científicos,*científico*.**N1:mp**
documental,*documental*.**A2:ms:fs**
documentales,*documental*.**A2:mp:fp**
ésta,*éste*.**PRODE1:fs**
éstos,*éste*.**PRODE1:mp**
éste,*éste*.**PRODE1:ms**
éstas,*éste*.**PRODE1:fp**
esto,*éste*.**PRODE1:n**
experimental,*experimental*.**A2:ms:fs**
experimentales,*experimental*.**A2:mp:fp**
iniciaréis,*iniciar*.**V1:F2p**
iniciado,*iniciar*.**V1:P**
iniciaran,*iniciar*.**V1:IS3p**
iniciaría,*iniciar*.**V1:C1s:C3s**
iniciaríamos,*iniciar*.**V1:C1p**
iniciar,*iniciar*.**V1:W**
iniciara,*iniciar*.**V1:IS1s:IS3s**
iniciarían,*iniciar*.**V1:C3p**
iniciaras,*iniciar*.**V1:IS2s**
iniciase,*iniciar*.**V1:IS1s:IS3s**
iniciaríais,*iniciar*.**V1:C2p**
iniciaré,*iniciar*.**V1:F1s**
iniciasteis,*iniciar*.**V1:J2P**
iniciarías,*iniciar*.**V1:C2s**
inician,*iniciar*.**V1:P3p**
iniciasen,*iniciar*.**V1:IS3p**
iniciaste,*iniciar*.**V1:J2s**
iniciando,*iniciar*.**V1:G**
iniciaba,*iniciar*.**V1:I3s**
iniciaba,*iniciar*.**V1:I1s**
iniciaban,*iniciar*.**V1:I3p**
inicias,*iniciar*.**V1:P2s**
iniciaron,*iniciar*.**V1:J3P**
iniciases,*iniciar*.**V1:IS2s**
iniciaremos,*iniciar*.**V1:F1p**
iniciabas,*iniciar*.**V1:I2s**
iniciamos,iniciar.**V1:P1p:J1p**
iniciarán,*iniciar*.**V1:F3p**
iniciabais,*iniciar*.**V1:I2p**
iniciarais,*iniciar*.**V1:IS2p**
iniciaseis,iniciar.**V1:IS2p**
inicia,*iniciar*.**V1:P3s:Y2s**
iniciarás,*iniciar*.**V1:F2s**
iniciará,*iniciar*.**V1:F3s**
iniciáramos,*iniciar*.**V1:IS1p**
iniciásemos,*iniciar*.**V1:IS1p**
iniciábamos,*iniciar*.**V1:I1p**
inicié,*iniciar*.**V1:J1s**
inició,*iniciar*.**V1:J3s**
iniciáis,*iniciar*.**V1:P2p**
inicio,*iniciar*.**V1:P1s**
iniciemos,*iniciar*.**V1:S1p**
iniciéis,*iniciar*.**V1:S2p**
inicien,*iniciar*.**V1:S3p**
inicie,*iniciar*.**V1:S1s**
inicies,iniciar.**V1:S2s**
inicie,*iniciar*.**V1:S3s**
reportera,*reportero*.**N1:fs**
reportero,*reportero*.**N1:ms**
reporteras,*reportero*.**N1:fp**
reporteros,*reportero*.**N1:mp**
solucionaréis,*solucionar*.**V1:F2p**
solucionado,*solucionar*.**V1:P**
solucionaran,*solucionar*.**V1:IS3p**
solucionaría,*solucionar*.**V1:C1s:C3s**
solucionaríamos,*solucionar*.**V1:C1p**
solucionar,*solucionar*.**V1:W**
solucionara,*solucionar*.**V1:IS1s:IS3s**

solucionarían,*solucionar*.**V1:C3p**
solucionaras,*solucionar*.**V1:IS2s**
solucionase,*solucionar*.**V1:IS1s:IS3s**
solucionaríais,*solucionar*.**V1:C2p**
solucionaré,*solucionar*.**V1:F1s**
solucionasteis,*solucionar*.**V1:J2P**
solucionarías,*solucionar*.**V1:C2s**
solucionan,*solucionar*.**V1:P3p**
solucionasen,*solucionar*.**V1:IS3p**
solucionaste,*solucionar*.**V1:J2s**
solucionando,*solucionar*.**V1:G**
solucionaba,*solucionar*.**V1:I3s**
solucionaba,*solucionar*.**V1:I1s**
solucionaban,*solucionar*.**V1:I3p**
solucionas,*solucionar*.**V1:P2s**
solucionaron,*solucionar*.**V1:J3P**
solucionases,*solucionar*.**V1:IS2s**
solucionaremos,*solucionar*.**V1:F1p**
solucionabas,*solucionar*.**V1:I2s**
solucionamos,*solucionar*.**V1:P1p:J1p**
solucionarán,*solucionar*.**V1:F3p**
solucionabais,*solucionar*.**V1:I2p**
solucionarais,*solucionar*.**V1:IS2p**
solucionaseis,*solucionar*.**V1:IS2p**
soluciona,*solucionar*.**V1:P3s:Y2s**
solucionarás,*solucionar*.**V1:F2s**
solucionará,*solucionar*.**V1:F3s**
solucionáramos,*solucionar*.**V1:IS1p**
solucionásemos,*solucionar*.**V1:IS1p**
solucionábamos,*solucionar*.**V1:I1p**
solucioné,*solucionar*.**V1:J1s**
solucionó,*solucionar*.**V1:J3s**
solucionáis,*solucionar*.**V1:P2p**
soluciono,*solucionar*.**V1:P1s**
solucionemos,*solucionar*.**V1:S1p**
solucionéis,*solucionar*.**V1:S2p**
solucionen,*solucionar*.**V1:S3p**
solucione,*solucionar*.**V1:S1s**
soluciones,*solucionar*.**V1:S2s**
solucione,*solucionar*.**V1:S3s**

/.../

Las entradas del diccionario de formas flexionadas están compuestas por los siguientes elementos:

a) *Forma canónica*, o *lema*, establecida básicamente a partir de la oposición binaria de los términos *no-marcados*, o negativos, y *marcados*, o positivos. Dentro de la categoría general **N** (*Nombre*) y **A** (*Adjetivo*) se seleccionan los términos no-marcados, que son el *masculino/singular* y dentro de la categoría verbo se selecciona el *infinitivo*.

b) *Categorías léxico-gramaticales*, *categorías* POS, representadas por los códigos siguientes: **A** (*Adjetivo*), **ADV** (*Adverbio*), **ADVIN** (*Adverbio Interrogativo*),

ADVRE (*Adverbio Relativo*), **AIN** (*Adjetivo Interrogativo*), **ARE** (*Adjetivo Relativo*), **CARD** (*Cardinal*), **CUANT** (*Cuantificador*), **CONJC** (*Conjunción de Coordinación*), **CONJS** (*Conjunción de Subordinación*), **DEM** (*Determinante Demostrativo*), **DET** (*Determinante*), **ORD** (*Ordinal*), **PA** (*Participio Adjetival*), **POS** (*Posesivo*), **PREP** (*Preposición*), **PRO** (*Pronombre*), **PRODE** (*Pronombre Demostrativo*), **PROIN** (*Pronombre Interrogativo*), **PRORE** (*Pronombre Relativo*), **V** (*Verbo*).

c) *Información flexional*: **s** (*singular*), **p** (*plural*), **m** (*masculino*), **f** (*femenino*), **n** (neutro), **W** (*Infinitivo*), **P** (*Participio*), **G** (*Gerundio*), **P1s** (*1ª persona del singular del Presente de Indicativo*), **J1s** (*1ª persona del singular del Pretérito Indefinido*), **I1s** (*1ª persona del singular del Pretérito Imperfecto*), **F1s** (*1ª persona del singular del Futuro*), **C1s** (*1ª persona del singular del Condicional*), **Y2s** (*2ª persona del singular del Imperativo*), **S1s** (*1ª persona del singular del Presente de Subjuntivo*), **IS1s** (*1ª persona del singular del Pretérito Imperfecto del Subjuntivo*),...

Además, la clase de paradigma flexivo a la que pertenece cada palabra se indica con un *código numérico* y cada clase flexional se representa en un FST gráfico, que se encargará de generar automáticamente el conjunto de sufijos y códigos flexionales. Este procedimiento se aprecia mejor mostrándolo en los siguientes casos concretos:

- El **FST N2** agruparía la clase flexional de nombres como `descriptor`, `metabuscador`, `terminal`,... flexionados a:

descriptor,*descriptor*,**N10:ms**
descriptores,*descriptor*.**N10:mp**
metabuscador,*metabuscador*.**N10:ms**
metabuscadores,*metabuscador*.**N10:mp**
terminal,*terminal*.**N10:ms**
terminales,*terminal*.**N10:mp**

/.../

- El **FST A7** ordenaría la clase flexional de adjetivos, en los que la consonante `z` del *stem* se transforma en `c` para formar el plural, como `capaz, eficaz,...` flexionados a:

capaz,*capaz*.**A7:ms:fs**
capaces,*capaz*.**A7:mp:fp**
eficaz,*eficaz*.**A7:ms:fs**
eficaces,*eficaz*.**A7:mp:fp**
/.../

- El **FST V312** reuniría la clase de flexión irregular de verbos de la 3ª conjugación, en los que la consonante `c` del *stem* se transforma en `zc` delante de las vocales `a` y `o`, como `traducir, producir, introducir,...` flexionados a:

traduzco,*traducir*.**V312:P1s**
traduces,*traducir*.**V312:P2s**
produzco,*producir*.**V312:P1s**
produces,*producir*.**V312:P2s**
introduzco,*introducir*:**V312:P1s**
introduces,*introducir*.**V312:P2s**
/.../

Del mismo modo, las unidades léxicas compuestas, constituidas por formas simples, se representan en diccionarios electrónicos cuyas entradas son *lemas compuestos* seguidos del nombre de la categoría a la que pertenece. Una pequeña selección de estas entradas se muestra a continuación:

al igual,**ADV**
en consecuencia,**ADV**
a causa de que,**CONJS**

mientras que,**CONJS**
ciento sesenta,**N**
doscientos cincuenta y cinco,**N**
a disposición de,**PREP**
por motivo de,**PREP**
el cual,**PRORE1**
la cual,**PRORE1**
las cuales,**PRORE2**
los cuales,**PRORE2**
/.../

Al igual que las entradas del diccionario de palabras simples, los códigos morfo-sintácticos de las palabras compuestas se vinculan a los FST correspondientes que se encargan de realizar la flexión de forma automática de los *lemas compuestos*, aunque hayan unidades que no sean analizables morfológicamente, el resultado sería el siguiente:

a causa de que,*a causa de que*.**CONJS**
a disposición de,*a disposición de*.**PREP**
al igual,*al igual*.**ADV**
ciento sesenta,*ciento sesenta*.**N:ms**
doscientos cincuenta y cinco,*doscientos cincuenta y cinco*.**N:ms**
el cual,*el cual*.**PRORE1:s**
en consecuencia,*en consecuencia*.**ADV**
la cual,*la cual*.**PRORE1:s**
las cuales,*las cuales*.**PRORE2:p**
los cuales,*los cuales*.**PRORE2:p**
mientras que,*mientras que*.**CONJS**
por motivo de,*por motivo de*.**PREP**
/**.../**

Es necesario anotar que una característica importante de este tipo de diccionarios es que el *lema* de las formas compuestas se puede emplear para vincular distintas variantes a formas más explícitas o expresivas como:

Comunidad Valenciana,*Comunidad Autónoma de Valencia*.**N:fs**

Univ. Extremadura,*Universidad de Extremadura*.**N:fs**

Univ. Carlos III,*Universidad Carlos III*.**N:fs**

Univ. Salamanca,*Universidad de Salamanca*.**N:fs**

Al diccionario de palabras simples y compuestas también se incorporan entradas correspondientes al léxico general de la lengua como **Adverbios**, **Cuantificadores**, **Conjunciones**, **Determinantes**, **Posesivos**, **Preposiciones** y **Pronombres** a los que se vinculan transductores gráficos de flexión, aunque muchas veces no constituyan unidades analizables morfológicamente y los FST sólo tengan como función asignar la categoría *POS* a las referidas unidades léxicas (Fig. 5.6).

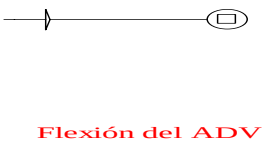


Fig. 5.6: Transductor gráfico de la categoría ADV (*Adverbio*)

Asimismo también se han desarrollado otros tipos de diccionarios que nos aportan las herramientas para reconocer **Términos Especializados**, **Abreviaturas**, **Topónimos**, **Locativos**, o **Siglas**. Las entradas de estos diccionarios están integradas igualmente por formas simples, o compuestas, y van seguidas de las categorías morfo-sintácticas vinculadas a los FST gráficos correspondientes, de este tipo serían las entradas que se indican a continuación:

Unión Europea,**N10+Loc**

Países Bajos,**N102+Top**

Asoc,asociación.**N:ms**

Dep,departamento.**N:mp**

Chem,chemistry.**N:fs**

HyperText,**N10**

network,**N10**

Dialog,**N+PR**

Google,**N+PR**

OPAC,**N103**

Trec,**N10**

URL,**N102**

TCP/IP,**N102**

/…/

Siguiendo el procedimiento anterior, hemos elaborado un total de *192 paradigmas flexivos* representados en FST gráficos que incluyen las variantes flexivas de palabras simples, compuestas, términos especializados, topónimos, locativos, abreviaturas, términos en latín, anglicismos y siglas. A través del proceso de vinculación de los diccionarios con los FST gráficos hemos obtenido el diccionario de formas flexionadas que es el que realmente vamos a utilizar en el proceso de reconocimiento de expresiones léxicas.

Hasta ahora, la descripción de los paradigmas flexivos se ha hecho en un sentido general, sin embargo es necesario sistematizar de algún modo cómo se han construido las herramientas léxicas anteriores. Con este objetivo, para desarrollar la flexión de las palabras se han distinguido en un primer momento las clases o categorías generales, como *nombres*, *verbos* y *adjetivos*, según sus propiedades flexivas. Después, dentro de cada una de estas clases se han diferenciado de forma más específica las oposiciones intracategoriales que se establecen entre las unidades del mismo paradigma flexivo. Cada paradigma constituye un sistema cerrado en el que las oposiciones se implantan según un número limitado de categorías léxico-gramaticales –como *género*, *numero*, *aspecto*, *modo* o *tiempo*– y que las mencionadas oposiciones intracategoriales son *recurrentes* y presentan *regularidad*. Siguiendo con esto, en

los apartados siguientes se van a tratar los aspectos formales que se han planteado en la representación de las estructuras de los distintos paradigmas flexivos –básicamente, *paradigma nominal*, *paradigma verbal* y *paradigma adjetival*–, teniendo en cuenta las cuestiones que se enumeran a continuación:

1. Cómo se organizan las distintas formas flexivas de una palabra.
2. Cómo se desarrolla formalmente la flexión, según la presencia o ausencia de las distintas vocales del *stem*, que dará lugar a la adscripción de cada *lema* a las distintas clases flexivas.
3. Cómo se resuelve el problema de las irregularidades en la flexión

5.1.1. Aspectos formales de la flexión nominal

En el paradigma nominal, las oposiciones se organizan en torno a las categorías de `género /número`. Dentro de la categoría léxico-gramatical *género* se configura la oposición `masculino / femenino`, a la que se pueda añadir el género `neutro`, dentro de la categoría léxico-gramatical *número* se configura la oposición `singular / plural`. Las oposiciones anteriores se realizan por medio de los denominados *afijos flexivos*, que se unen al *stem* según la siguiente estructura (Fig. 5.7):

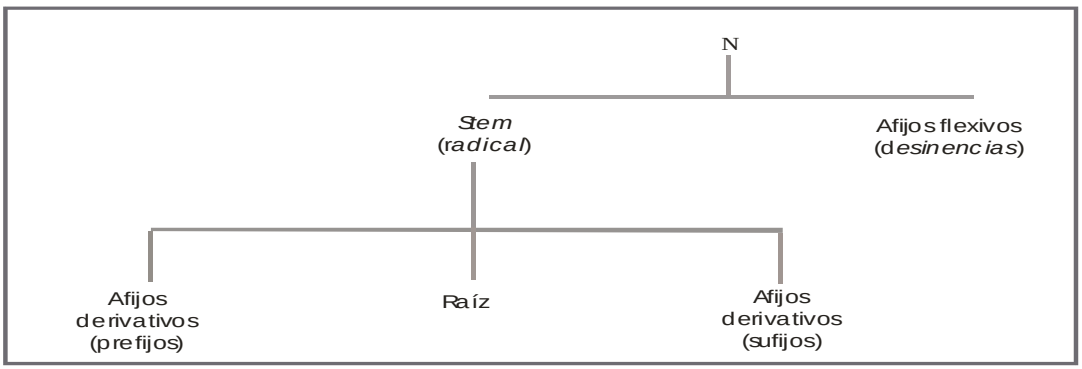


Fig. 5.7: Estructura de los constituyentes nominales

El modo en el que los nombres o substantivos materializan las oposiciones intracategoriales presenta diversos grados de complejidad: la oposición de *número* tiende a recibir una *solución formal única* –como la ausencia o presencia de –(e)s, en las formas singulares o plurales respectivamente–. Frente a esto, la oposición de género tiende a recibir *soluciones múltiples* que están en relación con *factores de diversa naturaleza semántica y formal* (Ambadiang 1994). En principio, a todos los substantivos se les puede asignar sólo uno de los rasgos de *género*, sin embargo a todos los substantivos se les puede asignar cualquiera de los rasgos de *número*. Sistematizar la forma en la que se manifiestan estas oposiciones supondría examinar en profundidad cómo los diversos grupos de substantivos (nombres animados, inanimados, genéricos, simples, derivados o compuestos) se comportan en los procesos flexivos de género /número. Sin embargo, las cuestiones relativas a los procesos flexivos generales, requerirían un tratamiento exhaustivo que no es pertinente realizar en este trabajo.

No obstante, y aunque sólo sea de forma simplificada, en la flexión de género sí se van a establecer distintos conjuntos organizados. La forma en la que el *género* gramatical se presenta en los substantivos sugiere la existencia de diversos subsistemas: *semántico, morfológico* y *fonológico en los nombres de persona, animados, inanimados, simples y derivados* (Ambadiang 1994). A primera vista, la oposición de rasgos masculino / femenino agrupa de un modo regular a substantivos a los que se adjuntan las desinencias -o /-a, o simplemente se adjunta el morfema -a. En otros casos, el *género* sólo es posible que se reconozca por medio de la concordancia que establece con el artículo –como en (el) documentalista / (la) documentalista–. Además, hay casos en los que se producen irregulares basadas en aspectos semánticos –como (el) lunes o (el) domingo–, o en la realidad extralingüística –como en (el) capital / (la) capital–. También existen agrupaciones más o menos extensa basadas en subsistemas morfológicos –como actor / actriz–. En consecuencia, en cada uno de estos subsistemas la flexión de género tiene una motivación y organización diferente y, aunque se indique normalmente por medio de las desinencias respectivas, se comporta sin

leyes claramente identificables, dando lugar a múltiples problemas en el intento por sistematizar la representación.

El problema anterior también se pone de manifiesto en la flexión de género de nombres propios, algunos de ellos compuestos –como `(el) Álgebra de Boole` o `(la) Universidad de Granada`–, en topónimos o locativos –como `(el) Atlántico` o `(los) Pirineos`– y en siglas –como `(la) IBM` o `(el) ISSN`–. En muchos de estos casos, la importancia de la función gramatical del género radica en establecer la concordancia, por medio de la cual los substantivos, simples o compuestos, estructuran todo el sintagma nominal y se encargan de señalar las relaciones con el resto de los elementos de la oración.

En comparación a la flexión de género, la flexión de número se comporta de un modo más uniforme y menos complejo afectando por igual a todas las formas comprendidas en los subsistemas anteriores. La oposición de número `singular / plural`, no muestra problemas en los casos regulares, en el sentido de que el singular se presenta sin marca específica de número, o desinencia cero $(\varnothing)$, y el plural se expresa por medio de los morfemas de plural, o desinencias `-s` o `-es`. Los pocos casos irregulares se deben a aspectos semánticos, en el sentido de que no haya cambios formales, por ser invariables los dos números –como en `hipótesis` o `análisis`–. De algún modo, la representación de los aspectos formales de la pluralización se fundamenta básicamente en torno a dos generalizaciones recogidas por Ambadiang (Ambadiang 1994):

1. El plural de los substantivos acabados en *vocal no acentuada* se establece adjuntando el morfema `-s` a su forma de singular.
2. El plural de los substantivos acabados en *consonante* que no sea `/s/` se establece añadiendo el morfema `-es` al singular, y dicha adjunción provoca a veces un cambio acentual –como en `volumen / volúmenes` o `interconexión /interconexiones`–.

Teniendo en cuenta los aspectos formales anteriores, los substantivos que pertenecen al mismo paradigma flexivo se identifican con el mismo código numérico, como ya se ha descrito, y en el desarrollo de las herramientas de análisis léxico cada una de estas unidades léxicas se vinculada a un FST gráfico que agruparía todas las palabras que se flexionan de la misma forma. Igualmente, las flexiones irregulares se representan en los FST gráficos correspondientes al paradigma flexivo irregular, en consecuencia no intervendría ningún componente basado en *reglas morfológicas*, sino que las irregularidades se representan directamente en FST, como sería el caso del **FST N15** (Fig. 5.8) que agrupa el paradigma flexivo de nombres –como `matriz`, `directriz`, o `luz`– con irregularidades entre formas flexivas y formas canónicas

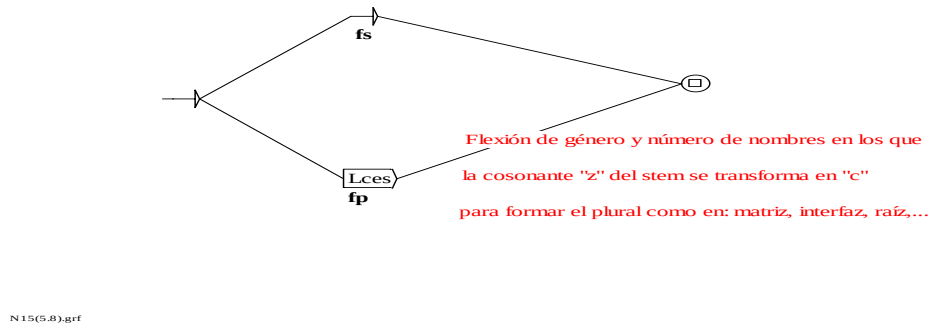


Fig. 5.8: Transductor gráfico N15

Con la técnica descrita arriba se han desarrollado *33 paradigmas de flexión nominal* representados en los siguientes transductores gráficos: **FST N**, **FST N1**, **FST N10**, **FST N101**, **FST N102**, **FST N103**, **FST N11**, **FST N12**, **FST N13**, **FST N14**, **FST N15**, **FST N16**, **FST N2**, **FST N21**, **FST N22**, **FST N23**, **FST N24**, **FST N25**, **FST N26**, **FST N27**, **FST N28**, **FST N29**, **FST N3**, **FST N30**, **FST N31**, **FST N32**, **FST N33**, **FST N4**, **FST N5**, **FST N6**, **FST N7**, **FST N8**, **FST N9**.

5.1.2. Aspectos formales de la flexión adjetival

En relación con los aspectos formales de la flexión adjetival hay que tener en cuenta que el adjetivo funciona normalmente como complemento nominal adjunto, en el sentido de que el número de formas que puede presentar está en relación con las formas que pueda presentar el substantivo. La formalización más simple de la flexión de los adjetivos presenta un mínimo de dos formas en la oposición de número, como `secuencial /secuenciales`, o un máximo de cuatro formas en la oposición de `género /número`, como en `estadístico / estadística` y `estadísticos / estadísticas`. Sin embargo, la complejidad de los adjetivos residiría en señalar las categorías de las que se derivan. Aunque el análisis derivacional no se va a representar en esta aplicación, sí es necesario subrayar la importancia que tiene en la génesis del adjetivo porque incide de algún modo en su flexión. A grandes rasgos, se puede establecer la siguiente clasificación elaborada por Faitelson-Weiser (Faitelson-Weiser 1993):

1. Adje*tivos formados a partir de verbos*, subdivididos en: *a) activos*, como `buscador` de `buscar`; *b)* pasivos, como `automatizado` de `automatizar` o `concerniente` de `concernir`
2. *Adjetivos formados a partir de nombres*, subdivididos en: *a)* de relación con el nombre del que se derivan, como `multivariante` de `multivarianza`; *b)* de relación con el lugar del nombre, o gentilicios, como `complutense`, `inglés`, `alemán`, `...`; *c)* de relación con la persona, como `chomskyano`; *d)* de posesión, o que expresa la cualidad del nombre, como `interesente` que posee `interés`.
3. *Adjetivos formados a partir de numerales*, subdivididos en: *a)* ordinales, como `undécimo`; *b)* partitivos, como `octavo`.

En la aplicación que vamos a realizar los subconjuntos que se han aislado para tratar la flexión adjetival están en relación con la forma en la se adjuntan las desinencias, y no con las categorías de las que se derivan. De este modo, se reúnen en el mismo paradigma aquellos

adjetivos cuya flexión de género se produce simplemente añadiendo `-o /-a` y cuya flexión de número se produce añadiendo `-s /-es`. Además de incorporar otros grupos con las distintas irregularidades. Con la única excepción de que los adjetivos pasivos derivados de verbos se tratan como una categoría diferencia con el código `PA (Participio-Adjetivo)` y lo mismo ocurre con los adjetivos derivados de un *numeral* que no se incluyen dentro de la categoría general de *Adjetivo* sino con el código `ORD (Ordinal)`.

Para formalizar todas las variantes flexionadas se han elaborado *27 paradigmas de flexión adjetival* representados en FST gráficos, en los que se agrupan distintas clases de conjuntos formados por adjetivos posesivos, ordinales, cuantificadores no-numerales, participios en su función de adjetivos, así como adjetivos interrogativos. Los transductores gráficos de flexión adjetival construidos son los siguientes: **FST A**, **FST A1**, **FST A2**, **FST A3**, **FST A4**, **FST A5**, **FST A6**, **FST A7**, **FST A8**, **FST A9**, **FST AIN**, **FST ARE**, **FST ARE1**, **FST CARD**, **FST CUANT**, **FST CUANT1**, **FST CUANT2**, **FST CUANT3**, **FST CUANT4**, **FST ORD1**, **FST ORD2**, **FST PA**, **FST POS1**, **FST POS2**, **FST POS3**, **FST POS4**, **FST POS5**.

5.1.3. Aspectos formales de la flexión verbal

La unidad lingüística por excelencia en la que actúa el paradigma flexivo es el verbo, por esta razón son muchos los estudios dedicados a esta cuestión (Matthews 1974; Harris 1987; Alcoba 1987; Ambadiang 1990; Mighetto 1992; Ambadiang 1993). Los verbos tienen una *estructura binaria* constituida por *stem* y *formas flexivas*. Un elemento fundamental en esta organización es la vocal de *stem* que determina la clasificación del verbo en los distintos tipos de conjugaciones. La vocal del *stem* `(-a, -e, -i)` varía según la conjugación de cada verbo `(1ª, 2ª y 3ª)`, y esta vocal se transforma en las terminaciones o desinencias según los distintos valores que toman las formas flexivas de `Tiempo / Modo / Aspecto` y

`Número /Persona`. La estructura de los constituyentes verbales anteriores se puede representar, según el modelo de Harris (Harris 1987), del modo siguiente (Fig. 5.9):

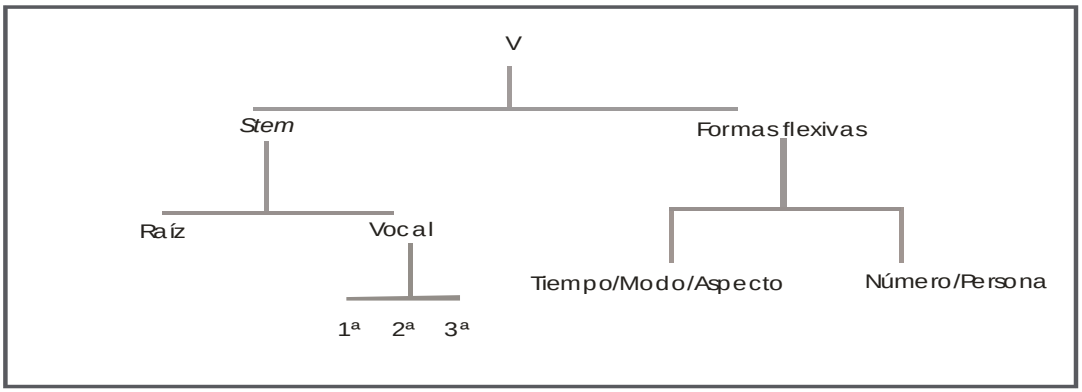


Fig. 5.9: Estructura de los constituyentes verbales

Las variantes de las formas flexivas `Tiempo / Modo / Aspecto` determinan los distintos significados gramaticales de las formas verbales. La categoría gramatical `Tiempo` expresa una relación entre el momento temporal del discurso, o enunciación, y de la acción expresada en el discurso (*simultaneidad*, *anterioridad*, *posterioridad*), dentro de esta categoría entraría el `Presente, Pasado, Futuro, Condicional`. La categoría gramatical `Modo` indica el grado de validez o de certeza de la expresión verbal por parte del hablante, la flexión dentro del paradigma se limita a tres constituyentes: `Indicativo, Subjuntivo, Imperativo`. La categoría gramatical `Aspecto` expresa el carácter de la acción de cada tiempo respecto a su desarrollo o resultado, se trata de una categoría binaria constituida por los términos: `Perfecto, Imperfecto`. En lo referente al constituyente `Número / Persona` está formado por las formas flexivas de número `singular, plural` y las formas flexivas de persona `1ª, 2ª, 3ª`.

Todos las formas flexionadas del verbo se adscriben a las múltiples combinaciones que permiten los constituyentes flexivos de `Tiempo / Modo / Aspecto` y `Número / Persona`, excepto las formas no personales (*Infinitivo*, *Gerundio* y *Participio*). El problema está en que en las formas flexionadas de las respectivas conjugaciones se

producen variantes e irregularidades en las combinaciones. La identificación, y posterior representación, de las alteraciones en la flexión verbal es bastante laboriosa porque muchas veces requiere considerar individualmente la conjugación particular de cada verbo.

El procedimiento que se ha seguido en un principio es clasificar todos los verbos en los tres grandes grupos tradicionales –o clases de verbos según la 1ª, 2ª y 3ª conjugación– teniendo en cuenta la vocal del *stem*. Después, dentro de cada grupo se distinguen los que pertenecen al paradigma regular y los que, por el contrario, presentan irregularidades en la flexión. Se consideran verbos regulares aquellos en los que el constituyente radical o *stem*, formado por raíz y vocal, permanecen invariable en todas las formas flexionadas de la conjugación (Tabla 5.1).

TABLA 5.1: Sufijos de flexión y etiquetas *part-of-speech* (POS) de los verbos regulares

1ª Conjugación		**2ª Conjugación**		**3ª Conjugación**	
ar.V1:W	ariais.V1:C2p	er.V2:W	eriais.V1:C2p	ir.V3:W	iriais.V3:C2p
ado.V1:P	arian.V1:C3p	ido.V2:P	erian.V2:C3p	ido.V3:P	irian.V3:C3p
ando.V1:G	e.V1:S1s	iendo.V2:G	a.V2:S1s	iendo.V3:G	a.V3:S1s
o.V1:P1s	Es.V1:S2s	o.V2:P1s	as.V2:S2s	o.V3:P1s	as.V3:S2s
as.V1:P2s	e.V1:S3s	es.V2:P2s	a.V2:S3s	es.V3:P2s	a.V3:S3s
a.V1:P3s	emos.V1:S1p	e.V2:P3s	amos.V2:S1p	e.V3:P3s	amos.V3:S1p
amos.V1:P1p	eis.V1:S2p	emos.V2:P1p	ais.V2:S2p	imos.V3:P1p	ais.V3:S2p
ais.V1:P2p	en.V1:S3p	eis.V2:P2p	an.V2:S3p	is.V3:P2p	an.V3:S3p
an.V1:P3p	ara.V1:IS1s	en.V2:P3p	iera.V2:IS1s	en.V3:P3p	iera.V3:IS1s
aba.V1:I1s	ase.V1:IS1s	ia.V2:I1s	iese.V2:IS1s	ia.V3:I1s	iese.V3:IS1s
abas.V1:I2s	aras.V1:IS2s	ias.V2:I2s	ieras.V2:IS2s	ias.V3:I2s	ieras.V3:IS2s
aba.V1:I3s	ases.V1:IS2s	ia.V2:I3s	ieses.V2:IS2s	ia.V3:I3s	ieses.V3:IS2s
abamos.V1:I1p	ara.V1:IS3s	iamos.V2:I1p	iera.V2:IS3s	iamos.V3:I1p	iera.V3:IS3s
abais.V1:I2p	ase.V1:IS3s	iais.V2:I2p	iese.V2:IS3s	iais.V3:I2p	iese.V3:IS3s
aban.V1:I3p	aramos.V1:IS1p	ian.V2:I3p	ieramos.V2:IS1p	ian.V3:I3p	ieramos.V3:IS1p
e.V1:J1s	asemos.V1:IS1p	i.V2:J1s	iesemos.V2:IS1p	i.V3:J1s	iesemos.V3:IS1p
aste.V1:J2s	arias.V1:IS2p	iste.V2:J2s	ierais.V2:IS2p	iste.V3:J2s	ierais.V3:IS2p
o.V1:J3s	aseis.V1:IS2p	io.V2:J3s	ieseis.V2:IS2p	io.V3:J3s	ieseis.V3:IS2p
amos.V1:J1p	aran.V1:IS3p	imos.V2:J1p	ieran.V2:IS3p	imos.V3:J1p	ieran.V3:IS3p
asteis.V1:J2p	asen.V1:IS3p	isteis.V2:J2p	iesen.V2:IS3p	isteis.V3:J2p	iesen.V3:IS3p
aron.V1:J3p	are.V1:FS1s	ieron.V2:J3p	iere.V2:FS1s	ieron.V3:J3p	iere.V3:FS1s
are.V1:F1s	ares.V1:FS2s	ere.V2:F1s	ieres.V2:FS2s	ire.V3:F1s	ieres.V3:FS2s
aras.V1:F2s	are.V1:FS3s	eras.V2:F2s	iere.V2:FS3s	iras.V3:F2s	iere.V3:FS3s
ara.V1:F3s	aremos.V1:FS1p	era.V2:F3s	ieramos.V2:FS1p	ira.V3:F3s	ieramos.V3:FS1p
aremos.V1:F1p	areis.V1:FS2p	eremos.V2:F1p	iereis.V2:FS2p	iremos.V3:F1p	iereis.V3:FS2p
areis.V1:F2p	aren.V1:FS3p	ereis.V2:F2p	ieren.V2:FS3p	ireis.V3:F2p	ieren.V3:FS3p
aran.V1:F3p	a.V1:Y2s	eran.V2:F3p	e.V2:Y2s	iran.V3:F3p	e.V3:Y2s
aria.V1:C1s	e.V1:Y3s	eria.V2:C1s	a.V2:Y3s	iria.V3:C1s	a.V3:Y3s
arias.V1:C2s	emos.V1:Y1p	erias.V2:C2s	amos.V2:Y1p	irias.V3:C2s	amos.V3:Y1p
Aria.V1:C3s	ad.V1:Y2p	eria.V2:C3s	ed.V2:Y2p	iria.V3:C3s	id.V3:Y2p
ariamos.V1:C1p	en.V1:Y3p	eriamos.V2:C1p	an.V2:Y3p	iriamos.V3:C1p	an.V3:Y3p

Por el contrario, los verbos irregulares son aquellos en los que el constituyente radical se altera total o parcialmente en las formas flexionadas de la conjugación. En las formas irregulares tiene mucha importancia la distribución del acento, que provoca variantes en algunos tiempos de la conjugación, entre las más usuales:

- Transformación de la vocal del *stem* cuando ésta se acentúa, como en las formas flexionadas `cierro, cuento, juego, pierdo, ...` de verbos del tipo: `cerrar, contar, jugar, perder, mover, sentir, ...`
- Transformación de una consonante por otra, como en las formas flexionadas `conozco, dirijo, reduzco,traduzco, ...` de verbos del tipo: `conocer, dirigir, reducir, traducir, ...`
- Irregularidades sin ninguna pauta identificable, como en las formas flexionadas `doy, soy, veo, traigo, voy, ...` de verbos del tipo: `dar, ser, ver, traer, ir, ...`

Como se puede observar la complicación está en la representación de las flexiones irregulares porque muchas veces requieren una estudio caso por caso. El procedimiento que hemos seguido para formalizar las flexiones regulares e irregulares ha consistido en agrupar todas las conjugaciones regulares en los tres grandes paradigmas de flexión regular y dentro de ellos, hemos identificado las distintas clases de los paradigmas de flexión irregular. Con este planteamiento se han desarrollado un total de *99 paradigmas de flexión verbal* clasificados en:

1) Paradigma de *flexión regular de la 1ª conjugación*: **FST V1**.
   - Paradigmas de *flexión irregular de la 1ª conjugación*: **FST V10**, **FST V101**, **FST V102**, **FST V103**, **FST V104**, **FST V105**, **FST V106**, **FST V107**, **FST V108**, **FST V109**, **FST V11**, **FST V110**, **FST V111**, **FST V112**, **FST V113**, **FST V114**, **FST V115**, **FST V116**, **FST V117**, **FST V118, FST V119, FST V12**, **FST V120**, **FST V121**, **FST V122**, **FST V123**, **FST V124**, **FST V125**,

FST V126, **FST V127**, **FST V128**, **FST V129**, **FST V13**, **FST V14**, **FST V15**, **FST V16**, **FST V17**, **FST V18**, **FST V19**.

2) Paradigma de *flexión regular de la 2ª conjugación*: **FST V2**.
   - Paradigmas de *flexión irregular de la 2ª conjugación*: **FST V20**, **FST V201**, **FST V202**, **FST V203**, **FST V204**, **FST V205**, **FST V206**, **FST V207**, **FST V208**, **FST V209**, **FST V21**, **FST V210**, **FST V211**, **FST V212**, **FST V213**, **FST V22**, **FST V23**, **FST V24**, **FST V25**, **FST V26**, **FST V27**, **FST V28**, **FST V29**.

3) Paradigma de *flexión regular de la 3ª conjugación*: **FST V3**.
   - Paradigmas de *flexión irregular de la 3ª conjugación*: **FST V30**, **FST V301**, **FST V302**, **FST V303**, **FST V304**, **FST V305**, **FST V306**, **FST V307**, **FST V308**, **FST V309**, **FST V31**, **FST V310**, **FST V311**, **FST V312**, **FST V313**, **FST V314**, **FST V315**, **FST V316**, **FST V317**, **FST V318**, **FST V319**, **FST V32**, **FST V320**, **FST V321**, **FST V322**, **FST V323**, **FST V324**, **FST V33**, **FST V34**, **FST V35**, **FST V36**, **FST V37**, **FST V38**, **FST V39**.

5.2. Construcción de Transductores Léxicos

Otra herramienta de reconocimiento léxico consiste en el desarrollo de FST léxicos. Los *Transductores Léxicos* tienen como función representar unidades lingüísticas, al igual que los diccionarios, pero lo hacen de forma *gráfica*. Esta características los hace especialmente adecuados para determinadas aplicaciones como agrupar las formas derivadas de un término o relacionar expresiones sinónimas. Además, el lenguaje producido por los *Transductores Léxicos* se puede utilizar en un *IRS para generar un índice* del conjunto de palabras representadas y a modo de *query para localizar en los textos todas las variantes de un término* (Silberztein 1999). La aplicación práctica más importantes de estos transductores es agrupar

todas las variantes de un término, pero además de forma detallada se pueden destacar las siguientes funciones:

a) *Reconocer expresiones léxicas y relacionarlas con sus respectivas formas canónicas.* Se trata de la función más sencilla consistente en desarrollar un FST gráfico que reconozca unidades lingüísticas en los textos y las vincule con sus respectivos *lemas*, de forma semejante a como lo hacen los diccionarios (Fig. 5.10).

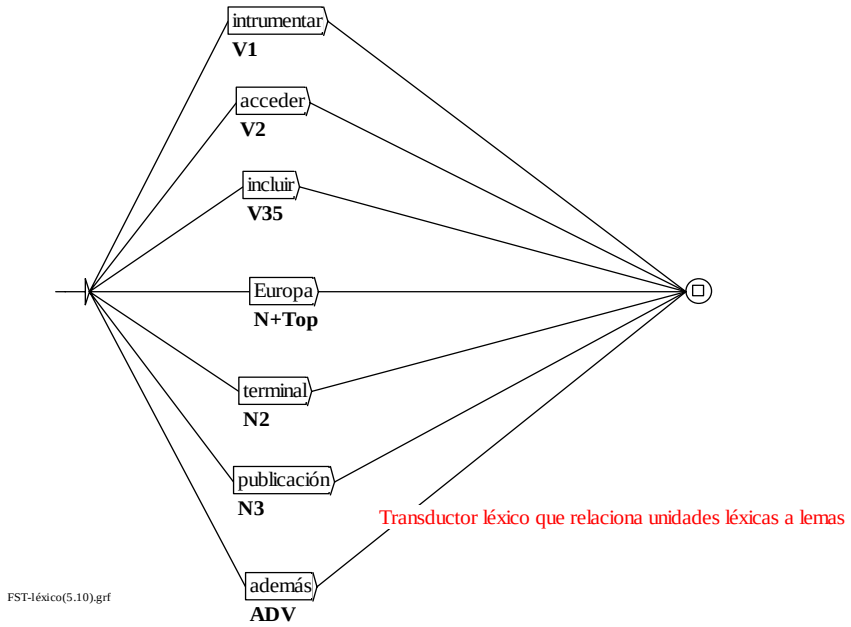


Fig. 5.10: Representación de unidades lingüísticas en un FST léxico

Este transductor se puede considerar un *gráfico de búsqueda* para localizar determinadas formas canónicas en los textos, a su vez se puede utilizar para generar un conjunto de unidades léxicas semejantes a un diccionario de formas canónicas, que posteriormente se vinculan a los FST de flexión:

acceder,**V2**
intrumentar,**V1**

terminal,**N2**

incluir,**V35**

publicación,**N3**

Europa,**N+Top**

además,**ADV**

b) *Reconocer y agrupar formas derivadas de una unidad léxica*. Los diccionarios proporcionan las herramientas necesarias para representar las formas flexionadas de una palabra, pero, como ya se mencionó, no son adecuados para representar las formas derivadas. Frente a esto, los transductores gráficos representan eficazmente las *formas derivadas* de una unidad léxica. Un FST gráfico puede relacionar formal y semánticamente un grupo de palabras mediante una representación gráfica (Fig. 5.11).

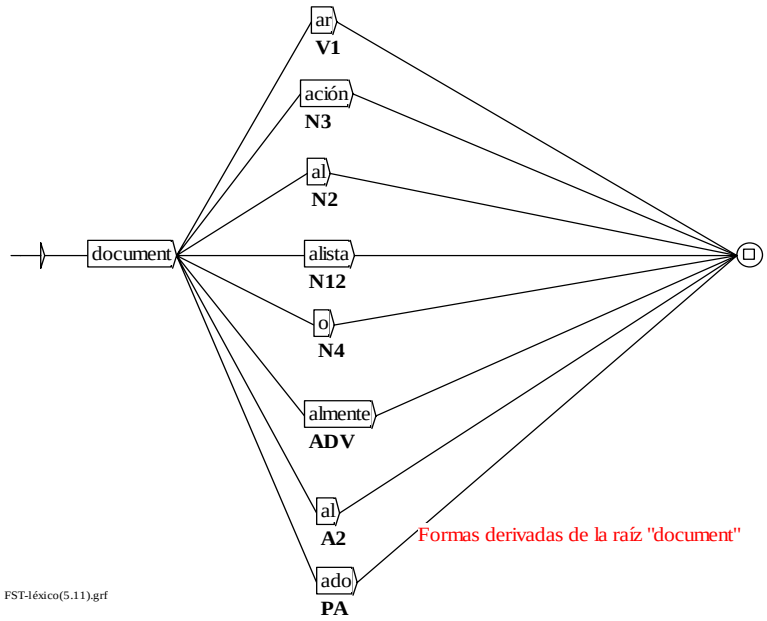


Fig. 5.11: Representación de unidades derivadas en un FST léxico

Además, el FST gráfico anterior no sólo se encarga de agrupar todas las formas derivadas de una palabra y asignarles los correspondientes códigos flexionales, sino que se puede emplear en un IRS con el objetivo de indizar de forma conjunta todas las apariciones de las distintas unidades derivadas. De este modo, el lenguaje generado por el FST léxico anterior sería el siguiente:

documentalmente,**ADV**
documentación,**N3**
documentalista,**N12**
documentado,**PA**
documentar,**V1**
documento,**N4**
documental,**N2**
documental,**A2**

El conjunto de unidades lingüísticas derivadas se representa a modo de un diccionario o léxico de *lemas*, que después de vincularlo a los FST de flexión se obtendría un conjunto de formas flexionadas conectadas semánticamente por la misma *raíz*. El resultado de este proceso se puede utilizar para generar un índice de los términos relacionados, que se podrán identificar posteriormente en los textos.

c) *Reconocer y agrupar las variantes ortográficas de un término*. La función de estos transductores es reemplazar una familia de variantes ortográficas por una *ortografía canónica* (Fig. 5.12). Después de la traducción de una serie de términos por una forma canónica, todos las variantes se indizarían por el mismo término en un IRS.

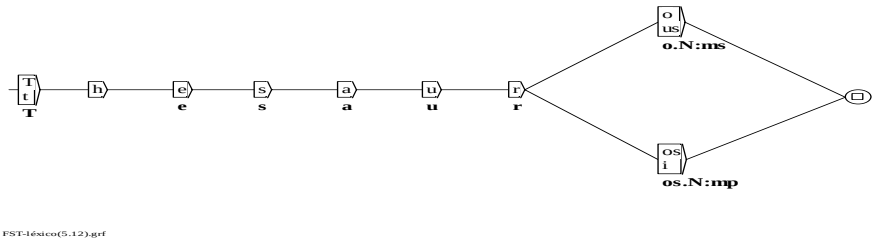


Fig. 5.12: Representación de las variantes de un término en un FST léxico

Las equivalencias que realiza el transductor anterior serían las siguientes:

Thesauro,*Tesauro*.**N:ms**
Thesauri,*Tesauros*.**N:mp**
Thesauros,*Tesauros*.**N:mp**
Thesaurus,*Tesauro*.**N:ms**
thesauro,*Tesauro*.**N:ms**
thesauri,*Tesauros*.**N:mp**
thesauros,*Tesauros*.**N:mp**
thesaurus,*Tesauro*.**N:ms**

En este trabajo vamos a proponer otras funciones que son de gran utilidad no sólo para agrupar las variantes de un término sino para el control y la normalización de las entradas a un índice:

a) *Reconocer y agrupar determinados subsistemas lingüísticos.* Mucha veces construir diccionarios que agrupen todos los elementos de un subconjunto del sistema de una lengua puede resultar una tarea laboriosa, como sucede con todas las variantes de un *Nombre Personal.* Los FST gráficos permiten representar de forma eficaz todos los elementos pertenecientes a una subclase del sistema lingüístico, como es el caso de las variantes de los nombres personales (Fig. 5.13). Igualmente, los FST gráficos

obtenidos se puede utilizar tanto para hacer búsquedas en los textos, como para sustituir todas las variantes por la forma canónica o normalizada. Además, nos permitirá agregar cualquier otro tipo de información especializada como *área de trabajo, afiliación, universidad, grupo de investigación,...*al que pertenece el nombre, o los nombres representados.

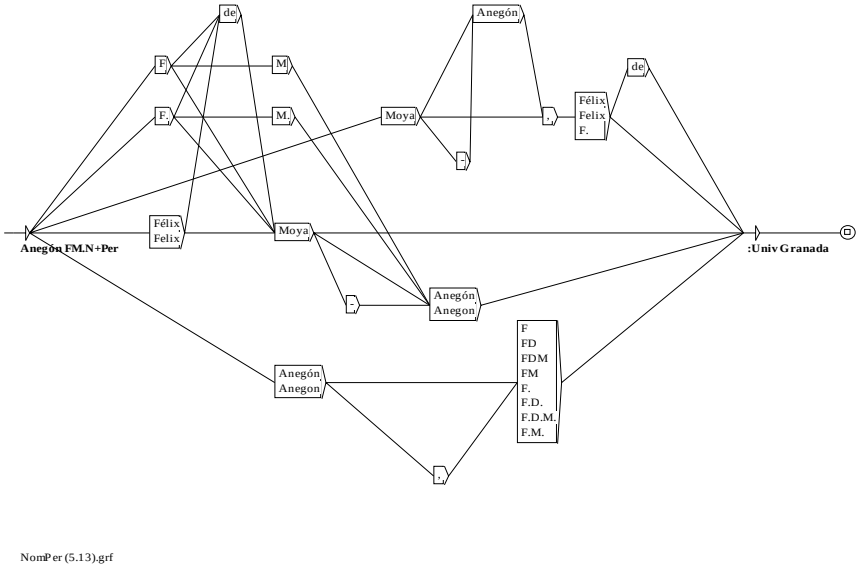


Fig. 5.13: Agrupación de las variantes de un *Nombre Personal* en un FST léxico

El FST anterior genera, reconoce y agrupa *94 variantes de un Nombre Personal*:

Anegon, F,*Anegón FM*.**N+Per:Univ Granada**

Anegón, F,*Anegón FM*.**N+Per:Univ Granada**

Anegon, F.,*Anegón FM*.**N+Per:Univ Granada**

Anegón, F.,*Anegón FM*.**N+Per:Univ Granada**

Anegon, F. D.,*Anegón FM*.**N+Per:Univ Granada**

Anegón, F. D.,*Anegón FM*.**N+Per:Univ Granada**

Anegon, F. D. M.,*Anegón FM*.**N+Per:Univ Granada**

Anegón, F. D. M.,*Anegón FM*.**N+Per:Univ Granada**

Anegon, F. M.,*Anegón FM*.**N+Per:Univ Granada**

Anegón, F. M.,*Anegón FM*.**N+Per:Univ Granada**

Anegon, FD,*Anegón FM*.**N+Per:Univ Granada**

Anegón, FD,*Anegón FM*.**N+Per:Univ Granada**

Anegon, FM,*Anegón FM*.**N+Per:Univ Granada**

Anegón, FM,*Anegón FM*.**N+Per:Univ Granada**

Anegon, FM,*Anegón FM*.**N+Per:Univ Granada**

Anegón, FM,*Anegón FM*.**N+Per:Univ Granada**

Anegon F,*Anegón FM*.**N+Per:Univ Granada**

Anegón F,*Anegón FM*.**N+Per:Univ Granada**

Anegon F.,*Anegón FM*.**N+Per:Univ Granada**

Anegón F.,*Anegón FM*.**N+Per:Univ Granada**

Anegon F. D.,*Anegón FM*.**N+Per:Univ Granada**

Anegón F. D.,*Anegón FM*.**N+Per:Univ Granada**

Anegon F. D. M.,*Anegón FM*.**N+Per:Univ Granada**

Anegón F. D. M.,*Anegón FM*.**N+Per:Univ Granada**

Anegon F. M.,*Anegón FM*.**N+Per:Univ Granada**

Anegón F. M.,*Anegón FM*.**N+Per:Univ Granada**

Anegon FD,*Anegón FM*.**N+Per:Univ Granada**

Anegón FD,*Anegón FM*.**N+Per:Univ Granada**

Anegon FM,*Anegón FM*.**N+Per:Univ Granada**

Anegón FM,*Anegón FM*.**N+Per:Univ Granada**

Anegon FM,*Anegón FM*.**N+Per:Univ Granada**

Anegón FM,*Anegón FM*.**N+Per:Univ Granada**

F. de Moya-Anegon,*Anegón FM*.**N+Per:Univ Granada**

F. de Moya-Anegón,*Anegón FM*.**N+Per:Univ Granada**

F. de Moya,*Anegón FM*.**N+Per:Univ Granada**

F. de Moya Anegon,*Anegón FM*.**N+Per:Univ Granada**

F. de Moya Anegón,*Anegón FM*.**N+Per:Univ Granada**

F. M. Anegon,*Anegón FM*.**N+Per:Univ Granada**

F. M. Anegón,*Anegón FM*.**N+Per:Univ Granada**

F. Moya-Anegon,*Anegón FM*.**N+Per:Univ Granada**

F. Moya-Anegón,*Anegón FM*.**N+Per:Univ Granada**

F. Moya,*Anegón FM*.**N+Per:Univ Granada**

F. Moya Anegon,*Anegón FM*.**N+Per:Univ Granada**

F . Moya Anegón,*Anegón FM*.**N+Per:Univ Granada**

F de Moya-Anegon,*Anegón FM*.**N+Per:Univ Granada**

F de Moya-Anegón,*Anegón FM*.**N+Per:Univ Granada**

F de Moya,*Anegón FM*.**N+Per:Univ Granada**

F de Moya Anegon,*Anegón FM*.**N+Per:Univ Granada**

F de Moya Anegón,*Anegón FM*.**N+Per:Univ Granada**

F M Anegon,*Anegón FM*.**N+Per:Univ Granada**

F M Anegón,*Anegón FM*.**N+Per:Univ Granada**

F Moya-Anegon,*Anegón FM*.**N+Per:Univ Granada**

F Moya-Anegón,*Anegón FM*.**N+Per:Univ Granada**

F Moya,*Anegón FM*.**N+Per:Univ Granada**

F Moya Anegon,*Anegón FM*.**N+Per:Univ Granada**

F Moya Anegón,*Anegón FM*.**N+Per:Univ Granada**

Felix de Moya-Anegon,*Anegón FM*.**N+Per:Univ Granada**

Felix de Moya-Anegón,*Anegón FM*.**N+Per:Univ Granada**

Félix de Moya-Anegon,*Anegón FM*.**N+Per:Univ Granada**

Félix de Moya-Anegón,*Anegón FM*.**N+Per:Univ Granada**

Felix de Moya,*Anegón FM*.**N+Per:Univ Granada**

Félix de Moya,*Anegón FM*.**N+Per:Univ Granada**

Felix de Moya Anegon,*Anegón FM*.**N+Per:Univ Granada**

Felix de Moya Anegón,*Anegón FM*.**N+Per:Univ Granada**

Félix de Moya Anegon,*Anegón FM*.**N+Per:Univ Granada**

Félix de Moya Anegón,*Anegón FM*.**N+Per:Univ Granada**

Felix Moya-Anegon,*Anegón FM*.**N+Per:Univ Granada**

Felix Moya-Anegón,*Anegón FM*.**N+Per:Univ Granada**

Félix Moya-Anegon,*Anegón FM*.**N+Per:Univ Granada**

Félix Moya-Anegón,*Anegón FM*.**N+Per:Univ Granada**

Felix Moya,*Anegón FM*.**N+Per:Univ Granada**

Félix Moya,*Anegón FM*.**N+Per:Univ Granada**

Felix Moya Anegon,*Anegón FM*.**N+Per:Univ Granada**

Felix Moya Anegón,*Anegón FM*.**N+Per:Univ Granada**

Félix Moya Anegon,*Anegón FM*.**N+Per:Univ Granada**

Félix Moya Anegón,*Anegón FM*.**N+Per:Univ Granada**

Moya-Anegón, F.,*Anegón FM*.**N+Per:Univ Granada**

Moya-Anegón, F. de,*Anegón FM*.**N+Per:Univ Granada**

Moya-Anegón, Felix,*Anegón FM*.N+**Per:Univ Granada**

Moya-Anegón, Félix,*Anegón FM*.**N+Per:Univ Granada**
Moya-Anegón, Felix de,*Anegón FM*.**N+Per:Univ Granada**
Moya-Anegón, Félix de,*Anegón FM*.**N+Per:Univ Granada**
Moya, F. ,*Anegón FM*.**N+Per:Univ Granada**
Moya, F. de,*Anegón FM*.**N+Per:Univ Granada**
Moya, Felix,*Anegón FM*.**N+Per:Univ Granada**
Moya, Félix,*Anegón FM*.**N+Per:Univ Granada**
Moya, Felix de,*Anegón FM*.**N+Per:Univ Granada**
Moya, Félix de,*Anegón FM*.**N+Per:Univ Granada**
Moya Anegón, F. ,*Anegón FM*.**N+Per:Univ Granada**
Moya Anegón, F. de,*Anegón FM*.**N+Per:Univ Granada**
Moya Anegón, Felix,*Anegón FM*.**N+Per:Univ Granada**
Moya Anegón, Félix,*Anegón FM*.**N+Per:Univ Granada**
Moya Anegón, Felix de,*Anegón FM*.**N+Per:Univ Granada**
Moya Anegón, Félix de,*Anegón FM*.**N+Per:Univ Granada**

b) *Reconocer y agrupar expresiones vinculadas semánticamente con una sola expresión canónica*. Los FST léxicos se pueden emplear para relacionar expresiones similares con una única expresión que se implanta en este caso como forma aceptada (Fig. 5.14). De este modo, los transductores gráficos se utilizan como un mecanismo que asocia expresiones equivalentes, que coinciden total o parcialmente en su significado. Las equivalencias no tienen por qué ser completas y se pueden establecer a partir de objetivos específicos en los que se considere pertinente agrupar determinadas construcciones lingüísticas.

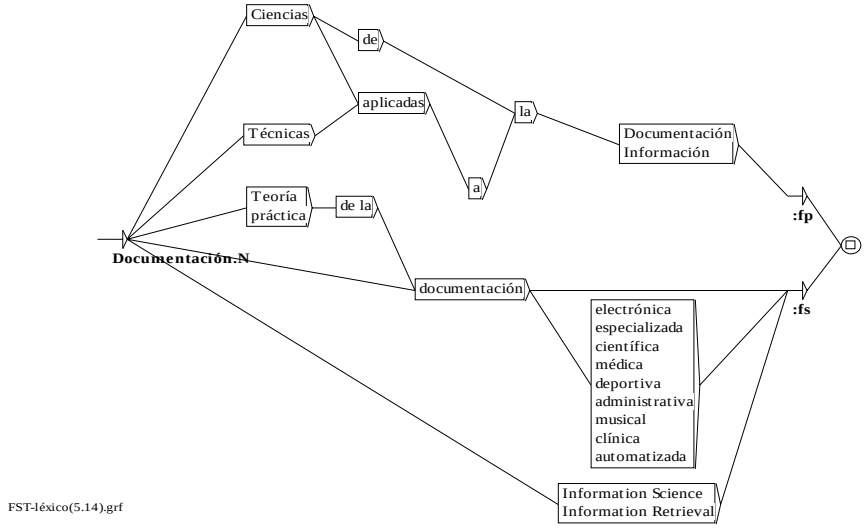


Fig. 5.14: Agrupación de expresiones consideradas sinónimas en un transductor gráfico

El lenguaje que genera y reconoce el FST léxico anterior es el siguiente:

documentación,*Documentación*.**N:fs**

Information Science,*Documentación*.**N:fs**

Information Retrieval,*Documentación*.**N:fs**

documentación electrónica,*Documentación*.**N:fs**

documentación clínica,*Documentación*.**N:fs**

documentación automatizada,*Documentación*.**N:fs**

documentación especializada,*Documentación*.**N:fs**

documentación administrativa,*Documentación*.**N:fs**

documentación deportiva,*Documentación*.**N:fs**

documentación científica,*Documentación*.**N:fs**

documentación musical,*Documentación*.**N:fs**

documentación médica,*Documentación*.**N:fs**

Ciencias de la Documentación,*Documentación*.**N:fp**

Ciencias de la Información,*Documentación*.**N:fp**

Teoría de la documentación,*Documentación*.**N:fs**

práctica de la documentación,*Documentación*.**N:fs**

práctica de la documentación electrónica,*Documentación*.**N:fs**

práctica de la documentación clínica,*Documentación*.**N:fs**
práctica de la documentación automatizada,*Documentación*.**N:fs**
práctica de la documentación especializada,*Documentación*.**N:fs**
práctica de la documentación administrativa,*Documentación*.**N:fs**
práctica de la documentación deportiva,*Documentación*.**N:fs**
práctica de la documentación científica,*Documentación*.**N:fs**
práctica de la documentación musical,*Documentación*.**N:fs**
práctica de la documentación médica,*Documentación*.**N:fs**
Técnicas aplicadas a la Documentación,*Documentación*.**N:fp**
Técnicas aplicadas a la Información,*Documentación*.**N:fp**
Teoría de la documentación electrónica,*Documentación*.**N:fs**
Teoría de la documentación clínica,*Documentación*.**N:fs**
Teoría de la documentación automatizada,*Documentación*.**N:fs**
Teoría de la documentación especializada,*Documentación*.**N:fs**
Teoría de la documentación administrativa,*Documentación*.**N:fs**
Teoría de la documentación deportiva,*Documentación*.**N:fs**
Teoría de la documentación científica,Documentación.**N:fs**
Teoría de la documentación musical,*Documentación*.**N:fs**
Teoría de la documentación médica,*Documentación*.**N:fs**
Ciencias aplicadas a la Documentación,*Documentación*.**N:fp**
Ciencias aplicadas a la Información,*Documentación*.**N:fp**

De la misma forma, las agrupaciones de expresiones se pueden establecer a partir de las relaciones de equivalencia extraídas de un Tesauro, en este caso los términos equivalentes remitirían al término preferente, o *descriptor* (Fig. 5.15)

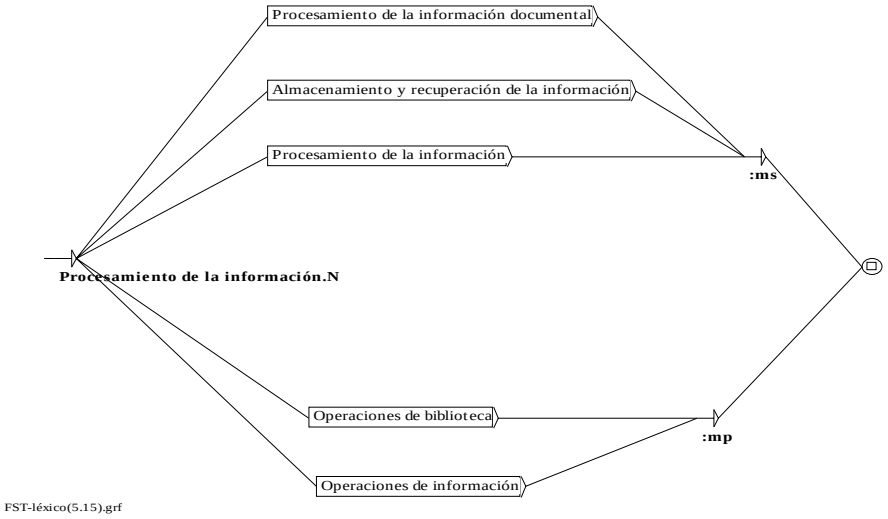


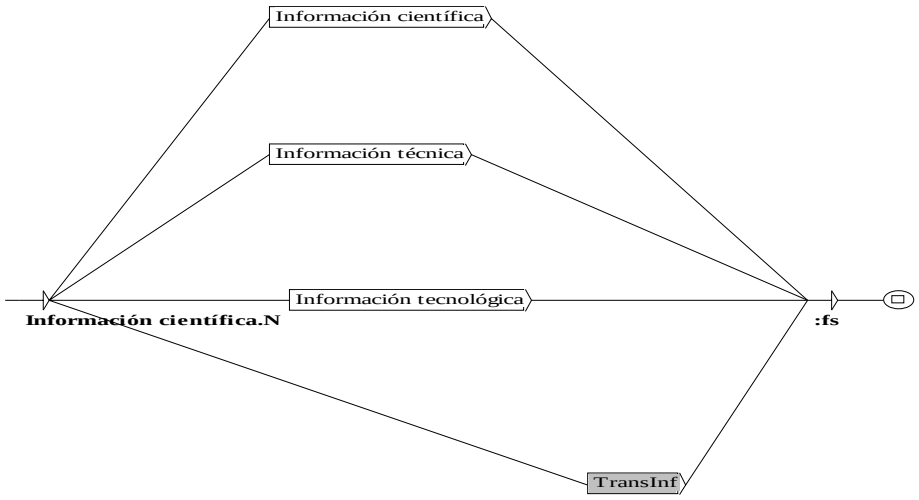
Fig. 5.15: Agrupación de términos equivalentes en un transductor gráfico

Las relaciones de equivalencia, que genera y reconoce el transductor gráfico anterior, son las siguientes:

Operaciones de biblioteca,*Procesamiento de la información*.**N:mp**
Operaciones de información,Procesamiento de la información.**N:mp**
Procesamiento de la información,*Procesamiento de la información*.**N:ms**
Procesamiento de la información documental,*Procesamiento de la información*.**N:ms**
Almacenamiento y recuperación de la información,*Procesamiento de la información*.**N:ms**

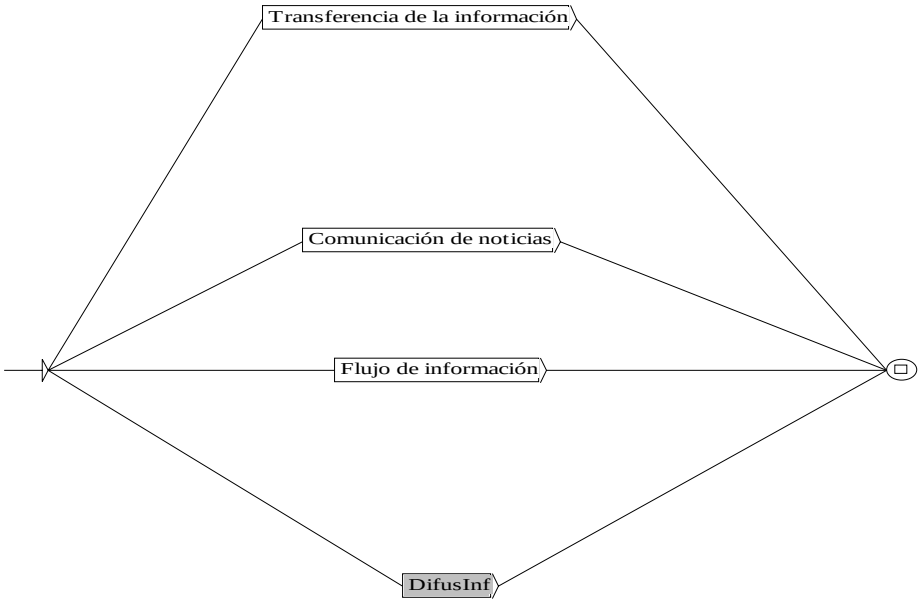
c) *Reconocer y reconocer determinadas expresiones fijas o expresiones de dominio*. Con objetivo de buscar en los textos un grupo de expresiones de dominio y vincularlas a una sola expresión canónica se pueden diseñar transductores gráficos imbricados o vinculados. Dependiendo de las relaciones semánticas que representen, los transductores son más o menos complejos. Los FST gráficos que se muestran a continuación, representan las relaciones semánticas vinculadas al dominio de la *difusión y transferencia de la información científica* y están seleccionados a partir de los descriptores de un tesauro. El primer transductor vincula distintas construcciones

a la expresión de dominio *Información científica* (Fig. 5.16), este transductor tiene imbricado otro transductor que relaciona expresiones sinónimas a la expresión *Transferencia de la Información*, TransInf (Fig. 5.17). A su vez, este transductor tiene imbricado otro transductor en el que se incluyen las expresiones sinónimas correspondientes al dominio *Difusión de la Información*, DifusInf (Fig. 5.18).



InforCien (5.16).grf

Fig. 5.16: Representación de la expresión «*Información científica*» en un FST gráfico



TransInf.grf

Fig. 5.17: Representación de la expresión «*Transferencia de la información*» en un FST gráfico

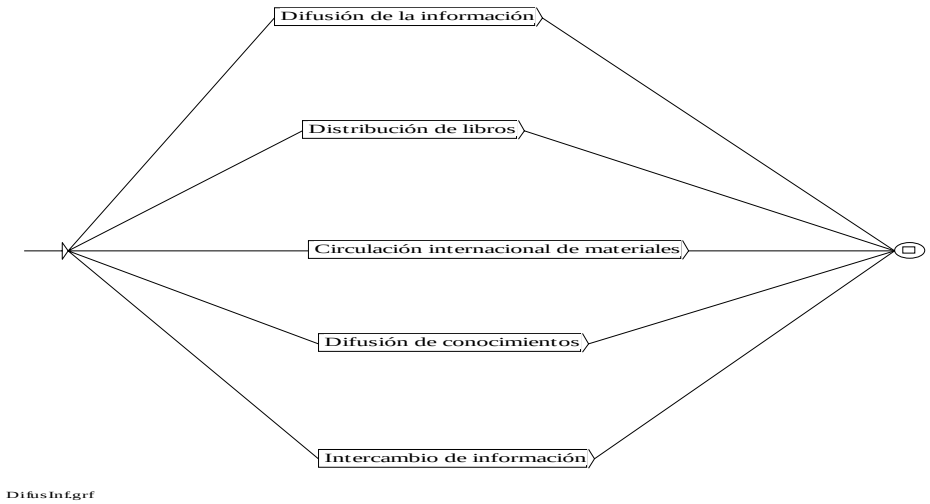


Fig.5.18: Representación de la expresión «*Difusión de la información*» en un FST gráfico

El lenguaje que genera y reconoce el conjunto de los transductores conectados es el siguiente:

Información científica,*Información científica*.**N:fs**
Información tecnológica,*Información científica*.**N:fs**
Información técnica,*Información científica*.**N:fs**
:TransInf,*Información científica*.**N:fs**
    Comunicación de noticias,*Información científica*.**N:fs**
    Flujo de información,*Información científica*.**N:fs**
    Transferencia de la información,*Información científica*.**N:fs**
    :DifusInf,*Información científica*.**N:fs**
        Circulación internacional de materiales,*Información científica*.**N:fs**
        Difusión de conocimientos,*Información científica*.**N:fs**
        Difusión de la información,*Información científica*.**N:fs**
        Distribución de libros,*Información científica*.**N:fs**
        Intercambio de información,*Información científica*.**N:fs**

Mediante las series de FST léxicos imbricados se pueden reconocer todas aquellas secuencias de los textos que se equiparen a cualquiera de las expresiones representadas y, como en otros casos, el índice resultante se puede usar en un IRS para agrupar todas las apariciones de las secuencias indizadas.

En el Capítulo 7 se mostrará el resultado de la aplicación de los FST gráficos al *corpus de verificación* con el objetivo básico de localizar e indizar todas las expresiones que se equiparen a los transductores gráficos desarrollados en este trabajo.

Capítulo 6

CONSTRUCCIÓN DE ANALIZADORES DE SINTAGMAS NOMINALES CON TÉCNICAS DE ESTADO-FINITO

Si en el capítulo anterior consideramos que el *núcleo* de una palabra se formalizaba en el *stem*, en este capítulo partimos de que el *núcleo* de un Sintagma Nominal (SN) se formaliza en el *nombre*. Bajo este planteamiento, el objetivo ahora es desarrollar la metodología que se ha seguido para la construcción de los recursos de análisis sintáctico –básicamente *Gramáticas Parciales*– que se utilizarán para reconocer las estructuradas de las variantes de los Sintagmas Nominales (SSNN). Una vez obtenidas las gramáticas se procederá a su representación equivalente en autómatas y transductores por medio de una interfaz gráfica, *FSGraph*, desarrollado por Silberztein (Silberztein 1996; Silberztein 2000).

Las estructuras de los SSNN se van a especificar como *hipótesis explicativas* formuladas en términos de Expresiones Regulares, que se trasladarán a reconocedores de estado-finito, y se comprobarán posteriormente en un *corpus de verificación*. El proceso de *deducción de hipótesis* se puede desarrollar bien de forma intuitiva, trasladando las expresiones sintácticas directamente a AFD gráficos, o bien de forma sistemática obteniendo las Gramáticas Regulares a partir de las *derivadas* de las Expresiones Regulares y creando a continuación los AFD, que se encargarán de reconocer el lenguaje descrito por tales expresiones. El primer paso de este proceso es la *especificación* detallada de las distintas estructuras de los SSNN, a continuación se obtendrán las *Gramáticas Regulares Parciales, o Locales,* que identifiquen tales estructuras. Por último, dichas gramáticas se trasladarán, según su complejidad, al mecanismo de reconocimiento más adecuado.

Un SN es una construcción compleja compuesta por un *nombre*, ***núcleo*** del SN, y opcionalmente por un ***determinante***, o un **cuantificador**, y distintos ***modificadores***. Todos estos elementos se consideran los ***constituyentes***, o componentes de las formas complejas, que funcionan como unidades en las construcciones nominales. En la estructura de constituyentes de los SSNN, un componente subordinado a un componente nuclear se denomina comúnmente ***modificador*** y su función es especificar, delimitar o precisar un *nombre nuclear*, o *head noun* (N_h). Se trata de una noción *funcional* para distinguir un constituyente que especifica otro constituyente, considerado el *núcleo* de una construcción lingüística. La función de modificador la pueden realizar diversas categorías. A su vez, los modificadores pueden estar en posición *prenominal*, delante del núcleo que modifican:

$$\left\{\begin{array}{l}\texttt{NOMBRE}\\ \texttt{ADJETIVO}\\ \texttt{PARTICIPIO}\\ \texttt{ADVERBIO}\\ \texttt{POSESIVO}\\ \texttt{DEMOSTRATIVO}\\ \texttt{CARDINAL}\\ \texttt{ORDINAL}\end{array}\right\}\ \textbf{NOMBRE NUCLEAR}\ (\mathbf{N_h})$$

o en posición *posnominal*, detrás del núcleo que modifican:

$$\textbf{NOMBRE NUCLEAR}\ (\mathbf{N_h})\left\{\begin{array}{l}\texttt{ADJETIVO}\\ \texttt{PARTICIPIO}\\ \texttt{CARDINAL}\\ \texttt{ORDINAL}\\ \texttt{SINTAGMA PREPOSICIONAL}\\ \texttt{ORACIÓN}\end{array}\right\}$$

Aunque consideramos de forma genérica que la relación que mantienen determinados constituyentes –como *Adjetivos* (A), *Partitivos* (PA), *Sintagmas Preposicionales* (SSPP) u *Oraciones* (O) – con el *nombre* es la de ***modificador***, se puede establecer la siguiente matización según el tipo de relaciones entre los elementos constituyentes:

1. *Complementos argumentales*, o complementos reclamados por el núcleo nominal.
2. *Complementos adjuntos*, o complementos no reclamados por el núcleo nominal, denominados propiamente *modificadores*,

En el proceso de reconocimiento se podría distinguir la *obligatoriedad* y la *no-obligatoriedad* con la que se comportan los constituyentes que acompañan al *núcleo* del

SN. Los *complementos argumentales* estarían en relación con los nombres que exigen determinados constituyentes, y los *complementos adjuntos*, también denominados *modificadores*, estarían en relación con los nombres que no exigen constituyentes. Normalmente los nombres que exigen determinados constituyentes son nombres derivados de verbos, y los SSNN que tienen como *núcleo* tales nombres van acompañados obligatoriamente de determinados constituyentes, como *Adjetivos* (A), *Sintagmas Preposicionales* (SSPP), u *Oraciones subordinadas* (O). En estas construcciones la relación existente entre el nombre y sus constituyentes es una conexión *semántica*, similar a la de un verbo y su complemento, o los que es lo mismo, de un *predicado* y sus *argumentos*.

En este trabajo no vamos a tener en cuenta si el *núcleo* del SN reclama o no distintos tipos de complementos porque tendríamos que introducir aspectos semánticos que están fuera de nuestro alcance. Por consiguiente, nuestro interés se centrará en las distintas formas sintácticas en las que se realizan los SSNN y, a efectos puramente *descriptivos*, vamos a considerar que todos los constituyentes que acompañan al nombre funcionan como ***modificadores***. Es importante tener en cuenta esta observación para que las hipótesis explicativas, que vamos a proponer para la descripción de las estructuras sintácticas, no dén lugar a ningún tipo de confusión. A partir de aquí, se puede hacer la siguiente distinción:

- *Modificadores restrictivos*: cuando delimitan, precisan y restringen la referencia del sintagma al que modifican.
- *Modificadores no-restrictivos*: cuando no precisan ni restringen la referencia del sintagma porque no modifican sustancialmente su significado. Su función se limita a añadir una explicación o aclaración no necesaria para entender el sintagma, suelen ir separados por una pausa, o *coma*.

Básicamente, nuestro objetivo se va a dirigir a los ***modificadores restrictivos***, porque son los que inciden en la determinación y especificación del núcleo nominal, y según el tipo de constituyentes que realiza esta función se distinguen:

a) *SSNN de Estructura Simple*, o sintagmas con modificadores restrictivos simples, como *Adjetivos*, *Partitivos*, *Ordinales*, o *Cardinales*.
b) *SNSN de Estructura Compleja*, o sintagmas con modificadores restrictivos que incluyen otros sintagmas. Los grupos nominales de estructura compleja están formados por un SN en el que están incrustados *Sintagmas Preposicionales*, u *Oraciones de Relativo.*

La consecuencia directa que se sigue de todo lo anterior es que, aunque los modificadores puedan funcionar como *constituyentes restrictivos* y *no-restrictivos*, únicamente vamos a considerar los casos en los que los modificadores limiten o restrinjan las posibilidades referenciales del núcleo nominal modificado. Por otra parte, el análisis de constituyentes muestra que los SSNN no son simples secuencias lineales de elementos sino que están formados por agrupaciones según una determinada estructura, dándose dos tipos de ordenación:

- *Estructuras lineales*
- *Estructuras jerárquicas*

En las estructuras lineales los constituyentes se representan como una sucesión horizontal, o en la dirección de *izquierda-a-derecha*, en la que no se refleja la *estructura de constituyentes*. Por el contrario, en las estructuras jerárquicas los constituyentes se representan en una sucesión vertical, o en la dirección de *arriba-a-abajo*, en la que sí se refleja la *estructura de constituyentes*, por medio del establecimiento de relaciones sintácticas. Habitualmente esas relaciones sintácticas se muestran en los denominados *árboles de derivación*, o por medio de *corchetes etiquetados*. En el análisis de constituyentes, los componentes superiores, que se agrupan por un nudo común en el árbol estructural, se denominan *constituyentes*

inmediatos, y los componentes inferiores, que aparecen en la fase final del análisis en el plano de la palabra, se denominan *constituyentes terminales*.

El problema está en que las *Gramáticas de Estado-Finito* describen linealmente los SSNN. En los casos en los que un *nombre* aparece junto a cualquier elemento simple es suficiente una descripción lineal, sin embargo cuando un SN incluye otro sintagma, como un *Sintagma Preposicional* (SP), o una *Oración* (O), sería necesario obtener una descripción jerárquica de la estructura sintagmática. Dicho de otro modo, las *Gramáticas de Estado-Finito* sólo permiten una descripción *lineal* de los constituyentes de los SSNN, pero cuando estas construcciones presentan estructuras complejas en las que aparecen oraciones u otros sintagmas dentro del propio SN, por medio de las denominadas estructuras *incrustadas* o *anidadas*, sería necesario obtener una representación de dicha estructura, y no una simple descripción, para poder determinar claramente cuáles son los constituyentes que modifican al *núcleo* del SN.

Los SSNN que son objeto de esta investigación se pueden generar por medio de *Gramáticas de Estado-Finito*, pero si quisiéramos obtener una representación estructurada de dichos sintagmas necesitaríamos otro tipo de formalismo, como pueden ser las *Gramáticas Libres de Contexto*, o las *Gramáticas Sintagmáticas*. En relación con esto, es preciso aclarar que las *Gramáticas de Estado-Finito* son capaces de generar la mayoría de las construcciones sintagmáticas pero su limitación se encuentra, como ya se ha dicho, en su *incapacidad para generar determinadas estructuras.* El procedimiento para representar SSNN con estructuras complejas se desarrollará en los próximos apartados, y básicamente consistirá en trasladar los constituyentes que forman los SSNN a *Transductores Sintácticos*.

Otro aspecto fundamental del análisis de constituyentes es la posibilidad de repetición de determinados elementos, cadenas de etiquetas POS, o de la repetición de determinadas estructuras *incrustadas* dentro de la estructura de los SSNN. Estos dos fenómenos lingüísticos se pueden expresar por medio de dos técnicas:

1. *Iteración de constituyentes*.
2. *Recursividad de constituyentes*.

La capacidad para describir las estructuras sintagmáticas en las que aparezca *iteración* o *repetición* de constituyentes se puede formular por medio de *operadores de Kleene*, como $\texttt{SN} \rightarrow \texttt{N}^{+}\ \texttt{A}$, posteriormente las *construcciones iterativas* se pueden generar con *Gramáticas Regulares*. Sin embargo, hay otro tipo de estructuras, como las denominadas *construcciones recursivas*, que sólo se pueden representan por medio de *Gramáticas Libres de Contexto*, o *Gramáticas Sintagmáticas*, pero que en algunos casos se pueden obtener representaciones equivalentes por medio de *Gramáticas Regulares*. Para la representación de la *recursividad* de estructuras incrustadas dentro de SSNN (como puede ser el caso concreto de la construcción sintagmática: «*el sistema de recuperación de información del catálogo de la biblioteca de la Universidad de Granada de...*») sería preciso utilizar repetidamente reglas del tipo:

$\texttt{SN} \rightarrow (\texttt{DET})\ \texttt{N SP}$

$\texttt{SP} \rightarrow \texttt{Prep SN}$

$\texttt{SP} \rightarrow \texttt{Prep}\ (\texttt{DET})\ \texttt{N SP}$

Las reglas de producción que se pueden aplicar un número indefinido de veces tienen en la parte *derecha* y en la parte *izquierda* algún elemento en común, como $\mathbf{A} \rightarrow \texttt{a}\ \mathbf{A}$. Con este tipo de reglas se podrían generar un número *infinito* de estructuras. Sin embargo, este tipo de reglas no pertenecen a las *Gramáticas Regulares*, ni se pueden generar con estas técnicas; para que esto fuera posible sería preciso establecer un límite a la posibilidad de su utilización indefinida. Por lo tanto, si quisiéramos que determinadas estructuras sintagmáticas se genen con *Gramáticas Regulares*, tendríamos que utilizar mecanismos *finitos* que permitieran liminar la reproducción indefinida de las reglas, esto se consigue añadiendo reglas del tipo $\mathbf{A} \rightarrow \texttt{a}$, dando lugar al siguiente aparato de reglas:

SN → (DET) N SP

SN → N

SP → Prep SN

SP → Prep (DET) N SP

SP → Prep N

La propiedad según la cual las reglas se pueden aplicar un número indefinido de veces se denomina *recursividad*. Por medio de esta propiedad las gramáticas pueden generar un número indefinido de construcciones, o estructuras *anidadas*, esto es, pueden generar un *número infinito de construcciones sintácticas*. Este tipo de estructuras se puede generar con reglas pertenecientes a las *Gramáticas Libres de Contexto*, o a las *Gramáticas Sintagmáticas*. No obstante, para poder generar estructuras recursivas con técnicas de estado-finito sería preciso construir Gramáticas Regulares equivalentes a las mencionadas. Pero el problema radica en que no siempre se pueden realizar esas equivalencias, a no ser que establezcan *límites a los fenómenos recursivos*, tal y como se va a proponer en el apartado correspondiente.

Para solucionar el problema anterior, nos vamos a basar en la concluyente declaración de Chomsky, a la que ya hicimos alusión, en la que se argumentaba que en las lenguas naturales existen *procesos de formación de construcciones sintácticas que las gramáticas de estado-finito son incapaces de manipular*, sin embargo *si los procesos recursivos*, es decir, las estructuras anidadas tienen *un limite finito sí es posible que se puedan generar con gramáticas de estado-finito* (Chomsky 1957). La observación anterior es clave y, apoyándonos en ella, en los siguientes apartados vamos a aplicar, en los casos en los que sea necesario representar las estructuras de los SSNN con constituyentes recursivos, un mecanismo de estado-finito que límite al número de estructuras incrustadas.

De cualquier forma, aunque operáramos con otro formalismo más expresivo que nos permitiera aplicar reglas capaces de generar un número infinito de estructuras *incrustadas*, en las lenguas naturales es un hecho observable que las construcciones recursivas nunca son de longitud infinita. Por esta razón, es preferible utilizar formalismos menos expresivos y mecanismos más débiles, pero más controlados, para representar tales estructuras: El anterior argumento es el que se va a plantear para la representación de las estructuras de los SSNN con recursividad de constituyentes.

6.1. Construcción de Gramáticas Parciales

Los SSNN son objetos lingüísticos estructurados, aunque su estructura no se manifieste. El primer paso para el desarrollo de las gramáticas que generen esos objetos lingüísticos es la *especificación* detallada de las estructuras sintácticas que se van a reconocer. Para la identificación de las estructuras sintagmáticas más simples, el procedimiento que vamos a seguir es la metodología formulada anteriormente, consistente en el desarrollo de los siguientes procesos:

1. *Especificar la estructura de los SSNN* por medio de Expresiones Regulares.
2. *Construir las Gramáticas Regulares* a partir de las derivaciones de las Expresiones Regulares.
3. *Trasladar las Gramáticas Regulares* a Autómatas y Transductores Gráficos.
4. *Compilar* los Autómatas y Transductores Gráficos en *Transductores de Estado-Finito Deterministas* (FST).
5. *Minimizar los Transductores de Estado-Finito Determinista*s.
6. *Obtener los FST* que se encarguen de insertar marcas alrededor de las variantes de los SSNN especificados.

El editor gráfico *FSGraph* nos va a proporcionar la herramienta informática para trasladar las gramáticas directamente a FST gráficos. Mediante esta aplicación los autómatas y los transductores que representan las gramáticas reconocedoras de SSNN se pueden compilar y *minimizar* de forma automática en autómatas y transductores deterministas. De este modo, las fases *cinco*, *seis* y *siete* del proceso anterior se van a realizar automáticamente, lo cual nos va a facilitar el reconocimiento de estructuras sintagmáticas.

Partimos de la hipótesis de que un *Sintagma Nominal* está formado por un *nombre* y diferentes elementos que se agrupan en torno a esta categoría, las estructuras más simples de estas construcciones nominales se pueden especificar de la siguiente forma:

A. Sintagmas compuestos por un **`Nombre`**, *núcleo*, acompañado de un `Determinante`, o de un `Cuantificador`.

$SN_0 \rightarrow$ `DET N` (`Deteminante` **Nombre)**

$SN_1 \rightarrow$ `CUANT N` (`Cuantificador` **Nombre**)

B. Sintagmas compuestos por un solo **`Nombre`**, *núcleo*, o un **`Nombre`** acompañado de un **Modificador**. De la misma forma, un sintagma puede estar compuesto por un *pronombre*, aunque para nuestro objetivo sólo serán de interés los SSNN cuyos núcleos sean *nombres*.

$SN_2 \rightarrow$ `N` (**Nombre**)

$SN_3 \rightarrow$ `N N` (**Nombre** `Nombre`)

$SN_4 \rightarrow$ `N A` (**Nombre** `Adjetivo`)

$SN_5 \rightarrow$ `A N` (`Adjetivo` **Nombre**)

$SN_6 \rightarrow$ `N PA` (**Nombre** `Participio-Adj`)

SN_7 → PA N (Participio-Adj **Nombre**)

SN_8 → POS N (Posesivo **Nombre**)

SN_9 → DEM N (Demostrativo **Nombre**)

SN_{10} → CARD N (Cardinal **Nombre**)

SN_{11} → N CARD (**Nombre** Cardinal)

SN_{12} → ORD N (Ordinal **Nombre**)

SN_{13} → N ORD (**Nombre** Ordinal)

SN_{14} → DET N A (Determinante **Nombre** Adjetivo)

SN_{15} → DET A N (Determinante Adjetivo **Nombre**)

SN_{16} → CUANT N A (Cuantificador **Nombre** Adjetivo)

SN_{17} → CUANT A N (Cuantificador Adjetivo **Nombre**)

SN_{18} → POS N A (Posesivo **Nombre** Adjetivo)

SN_{19} → POS A N (Posesivo Adjetivo **Nombre**)

SN_{20} → DEM N A (Demostrativo **Nombre** Adjetivo)

SN_{21} → DEM A N (Demostrativo Adjetivo **Nombre**)

SN_{22} → DET N PA (Determinante **Nombre** Participio-Adj)

SN_{23} → DET PA N (Determinante Participio-Adj **Nombre**)

SN_{24} → CUANT N PA (Cuantificador **Nombre** Participio-Adj)

SN_{25} → CUANT PA N (Cuantificador Participio-Adj **Nombre**)

SN_{26} → POS N PA (Posesivo **Nombre** Participio-Adj)

SN_{27} → POS PA N (Posesivo Participio-Adj **Nombre**)

SN_{28} → DEM N PA (Demostrativo **Nombre** Participio-Adj)

SN_{29} → DEM PA N (Demostrativo Participio-Adj **Nombre**)

SN_{30} → DET CARD N (Determinante Cardinal **Nombre**)

SN_{31} → DET N CARD (Determinante **Nombre** Cardinal)

SN_{32} → POS CARD N (Posesivo Cardinal **Nombre**)

SN_{33} → POS N CARD (Posesivo **Nombre** Cardinal)

SN_{34} → DEM CARD N (Demostrativo Cardinal **Nombre**)

SN_{35} → DEM N CARD (Demostrativo **Nombre** Cardinal)

SN_{36} → DET ORD N (Determinante Ordinal **Nombre**)

SN_{37} → DET N ORD (Determinante **Nombre** Ordinal)

SN_{38} → POS ORD N (Posesivo Ordinal **Nombre**)

SN_{39} → POS N ORD (Posesivo **Nombre** Ordinal)

SN_{40} → DEM ORD N (Demostrativo Ordinal **Nombre**)

SN_{41} → DEM N ORD (Demostrativo **Nombre** Ordinal)

SN_{42} → DET A N A (Determinante Adjetivo **Nombre** Adjetivo)

SN_{43} → DET PA N PA (Determinante Partic-Adj **Nombre** Partic-Adj)

SN_{44} → DET A N PA (Determinante Adjetivo **Nombre** Partic-Adj)

SN_{45} → DET PA N A (Determinante Partic-Adj **Nombre** Adjetivo)

SN_{46} → CUANT DET N (Cuantificador Determinante **Nombre**)

SN_{47} → CUANT DEM N (Cuantificador Demostrativo **Nombre**)

SN_{48} → CUANT POS ORD N (Cuantificador Posesivo Ordinal **Nombre**)

SN_{49} → CUANT POS N ORD (Cuantificador Posesivo **Nombre** Ordinal)

SN_{50} → CUANT POS CARD N (Cuantificador Posesivo Cardinal **Nombre**)

SN_{51} → DET ADV A N (Determinante Adverbio Adjetivo **Nombre**)

SN_{52} → DET ADV PA N (Determinante Adverbio Partic-Adj **Nombre**)

SN_{53} → CARD ORD N (Cardinal Ordinal **Nombre**)

SN_{54} → DET CARD ORD N (Determinante Cardinal Ordinal **Nombre**)

A su vez, los SSNN anteriores y todas sus variantes se van a representar en *sintagmas controlados* que se encargarían de agrupar las distintas construcciones sintagmáticas:

$SN_{55} \rightarrow$ N
N N
DET N
CUANT N
POS N
DEM N
ORD N
CARD N
CUANT POS N
CUANT DET N
CUANT DEM N
DEM ORD N
DEM CARD N
DET ORD N
DET CARD N
POS ORD N
POS CARD N
CUANT DEM N
CUANT DEM CARD N
CUANT DET ORD N
CUANT DET CARD N
CUANT POS ORD N
CUANT POS CARD N
CARD ORD N
DET CARD ORD N
POS CARD ORD N
DEM CARD ORD N

$SN_{56} \rightarrow$ N A
A N
N PA
PA N
DET A N
DET N A
DET PA N
DET N PA
CUANT A N
CUANT N A
CUANT PA N
CUANT N PA
DEM A N
DEM N A
DEM PA N
DEM N PA
A N A
A N PA
PA N A
PA N PA
POS N A
POS A N
POS N PA
POS PA N
POS A N A
POS A N PA
POS PA N A
POS PA N PA
DET A N A
DET A N PA
DET PA N A
DET PA N PA
DEM A N A
DEM A N PA
DEM PA N A
DEM PA N PA
CUANT A N A
CUANT A N PA
CUANT PA N A
CUANT PA N PA
DET ADV A N
DET ADV PA N

Por otra parte, con el objetivo de describir las estructuras de los SSNN en las que se produce repetición de constituyentes se va a emplear la *clausura positiva de Kleene*. El uso de esta notación origina que el número de estructuras se multiplique, para evitar que las derivaciones posteriores sean demasiado extensas hemos adoptado la decisión de *fusionar*, o *conflactar*, las estructuras similares. Indicamos a continuación las construcciones sintagmáticas con *iteración de constituyentes* más representativas,

teniendo en cuenta que las estructuras de estos SSNN agrupan una serie de variantes que se especificarán en el apartado correspondiente:

$SN_{57} \rightarrow$ DET N^+ (Deteminante iter. **Nombre**)

$SN_{58} \rightarrow$ DET N^+ A (Determinante iter. **Nombre** Adjetivo)

$SN_{59} \rightarrow$ DET N A^+ (Determinate **Nombre** iter. Adjetivo)

$SN_{60} \rightarrow$ DET N^+ A^+ (Determinante iter. **Nombre** iter. Adjetivo)

$SN_{61} \rightarrow$ DET A N^+ (Determinante Adjetivo iter. **Nombre**)

$SN_{62} \rightarrow$ DET A^+ N (Determinante iter. Adjetivo **Nombre**)

$SN_{63} \rightarrow$ DET A^+ N^+ (Determinante iter. Adjetivo iter. **Nombre**)

$SN_{64} \rightarrow$ DET A N^+ A (Determinante Adjetivo iter. **Nombre** Adjetivo)

$SN_{65} \rightarrow$ DET A^+ N A (Determinante iter. Adjetivo **Nombre** Adjetivo)

$SN_{66} \rightarrow$ DET A N A^+ (Determinante Adjetivo **Nombre** iter. Adjetivo)

$SN_{67} \rightarrow$ DET A^+ N^+ A (Determinante iter. Adjetivo iter. **Nombre** Adjetivo)

$SN_{68} \rightarrow$ DET A N^+ A^+ (Determinante Adjetivo iter. **Nombre** iter. Adjetivo)

$SN_{69} \rightarrow$ DET A^+ N A^+ (Determinante iter. Adjetivo **Nombre** iter. Adjetivo)

$SN_{70} \rightarrow$ DET A^+ N^+ A^+ (Det iter. Adjetivo iter. **Nombre** iter. Adjetivo)

6.1.1. SSNN de Estructura Simple

Para desarrollar las gramáticas que sean capaces de generar las estructuras simples de los SSNN se va a aplicar la metodología expuesta anteriormente. Para ello, las estructuras sintácticas se plantean como *hipótesis explicativas* representadas en términos de *Expresiones Regulares*, a partir de las cuales obtendremos las *Gramáticas*

Parciales que sean capaces de generarlas, teniendo en cuenta además que las estructuras reconocidas se podrán reutilizar posteriormente como parte de otras *Gramáticas Globales*. A partir de las gramáticas se van a desarrollar los AFD que las reconocen y, por último, se van a obtener los transductores gráficos que se encarguen de insertar etiquetas a los SSNN identificados por medio de la interfaz *FSGraph*:

1. **$SN_0 \rightarrow$ DET N (Determinante Nombre)**

$ER_0 = DET\ N$

$D_{DET}(ER_0) =$
$[\ D_{DET}(DET)\ N + \alpha(DET)\ D_{DET}(N)\ ] =$
$\lambda\ N + \varnothing = N = ER_1$

$D_N(ER_0) =$
$[\ D_N(DET)\ N + \alpha(DET)\ D_N(N)\ ] = \varnothing + \varnothing = \varnothing$

$D_{DET}(ER_1) =$
$D_{DET}(N) = \varnothing$

$D_N(ER_1) =$
$D_N(N) = \lambda$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_0 es la siguiente:

$G = (\{DET, N\}, \{ER_0, ER_1\}, ER_0, P)$

Temiendo en cuenta que las categorías gramaticales son los símbolos terminales de la gramática y que las *reglas de producción*, P, se definirían como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= N$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$AF = (\{DET, N\}, \{ER_0, ER_1, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, \text{DET}) = ER_1$
$f\ (ER_0, \text{N}) = \varnothing$
$f\ (ER_1, \text{DET}) = \varnothing$
$f\ (ER_1, \text{N}) = F$
$f\ (F, \text{DET}) = \varnothing$
$f\ (F, \text{N}) = \varnothing$

se renombran los estados:

$f\ (q_0, \text{DET}) = q_1$
$f\ (q_0, \text{N}) = \varnothing$
$f\ (q_1, \text{DET}) = \varnothing$
$f\ (q_1, \text{N}) = q_2$
$f\ (q_2, \text{DET}) = \varnothing$
$f\ (q_2, \text{N}) = \varnothing$

El autómata obtenido no se puede reducir, y por lo tanto se considera el *Autómata Finito Determinista Mínimo*, a continuación se eliminan las transiciones vacías y se redefine la función de transición como:

$f\ (q_0, \text{DET}) = q_1$
$f\ (q_1, \text{N}) = q_2$

A su vez, el AFD se puede representar de forma gráfica en un *diagrama de transiciones* (Fig. 6.1), o en una matriz bidimensional, *tabla de transiciones* (Fig. 6.2), cuyos elementos corresponden con los del diagrama de transiciones. En relación con esto, la represtación gráfica en un diagrama de transiciones sería equivalente a una *Red de Transición* (RT) en la que los arcos estarían etiquetados con las categorías gramaticales.

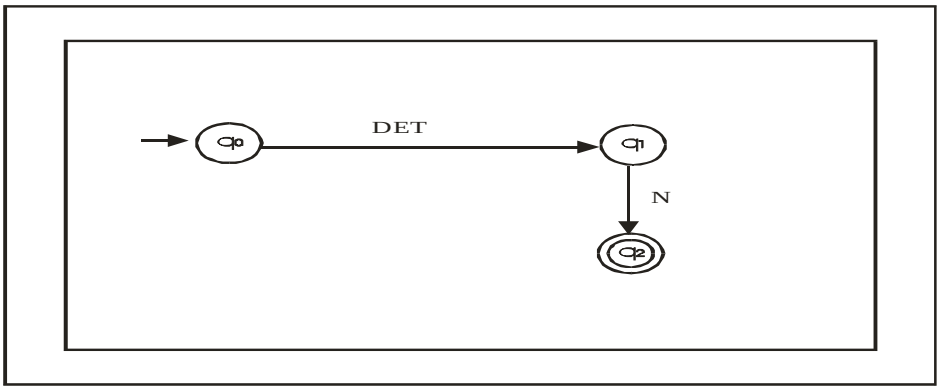


Fig. 6.1: Diagrama de transiciones del AFD que reconoce la estructura SN_0

f	D E T	N
→ q_0	q_1	∅
q_1	∅	q_2
* q_2	∅	∅

Fig. 6.2: Tabla de transiciones del AFD que reconoce la estructura SN_0

Sin embargo, para simplificar todo el proceso esta última fase se va a excluir a partir de ahora, de tal forma que la representación de las gramáticas se va a realizar en los siguientes casos directamente en el editor gráfico *FSGraph* (Fig. 6.3). Además, la interfaz no sólo nos permitirá obtener los transductores gráficos que se encargarán de establecer las marcas estructurales a los sintagmas identificados sino que hará posible minimizar los autómatas y transductores de forma automática cuando sea necesario.

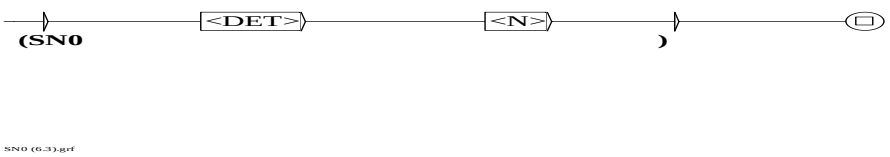


Fig. 6.3: Tabla de transiciones del FST gráfico que reconoce la estructura SN_0

2. **$SN_1 \rightarrow$ CUANT N (Cuantificador Nombre)**

ER_0 = CUANT N

$D_{CUANT}(ER_0)=$
$[\ D_{CUANT}(CUANT)\ N + \alpha(CUANT)\ D_{CUANT}(N)\ ]=$
$\lambda\ N + \varnothing = N = ER_1$

$D_N(ER_0)=$
$[\ D_N(CUANT)\ N + \alpha(CUANT)\ D_N(N)\ ]=\varnothing + \varnothing = \varnothing$

$D_{CUANT}(ER_1)=$
$D_{CUANT}(N)=\varnothing$

$D_N(ER_1)=$
$D_N(N)=\lambda$

La *Gramática Regular* que reconoce el sintagma SN_1 es la siguiente:

$G = (\{CUANT, N\}, \{ER_0, ER_1\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= CUANT\ ER_1$
$ER_1 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{CUANT, N\}, \{ER_0, ER_1, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, CUANT) = ER_1$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_1, CUANT) = \varnothing$
$f\ (ER_1, N) = F$
$f\ (F, CUANT) = \varnothing$
$f\ (F, N) = \varnothing$

A continuación, se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0,\ \text{CUANT}) = q_1$
$f\ (q_1,\ \text{N}) = q_2$

El transductor gráfico que reconoce el sintagma SN_1 es el siguiente (Fig. 6.4):

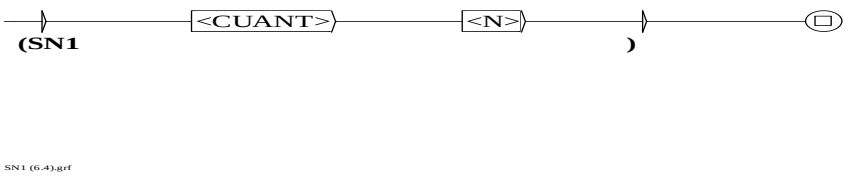


Fig. 6.4: FST gráfico que reconoce la estructura SN_1

3. **$SN_2 \rightarrow$ N (Nombre)**

$ER_0 = N$

$D_N(ER_0) =$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_2 es la siguiente:

$G = (\{N\},\ \{ER_0\},\ ER_0,\ P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{N\},\ \{ER_0,\ F\},\ f,\ ER_0,\ F)$

donde la función de transición, f, se define como:

$f\ (ER_0,\ N) = F$
$f\ (F,\ N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0,\ N) = q_1$

El transductor gráfico que reconoce el sintagma SN_2 es el siguiente (Fig. 6.5):

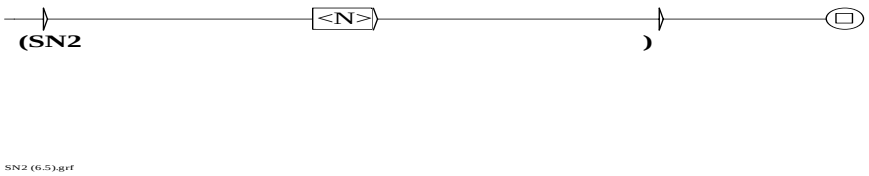


Fig. 6.5: FST gráfico que reconoce la estructura SN_2

4. **SN_3 → N N (Nombre Nombre)**

$ER_0 = N\ N$

$D_N(ER_0) =$
$[\ D_N(N)\ N + \alpha(N)\ D_N(N)\ ] =$
$\lambda\ N + \varnothing\ N =$
$N = ER_1$

$D_N(ER_1) =$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_3 es la siguiente:

$G = (\{N\},\ \{ER_0,\ ER_1\},\ ER_0,\ P)$

donde las *reglas de producción*, P, se definen como:

ER_0 ::= N ER_1
ER_1 ::= N

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{N\}, \{ER_0, ER_1, F\}, f, ER_0, F)$

donde la función de transición, f , se define como:

$f\ (ER_0, N) = ER_1$
$f\ (ER_1, N) = F$
$f\ (F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, N) = q_1$
$f\ (q_1, N) = q_2$

El transductor gráfico que reconoce el sintagma SN_3 es el siguiente (Fig. 6.6):

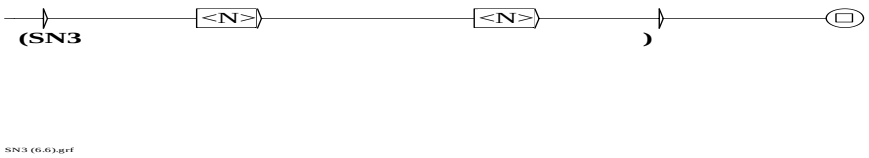


Fig. 6.6: FST gráfico que reconoce la estructura SN_3

5. **SN_4 → N A (Nombre Adjetivo)**

ER_0 = N A

$D_N(ER_0) =$
$[\ D_N(N)\ A + \alpha(N)\ D_N(A)\ ] =$
$\lambda\ A + \varnothing = A = ER_1$

$D_A(ER_0) =$
$[\ D_A(N)\ A + \alpha(N)\ D_A(A)\ ] = \varnothing + \varnothing = \varnothing$

$D_N(ER_1) =$
$D_N(A) = \varnothing$

$D_A(ER_1) =$
$D_A(A) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_4 es la siguiente:

$G = (\{A, N\}, \{ER_0, ER_1\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= A\ ER_1$
$ER_1 ::= A$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{A, N\}, \{ER_0, ER_1, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, N) = ER_1$
$f\ (ER_0, A) = \varnothing$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_1, A) = F$
$f\ (F, N) = \varnothing$
$f\ (F, A) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, N) = q_1$
$f\ (q_1, A) = q_2$

El transductor gráfico que reconoce el sintagma SN_4 es el siguiente (Fig. 6.7):

Fig. 6.7: FST gráfico que reconoce la estructura SN_4

6. **$SN_5 \rightarrow$ A N (Adjetivo Nombre)**

ER_0 = A N

$D_A(ER_0)=$
$[\ D_A(A)\ N + \alpha(A)\ D_A(N)\ ]=$
$\lambda\ N + \varnothing = N = ER_1$

$D_N(ER_0)=$
$[\ D_N(A)\ N + \alpha(A)\ D_N(N)\ ]=\varnothing + \varnothing = \varnothing$

$D_A(ER_1)=$
$D_A(N)=\varnothing$

$D_N(ER_1)=$
$D_N(N)=\lambda$

La *Gramática Regular* que reconoce el sintagma SN_5 es la siguiente:

$G = (\{A, N\}, \{ER_0, ER_1\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= A\ ER_1$
$ER_1 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{A, N\}, \{ER_0, ER_1, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, A) = ER_1$
$f(ER_0, N) = \varnothing$
$f(ER_1, A) = \varnothing$
$f(ER_1, N) = F$
$f(F, A) = \varnothing$
$f(F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, A) = q_1$
$f(q_1, N) = q_2$

El transductor gráfico que reconoce el sintagma SN_5 es el siguiente (Fig. 6.8):

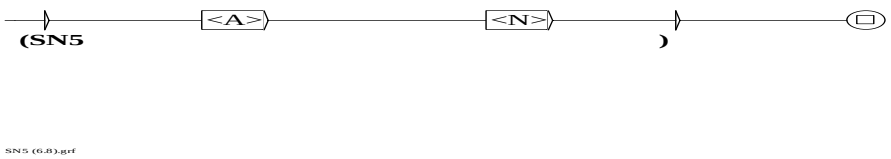


Fig. 6.8: FST gráfico que reconoce la estructura SN_5

7. **SN_6 → N PA (Nombre Participio- Adj)**

ER_0 = N PA

$D_N(ER_0) =$
$[\ D_N(N)\ PA + \alpha(N)\ D_N(PA)\ ] =$
$\lambda\ PA + \varnothing = PA = ER_1$

$D_{PA}(ER_0) =$
$[\ D_{PA}(N)\ PA + \alpha(N)\ D_{PA}(PA)\ ] = \varnothing + \varnothing = \varnothing$

$D_N(ER_1) =$
$D_N(PA) = \varnothing$

$D_{PA}(ER_1) =$
$D_{PA}(PA) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_6 es la siguiente:

$G = (\{N, PA\}, \{ER_0, ER_1\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= N\ ER_1$
$ER_1 ::= PA$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{N, PA\}, \{ER_0, ER_1, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, N) = ER_1$
$f\ (ER_0, PA) = \varnothing$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_1, PA) = F$
$f\ (F, N) = \varnothing$
$f\ (F, PA) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, N) = q_1$
$f\ (q_1, PA) = q_2$

El transductor gráfico que reconoce el sintagma SN_6 es el siguiente (Fig. 6.9):

Fig. 6.9: FST gráfico que reconoce la estructura SN_6

8. **$SN_7 \rightarrow$ PA N (Participio - Adj Nombre)**

ER_0 = PA N

$D_{PA}(ER_0) =$
$[\ D_{PA}(PA)\ N + \alpha(PA)\ D_A(N)\ ] =$
$\lambda\ N + \varnothing = N = ER_1$

$D_N(ER_0) =$
$[\ D_N(PA)\ N + \alpha(PA)\ D_N(N)\ ] = \varnothing + \varnothing = \varnothing$

$D_{PA}(ER_1) =$
$D_{PA}(N) = \varnothing$

$D_N(ER_1) =$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_7 es la siguiente:

$G = (\{PA, N\}, \{ER_0, ER_1\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= PA\ ER_1$
$ER_1 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{PA, N\}, \{ER_0, ER_1, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, PA) = ER_1$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_1, PA) = \varnothing$
$f\ (ER_1, N) = F$
$f\ (F, PA) = \varnothing$
$f\ (F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, PA) = q_1$
$f(q_1, N) = q_2$

El transductor gráfico que reconoce el sintagma SN_7 es el siguiente (Fig. 6.10):

Fig. 6.10: FST gráfico que reconoce la estructura SN_7

9. **SN_8 → POS N (Posesivo Nombre)**

$ER_0 = POS\ N$

$D_{POS}(ER_0) =$
$[\ D_{POS}(POS)\ N + \alpha(POS)\ D_{POS}(N)\ ] =$
$\lambda\ N + \varnothing = N = ER_1$

$D_N(ER_0) =$
$[\ D_N(POS)\ N + \alpha(POS)\ D_N(N)\ ] = \varnothing + \varnothing = \varnothing$

$D_{POS}(ER_1) =$
$D_{POS}(N) = \varnothing$

$D_N(ER_1) =$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_8 es la siguiente:

$G = (\{POS, N\}, \{ER_0, ER_1\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= POS\ ER_1$
$ER_1 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{POS, N\}, \{ER_0, ER_1, F\}, f, ER_0, F)$

donde la función de transición, f , se define como:

$f\ (ER_0, POS) = ER_1$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_1, POS) = \varnothing$
$f\ (ER_1, N) = F$
$f\ (F, POS) = \varnothing$
$f\ (F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, POS) = q_1$
$f\ (q_1, N) = q_2$

El transductor gráfico que reconoce el sintagma SN_8 es el siguiente (Fig. 6.11):

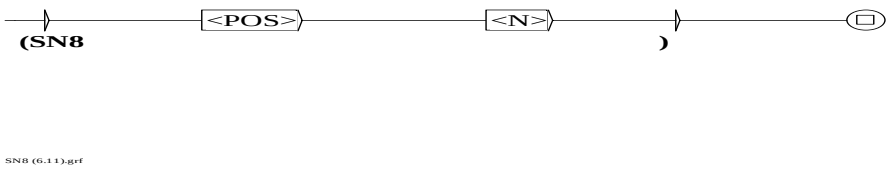


Fig. 6.11: FST gráfico que reconoce la estructura SN_8

10. **SN_9 → DEM N (Demostrativo Nombre)**

ER_0 = DEM N

$D_{POS}(ER_0)=$
$[\ D_{DEM}(DEM)\ N\ +\ \alpha(DEM)\ D_{DEM}(N)\ ]=$
$\lambda\ N\ +\ \varnothing=\ N\ =\ ER_1$

$D_N(ER_0)=$
$[\ D_N(DEM)\ N\ +\ \alpha(DEM)\ D_N(N)\ ]=\varnothing\ +\ \varnothing=\ \varnothing$

$D_{DEM}(ER_1)=$
$D_{DEM}(N)=\varnothing$

$D_N(ER_1)=$
$D_N(N)=\lambda$

La *Gramática Regular* que reconoce el sintagma SN_9 es la siguiente:

$G = (\{DEM,\ N\},\ \{ER_0,\ ER_1\},\ ER_0,\ P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DEM\ ER_1$
$ER_1 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DEM,\ N\},\ \{ER_0,\ ER_1,\ F\},\ f,\ ER_0,\ F)$

donde la función de transición, f, se define como:

$f\ (ER_0,\ DEM)=\ ER_1$
$f\ (ER_0,\ N)\ =\varnothing$
$f\ (ER_1,\ DEM)=\varnothing$
$f\ (ER_1,\ N)\ =\ F$
$f\ (F,\ DEM)\ =\ \varnothing$
$f\ (F,\ N)\ =\ \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0,\ DEM)=\ q_1$
$f\ (q_1,\ N)\ =\ q_2$

El transductor gráfico que reconoce el sintagma SN_9 es el siguiente (Fig. 6.12):

Fig. 6.12: FST gráfico que reconoce la estructura SN_9

11. **SN_{10} → CARD N (Cardinal Nombre)**

ER_0 = CARD N

$D_{CARD}(ER_0)=$
$[\ D_{CARD}(CARD)\ N + \alpha(CARD)\ D_{CARD}(N)\ ]=$
$\lambda\ N + \varnothing = N = ER_1$

$D_N(ER_0)=$
$[\ D_N(CARD)\ N + \alpha(CARD)\ D_N(N)\ ] = \varnothing + \varnothing = \varnothing$

$D_{CARD}(ER_1)=$
$D_{CARD}(N) = \varnothing$

$D_N(ER_1)=$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{10} es la siguiente:

$G = (\{CARD, N\}, \{ER_0, ER_1\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= CARD\ ER_1$
$ER_1 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{CARD, N\}, \{ER_0, ER_1, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, CARD) = ER_1$
$f(ER_0, N) = \varnothing$
$f(ER_1, CARD) = \varnothing$
$f(ER_1, N) = F$
$f(F, CARD) = \varnothing$
$f(F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, CARD) = q_1$
$f(q_1, N) = q_2$

El transductor gráfico que reconoce el sintagma SN_{10} es el siguiente (Fig. 6.13):

Fig. 6.13: FST gráfico que reconoce la estructura SN_{10}

12. **SN_{11} → N CARD (Nombre Cardinal)**

ER_0 = N CARD

$D_N(ER_0) =$
$[\ D_N(N)\ CARD + \alpha(N)\ D_N(CARD)\ ] =$
$\lambda\ CARD + \varnothing = CARD = ER_1$

$D_{CARD}(ER_0) =$
$[\ D_{CARD}(N)\ CARD + \alpha(N)\ D_{CARD}(CARD)\ ] = \varnothing + \varnothing = \varnothing$

$D_N(ER_1) =$
$D_N(CARD) = \varnothing$

$D_{CARD}(ER_1) =$
$D_{CARD}(CARD) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{11} es la siguiente:

$G = (\{N, CARD\}, \{ER_0, ER_1\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= N\ ER_1$
$ER_1 ::= CARD$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{N, CARD\}, \{ER_0, ER_1, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, N) = ER_1$
$f\ (ER_0, CARD) = \varnothing$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_1, CARD) = F$
$f\ (F, N) = \varnothing$
$f\ (F, CARD) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, N) = q_1$
$f\ (q_1, CARD) = q_2$

El transductor gráfico que reconoce el sintagma SN_{11} es el siguiente (Fig. 6.14):

Fig. 6.14: FST gráfico que reconoce la estructura de SN_{11}

13. **SN_{12} → ORD N (Ordinal Nombre)**

$ER_0 = ORD\ N$

$D_{ORD}(ER_0) =$
$[\ D_{ORD}(ORD)\ N + \alpha(ORD)\ D_{ORD}(N)\ ] =$
$\lambda\ N + \varnothing = N = ER_1$

$D_N(ER_0) =$
$[\ D_N(ORD)\ N + \alpha(ORD)\ D_N(N)\ ] = \varnothing + \varnothing = \varnothing$

$D_{ORD}(ER_1) =$
$D_{ORD}(N) = \varnothing$

$D_N(ER_1) =$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{12} es la siguiente:

$G = (\{ORD, N\}, \{ER_0, ER_1\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= ORD\ ER_1$
$ER_1 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{ORD, N\}, \{ER_0, ER_1, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, ORD) = ER_1$
$f(ER_0, N) = \varnothing$
$f(ER_1, ORD) = \varnothing$
$f(ER_1, N) = F$
$f(F, ORD) = \varnothing$
$f(F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, ORD) = q_1$
$f(q_1, N) = q_2$

El transductor gráfico que reconoce el sintagma SN_{12} es el siguiente (Fig. 6.15):

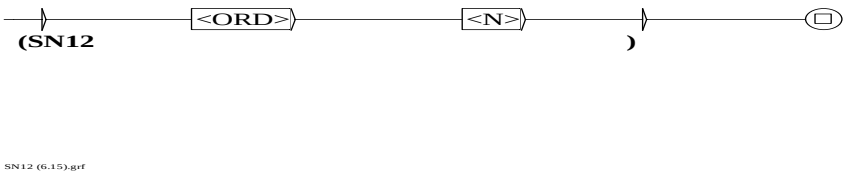


Fig. 6.15: FST gráfico que reconoce la estructura SN_{12}

14. **SN_{13} → N ORD (Nombre Ordinal)**

ER_0 = N ORD

$D_N(ER_0) =$
$[\ D_N(N)\ ORD + \alpha(N)\ D_N(ORD)\ ] =$
$\lambda\ ORD + \varnothing = ORD = ER_1$

$D_{ORD}(ER_0) =$
$[\ D_{ORD}(N)\ ORD + \alpha(N)\ D_{ORD}(ORD)\ ] = \varnothing + \varnothing = \varnothing$

$D_N(ER_1) =$
$D_N(ORD) = \varnothing$

$D_{ORD}(ER_1) =$
$D_{ORD}(ORD) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{13} es la siguiente:

$G = (\{N, ORD\}, \{ER_0, ER_1\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= N\ ER_1$
$ER_1 ::= ORD$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{N, ORD\}, \{ER_0, ER_1, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, N) = ER_1$
$f\ (ER_0, ORD) = \varnothing$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_1, ORD) = F$
$f\ (F, N) = \varnothing$
$f\ (F, ORD) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, N) = q_1$
$f\ (q_1, ORD) = q_2$

El transductor gráfico que reconoce el sintagma SN_{13} es el siguiente (Fig. 6.16):

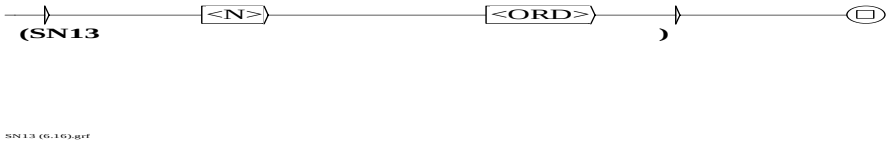


Fig. 6.16: FST gráfico que reconoce la estructura SN_{13}

15. **SN_{14} → DET N A (Determinante Nombre Adjetivo)**

ER_0 = DET N A

$D_{DET}(ER_0)=$
$D_{DET}(DET)$ N A + $\alpha(DET)$ D_{DET}(N A)=
λ N A + $\alpha(DET)$ [$D_{DET}(N)$ A + $\alpha(N)$ $D_{DET}(A)$]=
N A + $\varnothing$ [$\varnothing$ + $\varnothing$]=
N A = ER_1

$D_N(ER_0)=$
$D_N(DET)$ N A + $\alpha(DET)$ + D_N(N A)=
$\varnothing$ N A + $\alpha(DET)$ [$D_N(N)$ A + $\alpha(N)$ $D_N(A)$]=
$\varnothing$ + $\varnothing$ [λ A + $\varnothing$ $\varnothing$] = $\varnothing$

$D_A(ER_0)=$
$D_A(DET)$ N A + $\alpha(DET)$ D_A(N A)=
$\varnothing$ N A + $\alpha(DET)$ [$D_A(N)$ A + $\alpha(N)$ $D_A(A)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ A + $\varnothing$ λ] = $\varnothing$

$D_{DET}(ER_1)=$
$D_{DET}(N)$ A + $\alpha(N)$ $D_{DET}(A)$=
$\varnothing$ A + $\varnothing$ $\varnothing$ = $\varnothing$

$D_N(ER_1)=$
$D_N(N)$ A + $\alpha(N)$ $D_N(A)$=
λ A + $\varnothing$ $\varnothing$ =
A = ER_2

$D_A(ER_1)=$
$D_A(N)$ A + $\alpha(N)$ $D_A(A)$=
$\varnothing$ A + $\varnothing$ λ = $\varnothing$

$D_{DET}(ER_2) =$
$D_{DET}(A) = \varnothing$

$D_{N}(ER_2) =$
$D_{N}(A) = \varnothing$

$D_{A}(ER_2) =$
$D_{A}(A) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{14} es la siguiente:

$G = (\{DET, N, A\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= N\ ER_2$
$ER_2 ::= A$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DET, N, A\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, DET) = ER_1$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_0, A) = \varnothing$
$f\ (ER_1, DET) = \varnothing$
$f\ (ER_1, N) = ER_2$
$f\ (ER_1, A) = \varnothing$
$f\ (ER_2, DET) = \varnothing$
$f\ (ER_2, N) = \varnothing$
$f\ (ER_2, A) = F$
$f\ (F, DET) = \varnothing$
$f\ (F, N) = \varnothing$
$f\ (F, A) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, DET) = q_1$
$f\ (q_1, N) = q_2$
$f\ (q_2, A) = q_3$

El transductor gráfico que reconoce el sintagma SN_{14} es el siguiente (Fig. 6.17):

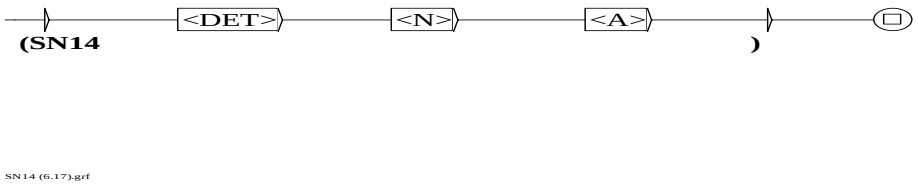


Fig. 6.17: FST gráfico que reconoce la estructura SN_{14}

16. **SN_{15} → DET A N (Determinante Adjetivo Nombre)**

ER_0 = DET A N

$D_{DET}(ER_0)=$
$D_{DET}(DET)$ A N + $\alpha(DET)$ $D_{DET}(A\ N)=$
λ A N + $\alpha(DET)$ [$D_{DET}(A)$ N + $\alpha(A)$ $D_{DET}(N)$]=
A N + $\varnothing$ [$\varnothing$ + $\varnothing$]=
A N = ER_1

$D_A(ER_0)=$
$D_A(DET)$ A N + $\alpha(DET)$ $D_A(A\ N)=$
$\varnothing$ A N + $\alpha(DET)$ [$D_A(A)$ N + $\alpha(A)$ $D_A(N)$]=
$\varnothing$ + $\varnothing$ [λ N + $\varnothing$ $\varnothing$] = $\varnothing$

$D_N(ER_0)=$
$D_N(DET)$ A N + $\alpha(DET)$ + $D_N(A\ N)=$
$\varnothing$ A N + $\alpha(DET)$ [$D_N(A)$ N + $\alpha(A)$ $D_N(N)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ N + $\varnothing$ λ] = $\varnothing$

$D_{DET}(ER_1) =$
$D_{DET}(A)\ N + \alpha(A)\ D_{DET}(N) =$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_A(ER_1) =$
$D_A(A)\ N + \alpha(A)\ D_A(N) =$
$\lambda\ N + \varnothing\ \varnothing =$
$N = ER_2$

$D_N(ER_1) =$
$D_N(A)\ N + \alpha(A)\ D_N(N) =$
$\varnothing\ N + \varnothing\ \lambda = \varnothing$

$D_{DET}(ER_2) =$
$D_{DET}(N) = \varnothing$

$D_A(ER_2) =$
$D_A(N) = \varnothing$

$D_N(ER_2) =$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{15} es la siguiente:

$G = (\{DET, A, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= A\ ER_2$
$ER_2 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, DET) = ER_1$
$f\ (ER_0, A) = \varnothing$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_1, DET) = \varnothing$
$f\ (ER_1, A) = ER_2$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_2, DET) = \varnothing$
$f\ (ER_2, A) = \varnothing$
$f\ (ER_2, N) = F$
$f\ (F, DET) = \varnothing$
$f\ (F, A) = \varnothing$
$f\ (F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, DET) = q_1$
$f\ (q_1, A) = q_2$
$f\ (q_2, N) = q_3$

El transductor gráfico que reconoce el sintagma SN_{15} es el siguiente (Fig. 6.18):

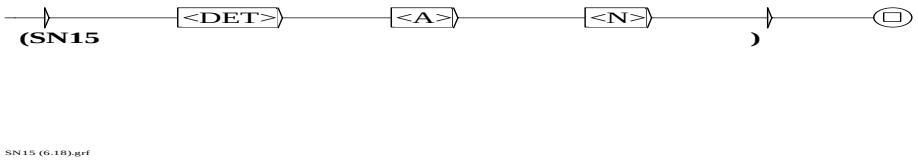


Fig. 6.18: FST gráfico que reconoce la estructura SN_{15}

17. **SN_{16} → CUANT N A (Cuantificador Nombre Adjetivo)**

ER_0 = CUANT N A

$D_{CUANT}(ER_0)=$
$D_{CUANT}(CUANT)\ N\ A + \alpha(CUANT)\ D_{CUANT}(N\ A)=$
$\lambda\ N\ A + \alpha(CUANT)\ [\ D_{CUANT}(N)\ A + \alpha(N)\ D_{CUANT}(A)\ ]=$
$N\ A + \varnothing\ [\varnothing + \varnothing]=$
$N\ A = ER_1$

$D_N(ER_0)=$
$D_N(CUANT)\ N\ A + \alpha(CUANT) + D_N(N\ A)=$
$\varnothing\ N\ A + \alpha(CUANT)\ [\ D_N(N)\ A + \alpha(N)\ D_N(A)\ ]=$
$\varnothing + \varnothing\ [\lambda\ A + \varnothing\ \varnothing] = \varnothing$

$D_A(ER_0)=$
$D_A(CUANT)\ N\ A + \alpha(CUANT)\ D_A(N\ A)=$
$\varnothing\ N\ A + \alpha(CUANT)\ [\ D_A(N)\ A + \alpha(N)\ D_A(A)\ ]=$
$\varnothing + \varnothing\ [\varnothing\ A + \varnothing\ \lambda] = \varnothing$

$D_{CUANT}(ER_1)=$
$D_{CUANT}(N)\ A + \alpha(N)\ D_{CUANT}(A)=$
$\varnothing\ A + \varnothing\ \varnothing = \varnothing$

$D_N(ER_1)=$
$D_N(N)\ A + \alpha(N)\ D_N(A)=$
$\lambda\ A + \varnothing\ \varnothing=$
$A = ER_2$

$D_A(ER_1)=$
$D_A(N)\ A + \alpha(N)\ D_A(A)=$
$\varnothing\ A + \varnothing\ \lambda = \varnothing$

$D_{CUANT}(ER_2)=$
$D_{CUANT}(A) = \varnothing$

$D_N(ER_2)=$
$D_N(A) = \varnothing$

$D_A(ER_2)=$
$D_A(A) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{16} es la siguiente:

$G = (\{CUANT, N, A\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= CUANT\ ER_1$
$ER_1 ::= N\ ER_2$
$ER_2 ::= A$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{CUANT, N, A\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, CUANT) = ER_1$
$f(ER_0, N) = \varnothing$
$f(ER_0, A) = \varnothing$
$f(ER_1, CUANT) = \varnothing$
$f(ER_1, N) = ER_2$
$f(ER_1, A) = \varnothing$
$f(ER_2, CUANT) = \varnothing$
$f(ER_2, N) = \varnothing$
$f(ER_2, A) = F$
$f(F, CUANT) = \varnothing$
$f(F, N) = \varnothing$
$f(F, A) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, CUANT) = q_1$
$f(q_1, N) = q_2$
$f(q_2, A) = q_3$

El transductor gráfico que reconoce el sintagma SN_{16} es el siguiente (Fig. 6.19):

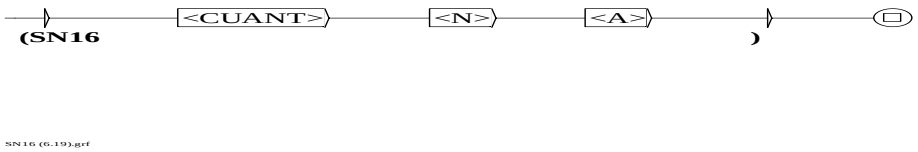


Fig. 6.19: FST gráfico que reconoce la estructura SN_{16}

18. **SN_{17} → CUANT A N (Cuantificador Adjetivo Nombre)**

$ER_0 = CUANT\ A\ N$

$D_{CUANT}(ER_0)=$
$D_{CUANT}(CUANT)\ A\ N + \alpha(CUANT)\ D_{CUANT}(A\ N)=$
$\lambda\ A\ N + \alpha(CUANT)\ [\ D_{CUANT}(A)\ N + \alpha(A)\ D_{CUANT}(N)\ ]=$
$A\ N + \varnothing\ [\varnothing + \varnothing]=$
$A\ N = ER_1$

$D_A(ER_0)=$
$D_A(CUANT)\ A\ N + \alpha(CUANT)\ D_A(A\ N)=$
$\varnothing\ A\ N + \alpha(CUANT)\ [\ D_A(A)\ N + \alpha(A)\ D_A(N)\ ]=$
$\varnothing + \varnothing\ [\lambda\ N + \varnothing\ \varnothing] = \varnothing$

$D_N(ER_0)=$
$D_N(CUANT)\ A\ N + \alpha(CUANT) + D_N(A\ N)=$
$\varnothing\ A\ N + \alpha(CUANT)\ [\ D_N(A)\ N + \alpha(A)\ D_N(N)\ ]=$
$\varnothing + \varnothing\ [\varnothing\ N + \varnothing\ \lambda] = \varnothing$

$D_{CUANT}(ER_1)=$
$D_{CUANT}(A)\ N + \alpha(A)\ D_{CUANT}(N)=$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_A(ER_1)=$
$D_A(A)\ N + \alpha(A)\ D_A(N)=$
$\lambda\ N + \varnothing\ \varnothing =$
$N = ER_2$

$D_N(ER_1)=$
$D_N(A)\ N + \alpha(A)\ D_N(N)=$
$\varnothing\ N + \varnothing\ \lambda = \varnothing$

$D_{CUANT}(ER_2)=$
$D_{CUANT}(N) = \varnothing$

$D_A(ER_2)=$
$D_A(N) = \varnothing$

$D_N(ER_2)=$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{17} es la siguiente:

$G = (\{CUANT, A, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= CUANT\ ER_1$
$ER_1 ::= A\ ER_2$
$ER_2 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{CUANT, A, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

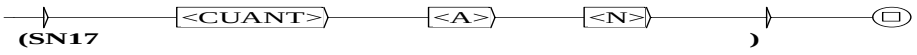
donde la función de transición, f, se define como:

$f(ER_0, CUANT) = ER_1$
$f(ER_0, A) = \varnothing$
$f(ER_0, N) = \varnothing$
$f(ER_1, CUANT) = \varnothing$
$f(ER_1, A) = ER_2$
$f(ER_1, N) = \varnothing$
$f(ER_2, CUANT) = \varnothing$
$f(ER_2, A) = \varnothing$
$f(ER_2, N) = F$
$f(F, CUANT) = \varnothing$
$f(F, A) = \varnothing$
$f(F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, CUANT) = q_1$
$f(q_1, A) = q_2$
$f(q_2, N) = q_3$

El transductor gráfico que reconoce el sintagma SN_{17} es el siguiente (Fig. 6.20):



SN17 (6.20).grf

Fig. 6.20: FST gráfico que reconoce la estructura SN_{17}

19. **SN_{18} → POS N A (Posesivo Nombre Adjetivo)**

ER_0 = POS N A

$D_{POS}(ER_0)=$
$D_{POS}(POS)\ N\ A + \alpha(POS)\ D_{POS}(N\ A)=$
$\lambda\ N\ A + \alpha(POS)\ [\ D_{POS}(N)\ A + \alpha(N)\ D_{POS}(A)\ ]=$
$N\ A + \varnothing\ [\varnothing + \varnothing]=$
$N\ A = ER_1$

$D_N(ER_0)=$
$D_N(POS)\ N\ A + \alpha(POS) + D_N(N\ A)=$
$\varnothing\ N\ A + \alpha(POS)\ [\ D_N(N)\ A + \alpha(N)\ D_N(A)\ ]=$
$\varnothing + \varnothing\ [\lambda\ A + \varnothing\ \varnothing] = \varnothing$

$D_A(ER_0)=$
$D_A(POS)\ N\ A + \alpha(POS)\ D_A(N\ A)=$
$\varnothing\ N\ A + \alpha(POS)\ [\ D_A(N)\ A + \alpha(N)\ D_A(A)\ ]=$
$\varnothing + \varnothing\ [\varnothing\ A + \varnothing\ \lambda] = \varnothing$

$D_{POS}(ER_1)=$
$D_{POS}(N)\ A + \alpha(N)\ D_{POS}(A)=$
$\varnothing\ A + \varnothing\ \varnothing = \varnothing$

$D_N(ER_1)=$
$D_N(N)\ A + \alpha(N)\ D_N(A)=$
$\lambda\ A + \varnothing\ \varnothing =$
$A = ER_2$

$D_A(ER_1)=$
$D_A(N)\ A + \alpha(N)\ D_A(A)=$
$\varnothing\ A + \varnothing\ \lambda = \varnothing$

$D_{POS}(ER_2)=$
$D_{POS}(A) = \varnothing$

$D_N(ER_2)=$
$D_N(A) = \varnothing$

$D_A(ER_2)=$
$D_A(A) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{18} es la siguiente:

$G = (\{POS, N, A\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= POS\ ER_1$
$ER_1 ::= N\ ER_2$
$ER_2 ::= A$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{POS, N, A\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, POS) = ER_1$
$f(ER_0, N) = \varnothing$
$f(ER_0, A) = \varnothing$
$f(ER_1, POS) = \varnothing$
$f(ER_1, N) = ER_2$
$f(ER_1, A) = \varnothing$
$f(ER_2, POS) = \varnothing$
$f(ER_2, N) = \varnothing$
$f(ER_2, A) = F$
$f(F, POS) = \varnothing$
$f(F, N) = \varnothing$
$f(F, A) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, POS) = q_1$
$f(q_1, N) = q_2$
$f(q_2, A) = q_3$

El transductor gráfico que reconoce el sintagma SN_{18} es el siguiente (Fig. 6.21):

SN18 (6.21).grf

Fig. 6.21: FST gráfico que reconoce la estructura SN_{18}

20. **SN_{19} → POS A N (Posesivo Adjetivo Nombre)**

$ER_0 = POS\ A\ N$

$D_{POS}(ER_0)=$
$D_{POS}(POS)\ A\ N + \alpha(POS)\ D_{POS}(A\ N)=$
$\lambda\ A\ N + \alpha(POS)\ [\ D_{POS}(A)\ N + \alpha(A)\ D_{POS}(N)\ ]=$
$A\ N + \varnothing\ [\varnothing + \varnothing]=$
$A\ N = ER_1$

$D_A(ER_0)=$
$D_A(POS)\ A\ N + \alpha(POS)\ D_A(A\ N)=$
$\varnothing\ A\ N + \alpha(POS)\ [\ D_A(A)\ N + \alpha(A)\ D_A(N)\ ]=$
$\varnothing + \varnothing\ [\lambda\ N + \varnothing\ \varnothing] = \varnothing$

$D_N(ER_0)=$
$D_N(POS)\ A\ N + \alpha(POS) + D_N(A\ N)=$
$\varnothing\ A\ N + \alpha(POS)\ [\ D_N(A)\ N + \alpha(A)\ D_N(N)\ ]=$
$\varnothing + \varnothing\ [\varnothing\ N + \varnothing\ \lambda] = \varnothing$

$D_{POS}(ER_1)=$
$D_{POS}(A)\ N + \alpha(A)\ D_{POS}(N)=$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_A(ER_1)=$
$D_A(A)\ N + \alpha(A)\ D_A(N)=$
$\lambda\ N + \varnothing\ \varnothing =$
$N = ER_2$

$D_N(ER_1)=$
$D_N(A)\ N + \alpha(A)\ D_N(N)=$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_{POS}(ER_2)=$
$D_{POS}(N) = \varnothing$

$D_A(ER_2)=$
$D_A(N) = \varnothing$

$D_N(ER_2)=$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{19} es la siguiente:

$G = (\{POS, A, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= POS\ ER_1$
$ER_1 ::= A\ ER_2$
$ER_2 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{POS, A, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

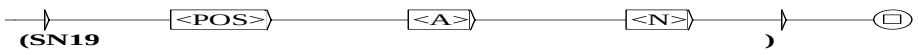
donde la función de transición, f, se define como:

$f(ER_0, POS) = ER_1$
$f(ER_0, A) = \varnothing$
$f(ER_0, N) = \varnothing$
$f(ER_1, POS) = \varnothing$
$f(ER_1, A) = ER_2$
$f(ER_1, N) = \varnothing$
$f(ER_2, POS) = \varnothing$
$f(ER_2, A) = \varnothing$
$f(ER_2, N) = F$
$f(F, POS) = \varnothing$
$f(F, A) = \varnothing$
$f(F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, POS) = q_1$
$f(q_1, A) = q_2$
$f(q_2, N) = q_3$

El transductor gráfico que reconoce el sintagma SN_{19} es el siguiente (Fig. 6.22):



SN19 (6.22).grf

Fig. 6.22: FST gráfico que reconoce la estructura SN_{19}

21. **SN_{20} → DEM N A (Demostrativo Nombre Adjetivo)**

ER_0 = DEM N A

$D_{DEM}(ER_0)=$
$D_{DEM}(DEM)\ N\ A + \alpha(DEM)\ D_{DEM}(N\ A)=$
$\lambda\ N\ A + \alpha(DEM)\ [\ D_{DEM}(N)\ A + \alpha(N)\ D_{DEM}(A)\ ]=$
$N\ A + \varnothing\ [\varnothing + \varnothing]=$
$N\ A = ER_1$

$D_N(ER_0)=$
$D_N(DEM)\ N\ A + \alpha(DEM) + D_N(N\ A)=$
$\varnothing\ N\ A + \alpha(DEM)\ [\ D_N(N)\ A + \alpha(N)\ D_N(A)\ ]=$
$\varnothing + \varnothing\ [\lambda\ A + \varnothing\ \varnothing] = \varnothing$

$D_A(ER_0)=$
$D_A(DEM)\ N\ A + \alpha(DEM)\ D_A(N\ A)=$
$\varnothing\ N\ A + \alpha(DEM)\ [\ D_A(N)\ A + \alpha(N)\ D_A(A)\ ]=$
$\varnothing + \varnothing\ [\varnothing\ A + \varnothing\ \lambda] = \varnothing$

$D_{DEM}(ER_1)=$
$D_{DEM}(N)\ A + \alpha(N)\ D_{DEM}(A)=$
$\varnothing\ A + \varnothing\ \varnothing = \varnothing$

$D_N(ER_1)=$
$D_N(N)\ A + \alpha(N)\ D_N(A)=$
$\lambda\ A + \varnothing\ \varnothing =$
$A = ER_2$

$D_A(ER_1)=$
$D_A(N)\ A + \alpha(N)\ D_A(A)=$
$\varnothing\ A + \varnothing\ \lambda = \varnothing$

$D_{DEM}(ER_2)=$
$D_{DEM}(A) = \varnothing$

$D_N(ER_2)=$
$D_N(A) = \varnothing$

$D_A(ER_2)=$
$D_A(A) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{20} es la siguiente:

$G = (\{DEM, N, A\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DEM\ ER_1$
$ER_1 ::= N\ ER_2$
$ER_2 ::= A$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$$AF = (\{DEM, N, A\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$$

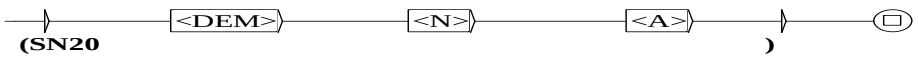
donde la función de transición, f, se define como:

$f(ER_0, DEM) = ER_1$
$f(ER_0, N) = \varnothing$
$f(ER_0, A) = \varnothing$
$f(ER_1, DEM) = \varnothing$
$f(ER_1, N) = ER_2$
$f(ER_1, A) = \varnothing$
$f(ER_2, DEM) = \varnothing$
$f(ER_2, N) = \varnothing$
$f(ER_2, A) = F$
$f(F, DEM) = \varnothing$
$f(F, N) = \varnothing$
$f(F, A) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, DEM) = q_1$
$f(q_1, N) = q_2$
$f(q_2, A) = q_3$

El transductor gráfico que reconoce el sintagma SN_{20} es el siguiente (Fig. 6.23):



SN20 (6.23).grf

Fig. 6.23: FST gráfico que reconoce la estructura SN_{20}

22. **SN_{21} → DEM A N (Demostrativo Adjetivo Nombre)**

ER_0 = DEM A N

$D_{DEM}(ER_0)=$
$D_{DEM}(DEM)$ A N + $\alpha(DEM)$ D_{DEM}(A N)=
λ A N + $\alpha(DEM)$ [$D_{DEM}(A)$ N + $\alpha(A)$ $D_{DEM}(N)$]=
A N + $\varnothing$ [$\varnothing$ + $\varnothing$]=
A N = ER_1

$D_A(ER_0)=$
$D_A(DEM)$ A N + $\alpha(DEM)$ D_A(A N)=
$\varnothing$ A N + $\alpha(DEM)$ [$D_A(A)$ N + $\alpha(A)$ $D_A(N)$]=
$\varnothing$ + $\varnothing$ [λ N + $\varnothing$ $\varnothing$] = $\varnothing$

$D_N(ER_0)=$
$D_N(DEM)$ A N + $\alpha(DEM)$ + D_N(A N)=
$\varnothing$ A N + $\alpha(DEM)$ [$D_N(A)$ N + $\alpha(A)$ $D_N(N)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ N + $\varnothing$ λ] = $\varnothing$

$D_{DEM}(ER_1)=$
$D_{DEM}(A)$ N + $\alpha(A)$ $D_{DEM}(N)$=
$\varnothing$ N + $\varnothing$ $\varnothing$ = $\varnothing$

$D_A(ER_1)=$
$D_A(A)$ N + $\alpha(A)$ $D_A(N)$=
λ N + $\varnothing$ $\varnothing$ =
N = ER_2

$D_N(ER_1)=$
$D_N(A)$ N + $\alpha(A)$ $D_N(N)$=
$\varnothing$ N + $\varnothing$ λ = $\varnothing$

$D_{DEM}(ER_2)=$
$D_{DEM}(N)$ = $\varnothing$

$D_A(ER_2)=$
$D_A(N)$ = $\varnothing$

$D_N(ER_2)=$
$D_N(N)$ = λ

La *Gramática Regular* que reconoce el sintagma SN_{21} es la siguiente:

$G = (\{DEM, A, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

ER_0 ::= DEM ER_1
ER_1 ::= A ER_2
ER_2 ::= N

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DEM, A, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, DEM) = ER_1$
$f(ER_0, A) = \varnothing$
$f(ER_0, N) = \varnothing$
$f(ER_1, DEM) = \varnothing$
$f(ER_1, A) = ER_2$
$f(ER_1, N) = \varnothing$
$f(ER_2, DEM) = \varnothing$
$f(ER_2, A) = \varnothing$
$f(ER_2, N) = F$
$f(F, DEM) = \varnothing$
$f(F, A) = \varnothing$
$f(F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, DEM) = q_1$
$f(q_1, A) = q_2$
$f(q_2, N) = q_3$

El transductor gráfico que reconoce el sintagma SN_{21} es el siguiente (Fig. 6.24):

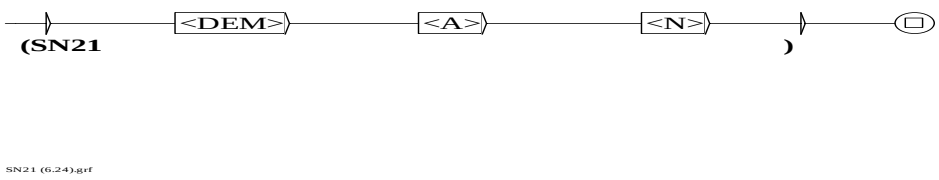


Fig. 6.24: FST gráfico que reconoce la estructura SN_{21}

23. **SN_{22} → DET N PA (Determinante Nombre Participio - Adj)**

ER_0 = DET N PA

$D_{DET}(ER_0)=$
$D_{DET}(DET)$ N PA + $\alpha(DET)$ $D_{DET}(N\ PA)=$
λ N PA + $\alpha(DET)$ [$D_{DET}(N)$ PA + $\alpha(N)$ $D_{DET}(PA)$]=
N PA + $\varnothing$ [$\varnothing$ + $\varnothing$]=
N PA = ER_1

$D_N(ER_0)=$
$D_N(DET)$ N PA + $\alpha(DET)$ + $D_N(N\ PA)=$
$\varnothing$ N PA + $\alpha(DET)$ [$D_N(N)$ PA + $\alpha(N)$ $D_N(PA)$]=
$\varnothing$ + $\varnothing$ [λ PA + $\varnothing$ $\varnothing$] = $\varnothing$

$D_{PA}(ER_0)=$
$D_{PA}(DET)$ N PA + $\alpha(DET)$ $D_{PA}(N\ PA)=$
$\varnothing$ N PA + $\alpha(DET)$ [$D_{PA}(N)$ PA + $\alpha(N)$ $D_{PA}(PA)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ A + $\varnothing$ λ] = $\varnothing$

$D_{DET}(ER_1)=$
$D_{DET}(N)$ PA + $\alpha(N)$ $D_{DET}(PA)=$
$\varnothing$ A + $\varnothing$ $\varnothing$ = $\varnothing$

$D_N(ER_1)=$
$D_N(N)$ PA + $\alpha(N)$ $D_N(PA)=$
λ PA + $\varnothing$ $\varnothing$ =
A = ER_2

$D_A(ER_1)=$
$D_{PA}(N)$ PA + $\alpha(N)$ $D_{PA}(PA)=$
$\varnothing$ PA + $\varnothing$ λ = $\varnothing$

$D_{DET}(ER_2)=$
$D_{DET}(PA)$ = $\varnothing$

$D_N(ER_2)=$
$D_N(PA)$ = $\varnothing$

$D_A(ER_2)=$
$D_A(PA)=\lambda$

La *Gramática Regular* que reconoce el sintagma SN_{22} es la siguiente:

$G = (\{DET, N, PA\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

ER_0 ::= DET ER_1
ER_1 ::= N ER_2
ER_2 ::= PA

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DET, N, PA\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, DET) = ER_1$
$f(ER_0, N) = \varnothing$
$f(ER_0, PA) = \varnothing$
$f(ER_1, DET) = \varnothing$
$f(ER_1, N) = ER_2$
$f(ER_1, PA) = \varnothing$
$f(ER_2, DET) = \varnothing$
$f(ER_2, N) = \varnothing$
$f(ER_2, PA) = F$
$f(F, DET) = \varnothing$
$f(F, N) = \varnothing$
$f(F, PA) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, DET) = q_1$
$f(q_1, N) = q_2$
$f(q_2, PA) = q_3$

El transductor gráfico que reconoce el sintagma SN_{22} es el siguiente (Fig. 6.25):

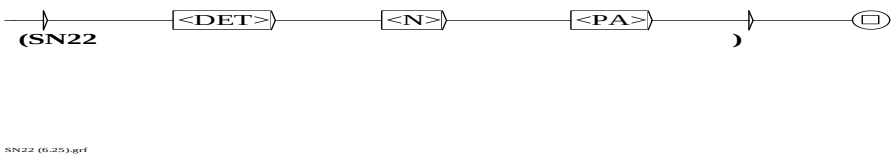


Fig. 6.25: FST gráfico que reconoce la estructura SN_{22}

24. **SN_{23} → DET PA N (Determinante Participio - Adj Nombre)**

ER_0 = DET PA N

$D_{DET}(ER_0)=$
$D_{DET}(DET)$ PA N + $\alpha(DET)$ $D_{DET}(PA\ N)=$
λ PA N + $\alpha(DET)$ [$D_{DET}(PA)$ N + $\alpha(PA)$ $D_{DET}(N)$]=
PA N + $\varnothing$ [$\varnothing$ + $\varnothing$]=
PA N = ER_1

$D_{PA}(ER_0)=$
$D_{PA}(DET)$ PA N + $\alpha(DET)$ $D_{PA}(PA\ N)=$
$\varnothing$ PA N + $\alpha(DET)$ [$D_{PA}(PA)$ N + $\alpha(PA)$ $D_{PA}(N)$]=
$\varnothing$ + $\varnothing$ [λ N + $\varnothing$ $\varnothing$] = $\varnothing$

$D_N(ER_0)=$
$D_N(DET)$ PA N + $\alpha(DET)$ + $D_N(PA\ N)=$
$\varnothing$ PA N + $\alpha(DET)$ [$D_N(PA)$ N + $\alpha(PA)$ $D_N(N)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ N + $\varnothing$ λ] = $\varnothing$

$D_{DET}(ER_1)=$
$D_{DET}(PA)$ N + $\alpha(PA)$ $D_{DET}(N)=$
$\varnothing$ N + $\varnothing$ $\varnothing$ = $\varnothing$

$D_A(ER_1)=$
$D_{PA}(PA)$ N + $\alpha(PA)$ $D_{PA}(N)=$
λ N + $\varnothing$ $\varnothing$ =
N = ER_2

$D_N(ER_1)=$
$D_N(PA)$ N + $\alpha(PA)$ $D_N(N)=$
$\varnothing$ N + $\varnothing$ λ = $\varnothing$

$D_{DET}(ER_2)=$
$D_{DET}(N)$ = $\varnothing$

$D_{PA}(ER_2)=$
$D_{PA}(N)=\varnothing$

$D_N(ER_2)=$
$D_N(N)$ = λ

La *Gramática Regular* que reconoce el sintagma SN_{23} es la siguiente:

$G = (\{DET, A, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

ER_0 ::= DET ER_1
ER_1 ::= PA ER_2
ER_2 ::= N

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DET, PA, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f , se define como:

$f\ (ER_0, DET) = ER_1$
$f\ (ER_0, PA) = \varnothing$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_1, DET) = \varnothing$
$f\ (ER_1, PA) = ER_2$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_2, DET) = \varnothing$
$f\ (ER_2, PA) = \varnothing$
$f\ (ER_2, N) = F$
$f\ (F, DET) = \varnothing$
$f\ (F, PA) = \varnothing$
$f\ (F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, DET) = q_1$
$f\ (q_1, PA) = q_2$
$f\ (q_2, N) = q_3$

El transductor gráfico que reconoce el sintagma SN_{23} es el siguiente (Fig. 6.26):

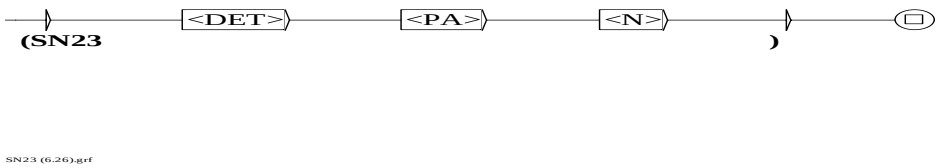


Fig. 6.26: FST gráfico que reconoce la estructura SN_{23}

25. **SN_{24} → CUANT N PA (Cuantificador Nombre Participio - Adj)**

ER_0 = CUANT N PA

$D_{CUANT}(ER_0)=$
$D_{CUANT}(CUANT)$ N PA + $\alpha(CUANT)$ $D_{CUANT}(N\ PA)=$
λ N PA + $\alpha(CUANT)$ [$D_{CUANT}(N)$ PA + $\alpha(N)$ $D_{CUANT}(PA)$]=
N PA + $\varnothing$ [$\varnothing$ + $\varnothing$]=
N PA = ER_1

$D_N(ER_0)=$
$D_N(CUANT)$ N PA + $\alpha(CUANT)$ + $D_N(N\ PA)=$
$\varnothing$ N PA + $\alpha(CUANT)$ [$D_N(N)$ PA + $\alpha(N)$ $D_N(PA)$]=
$\varnothing$ + $\varnothing$ [λ PA + $\varnothing$ $\varnothing$] = $\varnothing$

$D_A(ER_0)=$
$D_A(CUANT)$ N PA + $\alpha(CUANT)$ $D_A(N\ PA)=$
$\varnothing$ N PA + $\alpha(CUANT)$ [$D_A(N)$ PA + $\alpha(N)$ $D_{PA}(PA)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ PA + $\varnothing$ λ] = $\varnothing$

$D_{CUANT}(ER_1)=$
$D_{CUANT}(N)$ PA + $\alpha(N)$ $D_{CUANT}(PA)=$
$\varnothing$ PA + $\varnothing$ $\varnothing$ = $\varnothing$

$D_N(ER_1)=$
$D_N(N)$ PA + $\alpha(N)$ $D_N(PA)=$
λ PA + $\varnothing$ $\varnothing$ =
PA = ER_2

$D_{PA}(ER_1)=$
$D_{PA}(N)$ PA + $\alpha(N)$ $D_{PA}(PA)=$
$\varnothing$ PA + $\varnothing$ λ = $\varnothing$

$D_{CUANT}(ER_2)=$
$D_{CUANT}(PA)$ = $\varnothing$

$D_N(ER_2)=$
$D_N(PA)$ = $\varnothing$

$D_{PA}(ER_2)=$
$D_{PA}(PA)$ = λ

La *Gramática Regular* que reconoce el sintagma SN_{24} es la siguiente:

$G = (\{CUANT, N, PA\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

ER_0 ::= CUANT ER_1
ER_1 ::= N ER_2
ER_2 ::= PA

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{CUANT, N, PA\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f , se define como:

$f(ER_0, CUANT) = ER_1$
$f(ER_0, N) = \varnothing$
$f(ER_0, PA) = \varnothing$
$f(ER_1, CUANT) = \varnothing$
$f(ER_1, N) = ER_2$
$f(ER_1, PA) = \varnothing$
$f(ER_2, CUANT) = \varnothing$
$f(ER_2, N) = \varnothing$
$f(ER_2, PA) = F$
$f(F, CUANT) = \varnothing$
$f(F, N) = \varnothing$
$f(F, PA) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, CUANT) = q_1$
$f(q_1, N) = q_2$
$f(q_2, PA) = q_3$

El transductor gráfico que reconoce el sintagma SN_{24} es el siguiente (Fig. 6.27):

SN24 (6.27).grf

Fig. 6.27: FST gráfico que reconoce la estructura SN_{24}

26. **SN_{25} → CUANT PA N (Cuantificador Participio - Adj Nombre)**

ER_0 = CUANT PA N

$D_{CUANT}(ER_0)=$
$D_{CUANT}(CUANT)$ PA N + $\alpha(CUANT)$ D_{CUANT}(PA N)=
λ PA N + $\alpha(CUANT)$ [$D_{CUANT}(PA)$ N + $\alpha(PA)$ $D_{CUANT}(N)$]=
PA N + $\varnothing$ [$\varnothing$ + $\varnothing$]=
PA N = ER_1

$D_{PA}(ER_0)=$
$D_{PA}(CUANT)$ PA N + $\alpha(CUANT)$ D_{PA}(PA N)=
$\varnothing$ PA N + $\alpha(CUANT)$ [$D_{PA}(PA)$ N + $\alpha(PA)$ $D_{PA}(N)$]=
$\varnothing$ + $\varnothing$ [λ N + $\varnothing$ $\varnothing$] = $\varnothing$

$D_N(ER_0)=$
$D_N(CUANT)$ PA N + $\alpha(CUANT)$ + D_N(PA N)=
$\varnothing$ PA N + $\alpha(CUANT)$ [$D_N(PA)$ N + $\alpha(PA)$ $D_N(N)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ N + $\varnothing$ λ] = $\varnothing$

$D_{CUANT}(ER_1)=$
$D_{CUANT}(PA)$ N + $\alpha(PA)$ $D_{CUANT}(N)$=
$\varnothing$ N + $\varnothing$ $\varnothing$ = $\varnothing$

$D_{PA}(ER_1)=$
$D_{PA}(PA)$ N + $\alpha(PA)$ $Dp_A(N)$=
λ N + $\varnothing$ $\varnothing$ =
N = ER_2

$D_N(ER_1)=$
$D_N(PA)$ N + $\alpha(PA)$ $D_N(N)$=
$\varnothing$ N + $\varnothing$ λ = $\varnothing$

$D_{CUANT}(ER_2)=$
$D_{CUANT}(N)$ = $\varnothing$

$D_{PA}(ER_2)=$
$D_{PA}(N)$ = $\varnothing$

$D_N(ER_2)=$
$D_N(N)$ = λ

La *Gramática Regular* que reconoce el sintagma SN_{25} es la siguiente:

$G = (\{CUANT, PA, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

ER_0 ::= CUANT ER_1
ER_1 ::= PA ER_2
ER_2 ::= N

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{CUANT, PA, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f , se define como:

$f(ER_0, CUANT) = ER_1$
$f(ER_0, PA) = \varnothing$
$f(ER_0, N) = \varnothing$
$f(ER_1, CUANT) = \varnothing$
$f(ER_1, PA) = ER_2$
$f(ER_1, N) = \varnothing$
$f(ER_2, CUANT) = \varnothing$
$f(ER_2, PA) = \varnothing$
$f(ER_2, N) = F$
$f(F, CUANT) = \varnothing$
$f(F, N) = \varnothing$
$f(F, PA) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, CUANT) = q_1$
$f(q_1, PA) = q_2$
$f(q_2, N) = q_3$

El transductor gráfico que reconoce el sintagma SN_{25} es el siguiente (Fig. 6.28):

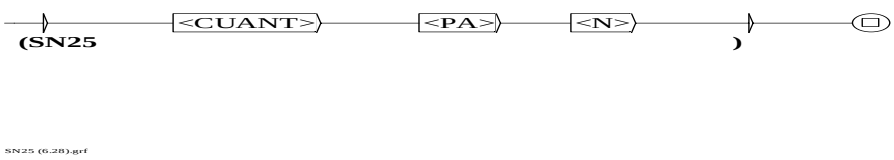


Fig. 6.28: FST gráfico que reconoce la estructura SN_{25}

27. **SN_{26} → POS N PA (Posesivo Nombre Parcicipio - Adj)**

ER_0 = POS N PA

$D_{POS}(ER_0)=$
$D_{POS}(POS)$ N PA + $\alpha(POS)$ $D_{POS}(N\ PA)=$
λ N PA + $\alpha(POS)$ [$D_{POS}(N)$ PA + $\alpha(N)$ $D_{POS}(PA)$]=
N PA + $\varnothing$ [$\varnothing$ + $\varnothing$]=
N PA = ER_1

$D_N(ER_0)=$
$D_N(POS)$ N PA + $\alpha(POS)$ + $D_N(N\ PA)=$
$\varnothing$ N PA + $\alpha(POS)$ [$D_N(N)$ PA + $\alpha(N)$ $D_N(PA)$]=
$\varnothing$ + $\varnothing$ [λ PA + $\varnothing$ $\varnothing$] = $\varnothing$

$D_{PA}(ER_0)=$
$D_{PA}(POS)$ N PA + $\alpha(POS)$ $D_{PA}(N\ PA)=$
$\varnothing$ N PA + $\alpha(POS)$ [$D_A(N)$ PA + $\alpha(N)$ $D_{PA}(PA)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ PA + $\varnothing$ λ] = $\varnothing$

$D_{POS}(ER_1)=$
$D_{POS}(N)$ PA + $\alpha(N)$ $D_{POS}(PA)=$
$\varnothing$ A + $\varnothing$ $\varnothing$ = $\varnothing$

$D_N(ER_1)=$
$D_N(N)$ PA + $\alpha(N)$ $D_N(PA)=$
λ PA + $\varnothing$ $\varnothing$ =
PA = ER_2

$D_{PA}(ER_1)=$
$D_{PA}(N)$ PA + $\alpha(N)$ $D_{PA}(PA)=$
$\varnothing$ PA + $\varnothing$ λ = $\varnothing$

$D_{POS}(ER_2)=$
$D_{POS}(A) = \varnothing$

$D_N(ER_2)=$
$D_N(PA) = \varnothing$

$D_{PA}(ER_2)=$
$D_{PA}(PA) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{26} es la siguiente:

$G = (\{POS, N, PA\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

ER_0 ::= POS ER_1
ER_1 ::= N ER_2
ER_2 ::= PA

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{POS, N, PA\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, POS) = ER_1$
$f(ER_0, N) = \varnothing$
$f(ER_0, PA) = \varnothing$
$f(ER_1, POS) = \varnothing$
$f(ER_1, N) = ER_2$
$f(ER_1, PA) = \varnothing$
$f(ER_2, POS) = \varnothing$
$f(ER_2, N) = \varnothing$
$f(ER_2, PA) = F$
$f(F, POS) = \varnothing$
$f(F, N) = \varnothing$
$f(F, PA) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, POS) = q_1$
$f(q_1, N) = q_2$
$f(q_2, PA) = q_3$

El transductor gráfico que reconoce el sintagma SN_{26} es el siguiente (Fig. 6.29):

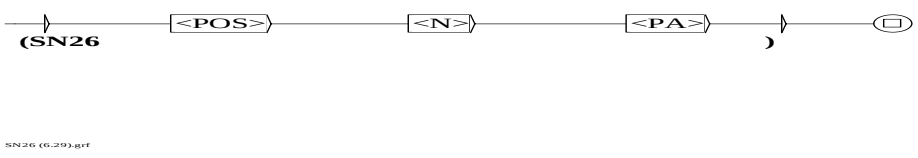


Fig. 6.29: FST gráfico que reconoce la estructura SN_{26}

28. **SN_{27} → POS PA N (Posesivo Participio- Adj Nombre)**

ER_0 = POS PA N

$D_{POS}(ER_0)=$
$D_{POS}(POS)$ PA N + $\alpha(POS)$ $D_{POS}(PA\ N)=$
λ PA N + $\alpha(POS)$ [$D_{POS}(PA)$ N + $\alpha(PA)$ $D_{POS}(N)$]=
PA N + $\varnothing$ [$\varnothing$ + $\varnothing$]=
PA N = ER_1

$D_{PA}(ER_0)=$
$D_{PA}(POS)$ PA N + $\alpha(POS)$ $D_{PA}(PA\ N)=$
$\varnothing$ PA N + $\alpha(POS)$ [$D_{PA}(PA)$ N + $\alpha(PA)$ $D_{PA}(N)$]=
$\varnothing$ + $\varnothing$ [λ N + $\varnothing$ $\varnothing$] = $\varnothing$

$D_N(ER_0)=$
$D_N(POS)$ PA N + $\alpha(POS)$ + $D_N(PA\ N)=$
$\varnothing$ PA N + $\alpha(POS)$ [$D_N(PA)$ N + $\alpha(PA)$ $D_N(N)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ N + $\varnothing$ λ] = $\varnothing$

$D_{POS}(ER_1)=$
$D_{POS}(PA)$ N + $\alpha(PA)$ $D_{POS}(N)=$
$\varnothing$ N + $\varnothing$ $\varnothing$ = $\varnothing$

$D_{PA}(ER_1)=$
$D_{PA}(PA)$ N + $\alpha(PA)$ $D_{PA}(N)=$
λ N + $\varnothing$ $\varnothing$ =
N = ER_2

$D_N(ER_1)=$
$D_N(PA)$ N + $\alpha(PA)$ $D_N(N)=$
$\varnothing$ N + $\varnothing$ $\varnothing$ = $\varnothing$

$D_{POS}(ER_2)=$
$D_{POS}(N)$ = $\varnothing$

$D_{PA}(ER_2)=$
$D_{PA}(N)$ = $\varnothing$

$D_N(ER_2)=$
$D_N(N)$ = λ

La *Gramática Regular* que reconoce el sintagma SN_{27} es la siguiente:

$G = (\{POS, PA, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

ER_0 ::= POS ER_1
ER_1 ::= PA ER_2
ER_2 ::= N

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{POS, PA, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f , se define como:

$f(ER_0, POS) = ER_1$
$f(ER_0, PA) = \varnothing$
$f(ER_0, N) = \varnothing$
$f(ER_1, POS) = \varnothing$
$f(ER_1, PA) = ER_2$
$f(ER_1, N) = \varnothing$
$f(ER_2, POS) = \varnothing$
$f(ER_2, PA) = \varnothing$
$f(ER_2, N) = F$
$f(F, POS) = \varnothing$
$f(F, PA) = \varnothing$
$f(F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, POS) = q_1$
$f(q_1, PA) = q_2$
$f(q_2, N) = q_3$

El transductor gráfico que reconoce el sintagma SN_{27} es el siguiente (Fig. 6.30):

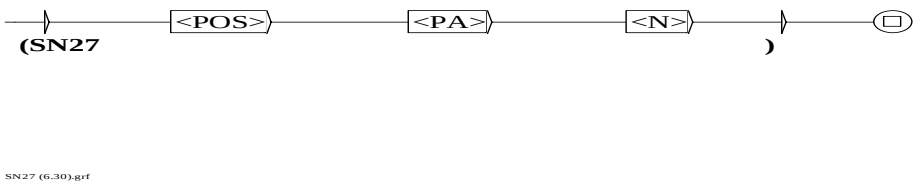


Fig. 6.30: FST gráfico que reconoce la estructura SN_{27}

29. **SN_{28} → DEM N PA (Demostrativo Nombre Participio - Adj)**

ER_0 = DEM N PA

$D_{DEM}(ER_0)=$
$D_{DEM}(DEM)$ N PA + $\alpha(DEM)$ $D_{DEM}(N\ PA)=$
λ N PA + $\alpha(DEM)$ [$D_{DEM}(N)$ PA + $\alpha(N)$ $D_{DEM}(PA)$]=
N PA + $\varnothing$ [$\varnothing$ + $\varnothing$]=
N PA = ER_1

$D_N(ER_0)=$
$D_N(DEM)$ N PA + $\alpha(DEM)$ + $D_N(N\ PA)=$
$\varnothing$ N PA + $\alpha(DEM)$ [$D_N(N)$ PA + $\alpha(N)$ $D_N(PA)$]=
$\varnothing$ + $\varnothing$ [λ PA + $\varnothing$ $\varnothing$] = $\varnothing$

$D_{PA}(ER_0)=$
$D_{PA}(DEM)$ N PA + $\alpha(DEM)$ $D_{PA}(N\ PA)=$
$\varnothing$ N PA + $\alpha(DEM)$ [$D_{PA}(N)$ PA + $\alpha(N)$ $D_{PA}(PA)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ PA + $\varnothing$ λ] = $\varnothing$

$D_{DEM}(ER_1)=$
$D_{DEM}(N)$ PA + $\alpha(N)$ $D_{DEM}(PA)=$
$\varnothing$ A + $\varnothing$ $\varnothing$ = $\varnothing$

$D_N(ER_1)=$
$D_N(N)$ PA + $\alpha(N)$ $D_N(PA)=$
λ PA + $\varnothing$ $\varnothing$ =
PA = ER_2

$D_{PA}(ER_1)=$
$D_{PA}(N)$ PA + $\alpha(N)$ $D_{PA}(PA)=$
$\varnothing$ PA + $\varnothing$ λ = $\varnothing$

$D_{DEM}(ER_2)=$
$D_{DEM}(PA) = \varnothing$

$D_N(ER_2)=$
$D_N(PA) = \varnothing$

$D_{PA}(ER_2)=$
$D_{PA}(PA) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{28} es la siguiente:

$G = (\{DEM, N, PA\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DEM\ ER_1$
$ER_1 ::= N\ ER_2$
$ER_2 ::= PA$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DEM, N, PA\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, DEM) = ER_1$
$f(ER_0, N) = \varnothing$
$f(ER_0, PA) = \varnothing$
$f(ER_1, DEM) = \varnothing$
$f(ER_1, N) = ER_2$
$f(ER_1, PA) = \varnothing$
$f(ER_2, DEM) = \varnothing$
$f(ER_2, N) = \varnothing$
$f(ER_2, PA) = F$
$f(F, DEM) = \varnothing$
$f(F, N) = \varnothing$
$f(F, PA) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, DEM) = q_1$
$f(q_1, N) = q_2$
$f(q_2, PA) = q_3$

El transductor gráfico que reconoce el sintagma SN_{28} es el siguiente (Fig. 6.31):

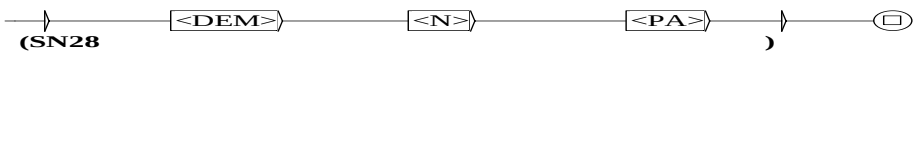


Fig. 6.31: FST gráfico que reconoce la estructura SN_{28}

30. **SN_{29} → DEM PA N (Demostrativo Participio - Adj Nombre)**

ER_0 = DEM PA N

$D_{DEM}(ER_0)=$
$D_{DEM}(DEM)$ PA N + $\alpha(DEM)$ $D_{DEM}(PA\ N)=$
λ PA N + $\alpha(DEM)$ [$D_{DEM}(PA)$ N + $\alpha(PA)$ $D_{DEM}(N)$]=
PA N + $\varnothing$ [$\varnothing$ + $\varnothing$]=
PA N = ER_1

$D_{PA}(ER_0)=$
$D_{PA}(DEM)$ PA N + $\alpha(DEM)$ $D_{PA}(PA\ N)=$
$\varnothing$ PA N + $\alpha(DEM)$ [$D_{PA}(PA)$ N + $\alpha(PA)$ $D_{PA}(N)$]=
$\varnothing$ + $\varnothing$ [λ N + $\varnothing$ $\varnothing$] = $\varnothing$

$D_N(ER_0)=$
$D_N(DEM)$ PA N + $\alpha(DEM)$ + $D_N(PA\ N)=$
$\varnothing$ PA N + $\alpha(DEM)$ [$D_N(PA)$ N + $\alpha(PA)$ $D_N(N)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ N + $\varnothing$ λ] = $\varnothing$

$D_{DEM}(ER_1)=$
$D_{DEM}(PA)$ N + $\alpha(PA)$ $D_{DEM}(N)=$
$\varnothing$ N + $\varnothing$ $\varnothing$ = $\varnothing$

$D_{PA}(ER_1)=$
$D_{PA}(PA)$ N + $\alpha(PA)$ $D_{PA}(N)=$
λ N + $\varnothing$ $\varnothing$ =
N = ER_2

$D_N(ER_1)=$
$D_N(PA)$ N + $\alpha(PA)$ $D_N(N)=$
$\varnothing$ N + $\varnothing$ λ = $\varnothing$

$D_{DEM}(ER_2)=$
$D_{DEM}(N)$ = $\varnothing$

$D_{PA}(ER_2)=$
$D_{PA}(N)$ = $\varnothing$

$D_N(ER_2)=$
$D_N(N)$ = λ

La *Gramática Regular* que reconoce el sintagma SN_{29} es la siguiente:

$G = (\{DEM, PA, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DEM\ ER_1$
$ER_1 ::= PA\ ER_2$
$ER_2 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DEM, PA, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(\mathbf{ER_0}, \mathbf{DEM}) = \mathbf{ER_1}$
$f(\mathbf{ER_0}, \mathbf{A}) = \varnothing$
$f(\mathbf{ER_0}, \mathbf{N}) = \varnothing$
$f(\mathbf{ER_1}, \mathbf{DEM}) = \varnothing$
$f(\mathbf{ER_1}, \mathbf{A}) = \mathbf{ER_2}$
$f(\mathbf{ER_1}, \mathbf{N}) = \varnothing$
$f(\mathbf{ER_2}, \mathbf{DEM}) = \varnothing$
$f(\mathbf{ER_2}, \mathbf{A}) = \varnothing$
$f(\mathbf{ER_2}, \mathbf{N}) = F$
$f(F, \mathbf{DEM}) = \varnothing$
$f(F, \mathbf{A}) = \varnothing$
$f(F, \mathbf{N}) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, DEM) = q_1$
$f(q_1, A) = q_2$
$f(q_2, N) = q_3$

El transductor gráfico que reconoce el sintagma SN_{29} es el siguiente (Fig. 6.32):

SN29 (6.32).grf

Fig. 6.32: FST gráfico que reconoce la estructura SN_{29}

31. **SN_{30} → DET CARD N (Determinante Cardinal Nombre)**

ER_0 = DET CARD N

$D_{DET}(ER_0)=$
$D_{DET}(DET)$ CARD N + $\alpha(DET)$ $D_{DET}(CARD\ N)=$
λ CARD N + $\alpha(DET)$ [$D_{DET}(CARD)$ N + $\alpha(CARD)$ $D_{DET}(N)$]=
CARD N + $\varnothing$ [$\varnothing$ + $\varnothing$]=
CARD N = ER_1

$D_{CARD}(ER_0)=$
$D_{CARD}(DET)$ CARD N + $\alpha(DET)$ $D_{CARD}(CARD\ N)=$
$\varnothing$ CARD N + $\alpha(DET)$ [$D_{CARD}(CARD)$ N + $\alpha(CARD)$ $D_{CARD}(N)$]=
$\varnothing$ + $\varnothing$ [λ N + $\varnothing$ $\varnothing$] = $\varnothing$

$D_N(ER_0)=$
$D_N(DET)$ CARD N + $\alpha(DET)$ + $D_N(CARD\ N)=$
$\varnothing$ CARD N + $\alpha(DET)$ [$D_N(CARD)$ N + $\alpha(CARD)$ $D_N(N)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ N + $\varnothing$ λ] = $\varnothing$

$D_{DET}(ER_1)=$
$D_{DET}(CARD)$ N + $\alpha(CARD)$ $D_{DET}(N)=$
$\varnothing$ N + $\varnothing$ $\varnothing$ = $\varnothing$

$D_{CARD}(ER_1)=$
$D_{CARD}(CARD)$ N + $\alpha(CARD)$ $D_{CARD}(N)=$
λ N + $\varnothing$ $\varnothing$ =
N = ER_2

$D_N(ER_1)=$
$D_N(CARD)$ N + $\alpha(CARD)$ $D_N(N)=$
$\varnothing$ N + $\varnothing$ λ = $\varnothing$

$D_{DET}(ER_2)=$
$D_{DET}(N)$ = $\varnothing$

$D_{CARD}(ER_2)=$
$D_{CARD}(N)=\varnothing$

$D_N(ER_2)=$
$D_N(N)$ = λ

La *Gramática Regular* que reconoce el sintagma SN_{30} es la siguiente:

$G = (\{DET, CARD, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= \text{DET } ER_1$
$ER_1 ::= \text{CARD } ER_2$
$ER_2 ::= \text{N}$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{\text{DET, CARD, N}\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, \text{DET}) = ER_1$
$f(ER_0, \text{CARD}) = \varnothing$
$f(ER_0, \text{N}) = \varnothing$
$f(ER_1, \text{DET}) = \varnothing$
$f(ER_1, \text{CARD}) = ER_2$
$f(ER_1, \text{N}) = \varnothing$
$f(ER_2, \text{DET}) = \varnothing$
$f(ER_2, \text{CARD}) = \varnothing$
$f(ER_2, \text{N}) = F$
$f(F, \text{DET}) = \varnothing$
$f(F, \text{CARD}) = \varnothing$
$f(F, \text{N}) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, DET) = q_1$
$f(q_1, CARD) = q_2$
$f(q_2, N) = q_3$

El transductor gráfico que reconoce el sintagma SN_{30} es el siguiente (Fig. 6.33):

SN30 (6.33).grf

Fig. 6.33: FST gráfico que reconoce la estructura SN_{30}

32. **SN_{31} → DET N CARD (Determinante Nombre Cardinal)**

ER_0 = DET N CARD

$D_{DET}(ER_0)=$
$D_{DET}(DET)$ N CARD + $\alpha(DET)$ $D_{DET}(N\ CARD)=$
λ N CARD + $\alpha(DET)$ [$D_{DET}(N)$ CARD + $\alpha(N)$ $D_{DET}(CARD)$]=
N CARD + $\varnothing$ [$\varnothing$ + $\varnothing$]=
N CARD = ER_1

$D_N(ER_0)=$
$D_N(DET)$ N CARD + $\alpha(DET)$ + $D_N(N\ CARD)=$
$\varnothing$ N CARD + $\alpha(DET)$ [$D_N(N)$ CARD + $\alpha(N)$ $D_N(CARD)$]=
$\varnothing$ + $\varnothing$ [λ CARD + $\varnothing$ $\varnothing$] = $\varnothing$

$D_{CARD}(ER_0)=$
$D_{CARD}(DET)$ N CARD + $\alpha(DET)$ $D_{CARD}(N\ CARD)=$
$\varnothing$ N CARD + $\alpha(DET)$ [$D_{CARD}(N)$ CARD + $\alpha(N)$ $D_{CARD}(CARD)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ CARD + $\varnothing$ λ] = $\varnothing$

$D_{DET}(ER_1)=$
$D_{DET}(N)$ CARD + $\alpha(N)$ $D_{DET}(CARD)=$
$\varnothing$ CARD + $\varnothing$ $\varnothing$ = $\varnothing$

$D_N(ER_1)=$
$D_N(N)$ CARD + $\alpha(N)$ $D_N(CARD)=$
λ CARD + $\varnothing$ $\varnothing$ =
CARD = ER_2

$D_{CARD}(ER_1)=$
$D_{CARD}(N)$ CARD + $\alpha(N)$ $D_{CARD}(CARD)=$
$\varnothing$ CARD + $\varnothing$ λ = $\varnothing$

$D_{DET}(ER_2) =$
$D_{DET}(CARD) = \varnothing$

$D_{N}(ER_2) =$
$D_{N}(CARD) = \varnothing$

$D_{CARD}(ER_2) =$
$D_{CARD}(CARD) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{31} es la siguiente:

$G = (\{DET, N, CARD\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= N\ ER_2$
$ER_2 ::= CARD$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DET, N, CARD\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (\mathbf{ER_0, DET}) = \mathbf{ER_1}$
$f\ (\mathbf{ER_0, N}) = \varnothing$
$f\ (\mathbf{ER_0, CARD}) = \varnothing$
$f\ (\mathbf{ER_1, DET}) = \varnothing$
$f\ (\mathbf{ER_1, N}) = \mathbf{ER_2}$
$f\ (\mathbf{ER_1, CARD}) = \varnothing$
$f\ (\mathbf{ER_2, DET}) = \varnothing$
$f\ (\mathbf{ER_2, N}) = \varnothing$
$f\ (\mathbf{ER_2, CARD}) = F$
$f\ (F, \mathbf{DET}) = \varnothing$
$f\ (F, \mathbf{N}) = \varnothing$
$f\ (F, \mathbf{CARD}) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, DET) = q_1$
$f\ (q_1, N) = q_2$
$f\ (q_2, CARD) = q_3$

El transductor gráfico que reconoce el sintagma SN_{31} es el siguiente (Fig. 6.34):

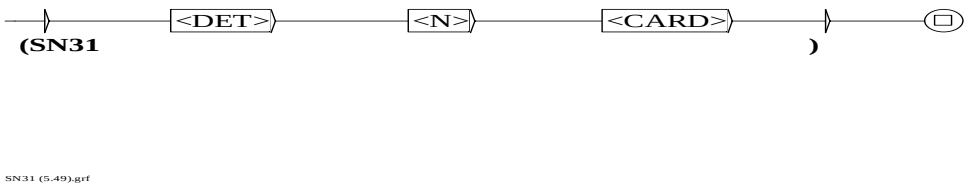


Fig. 6.34: FST gráfico que reconoce la estructura SN_{31}

33. **SN_{32} → POS CARD N (Posesivo Cardinal Nombre)**

ER_0 = POS CARD N

$D_{POS}(ER_0)=$
$D_{POS}(POS)$ CARD N + $\alpha(POS)$ $D_{POS}(CARD\ N)=$
λ CARD N + $\alpha(POS)$ [$D_{POS}(CARD)$ N + $\alpha(CARD)$ $D_{POS}(N)$]=
CARD N + $\varnothing$ [$\varnothing$ + $\varnothing$]=
CARD N = ER_1

$D_{CARD}(ER_0)=$
$D_{CARD}(POS)$ CARD N + $\alpha(POS)$ $D_{CARD}(CARD\ N)=$
$\varnothing$ CARD N + $\alpha(POS)$ [$D_{CARD}(CARD)$ N + $\alpha(CARD)$ $D_{CARD}(N)$]=
$\varnothing$ + $\varnothing$ [λ N + $\varnothing$ $\varnothing$] = $\varnothing$

$D_N(ER_0)=$
$D_N(POS)$ CARD N + $\alpha(POS)$ + $D_N(CARD\ N)=$
$\varnothing$ CARD N + $\alpha(POS)$ [$D_N(CARD)$ N + $\alpha(CARD)$ $D_N(N)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ N + $\varnothing$ λ] = $\varnothing$

$D_{POS}(ER_1) =$
$D_{POS}(CARD)\ N + \alpha(CARD)\ D_{POS}(N) =$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_{CARD}(ER_1) =$
$D_{CARD}(CARD)\ N + \alpha(CARD)\ D_{CARD}(N) =$
$\lambda\ N + \varnothing\ \varnothing =$
$N = ER_2$

$D_N(ER_1) =$
$D_N(CARD)\ N + \alpha(CARD)\ D_N(N) =$
$\varnothing\ N + \varnothing\ \lambda = \varnothing$

$D_{POS}(ER_2) =$
$D_{POS}(N) = \varnothing$

$D_{CARD}(ER_2) =$
$D_{CARD}(N) = \varnothing$

$D_N(ER_2) =$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{32} es la siguiente:

$G = (\{POS, CARD, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= POS\ ER_1$
$ER_1 ::= CARD\ ER_2$
$ER_2 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{POS, CARD, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, POS) = ER_1$
$f\ (ER_0, CARD) = \varnothing$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_1, POS) = \varnothing$
$f\ (ER_1, CARD) = ER_2$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_2, POS) = \varnothing$
$f\ (ER_2, CARD) = \varnothing$
$f\ (ER_2, N) = F$
$f\ (F, POS) = \varnothing$
$f\ (F, CARD) = \varnothing$
$f\ (F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, POS) = q_1$
$f\ (q_1, CARD) = q_2$
$f\ (q_2, N) = q_3$

El transductor gráfico que reconoce el sintagma SN_{32} es el siguiente (Fig. 6.35):

SN32 (6.35).grf

Fig. 6.35: FST gráfico que reconoce la estructura SN_{32}

34. **SN_{33} → POS N CARD (Posesivo Nombre Cardinal)**

ER_0 = POS N CARD

$D_{POS}(ER_0)=$
$D_{POS}(POS)\ N\ CARD + \alpha(POS)\ D_{POS}(N\ CARD)=$
$\lambda\ N\ CARD + \alpha(POS)\ [\ D_{POS}(N)\ CARD + \alpha(N)\ D_{POS}(CARD)\ ]=$
$N\ CARD + \varnothing\ [\varnothing + \varnothing]=$
$N\ CARD = ER_1$

$D_N(ER_0)=$
$D_N(POS)\ N\ CARD + \alpha(POS) + D_N(N\ CARD)=$
$\varnothing\ N\ CARD + \alpha(POS)\ [\ D_N(N)\ CARD + \alpha(N)\ D_N(CARD)\ ]=$
$\varnothing + \varnothing\ [\lambda\ CARD + \varnothing\ \varnothing] = \varnothing$

$D_{CARD}(ER_0)=$
$D_{CARD}(POS)\ N\ CARD + \alpha(POS)\ D_{CARD}(N\ CARD)=$
$\varnothing\ N\ CARD + \alpha(POS)\ [\ D_{CARD}(N)\ CARD + \alpha(N)\ D_{CARD}(CARD)\ ]=$
$\varnothing + \varnothing\ [\varnothing\ CARD + \varnothing\ \lambda] = \varnothing$

$D_{POS}(ER_1)=$
$D_{POS}(N)\ CARD + \alpha(N)\ D_{POS}(CARD)=$
$\varnothing\ CARD + \varnothing\ \varnothing = \varnothing$

$D_N(ER_1)=$
$D_N(N)\ CARD + \alpha(N)\ D_N(CARD)=$
$\lambda\ CARD + \varnothing\ \varnothing =$
$CARD = ER_2$

$D_{CARD}(ER_1)=$
$D_{CARD}(N)\ CARD + \alpha(N)\ D_{CARD}(CARD)=$
$\varnothing\ CARD + \varnothing\ \lambda = \varnothing$

$D_{POS}(ER_2)=$
$D_{POS}(CARD) = \varnothing$

$D_N(ER_2)=$
$D_N(CARD) = \varnothing$

$D_{CARD}(ER_2)=$
$D_{CARD}(CARD) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{33} es la siguiente:

$G = (\{POS, N, CARD\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= POS\ ER_1$
$ER_1 ::= N\ ER_2$
$ER_2 ::= CARD$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{POS, N, CARD\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

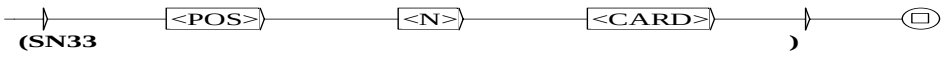
donde la función de transición, f, se define como:

$f(ER_0, POS) = ER_1$
$f(ER_0, N) = \varnothing$
$f(ER_0, CARD) = \varnothing$
$f(ER_1, POS) = \varnothing$
$f(ER_1, N) = ER_2$
$f(ER_1, CARD) = \varnothing$
$f(ER_2, POS) = \varnothing$
$f(ER_2, N) = \varnothing$
$f(ER_2, CARD) = F$
$f(F, POS) = \varnothing$
$f(F, N) = \varnothing$
$f(F, CARD) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, POS) = q_1$
$f(q_1, N) = q_2$
$f(q_2, CARD) = q_3$

El transductor gráfico que reconoce el sintagma SN_{33} es el siguiente (Fig. 6.36):



SN33 (6.36).grf

Fig. 6.36: FST gráfico que reconoce la estructura SN_{33}

35. **SN_{34} → DEM CARD N (Demostrativo Cardinal Nombre)**

ER_0 = DEM CARD N

$D_{DEM}(ER_0)=$
$D_{DEM}(DEM)\ CARD\ N + \alpha(DEM)\ D_{DEM}(CARD\ N)=$
$\lambda\ CARD\ N + \alpha(DEM)\ [\ D_{DEM}(CARD)\ N + \alpha(CARD)\ D_{DEM}(N)\ ]=$
$CARD\ N + \varnothing\ [\varnothing + \varnothing]=$
$CARD\ N = ER_1$

$D_{CARD}(ER_0)=$
$D_{CARD}(DEM)\ CARD\ N + \alpha(DEM)\ D_{CARD}(CARD\ N)=$
$\varnothing\ CARD\ N + \alpha(DEM)\ [\ D_{CARD}(CARD)\ N + \alpha(CARD)\ D_{CARD}(N)\ ]=$
$\varnothing + \varnothing\ [\lambda\ N + \varnothing\ \varnothing] = \varnothing$

$D_N(ER_0)=$
$D_N(DEM)\ CARD\ N + \alpha(DEM) + D_N(CARD\ N)=$
$\varnothing\ CARD\ N + \alpha(DEM)\ [\ D_N(CARD)\ N + \alpha(CARD)\ D_N(N)\ ]=$
$\varnothing + \varnothing\ [\varnothing\ N + \varnothing\ \lambda] = \varnothing$

$D_{DEM}(ER_1)=$
$D_{DEM}(CARD)\ N + \alpha(CARD)\ D_{DEM}(N)=$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_{CARD}(ER_1)=$
$D_{CARD}(CARD)\ N + \alpha(CARD)\ D_{CARD}(N)=$
$\lambda\ N + \varnothing\ \varnothing =$
$N = ER_2$

$D_N(ER_1)=$
$D_N(CARD)\ N + \alpha(CARD)\ D_N(N)=$
$\varnothing\ N + \varnothing\ \lambda = \varnothing$

$D_{DEM}(ER_2)=$
$D_{DEM}(N) = \varnothing$

$D_{CARD}(ER_2)=$
$D_{CARD}(N) = \varnothing$

$D_N(ER_2)=$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{34} es la siguiente:

$G = (\{DEM, CARD, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DEM\ ER_1$
$ER_1 ::= CARD\ ER_2$
$ER_2 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DEM, CARD, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f , se define como:

$f\ (ER_0, DEM) = ER_1$
$f\ (ER_0, CARD) = \varnothing$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_1, DEM) = \varnothing$
$f\ (ER_1, CARD) = ER_2$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_2, DEM) = \varnothing$
$f\ (ER_2, CARD) = \varnothing$
$f\ (ER_2, N) = F$
$f\ (F, DEM) = \varnothing$
$f\ (F, CARD) = \varnothing$
$f\ (F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, DEM) = q_1$
$f\ (q_1, CARD) = q_2$
$f\ (q_2, N) = q_3$

El transductor gráfico que reconoce el sintagma SN_{34} es el siguiente (Fig. 6.37):

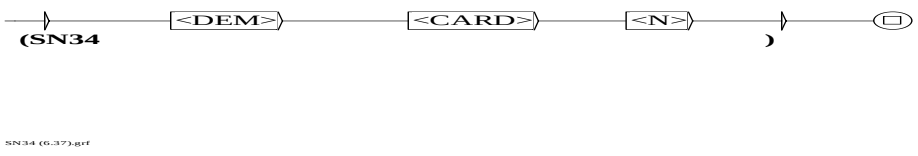


Fig. 6.37: FST gráfico que reconoce la estructura SN_{34}

36. **SN_{35} → DEM N CARD (Determinante Nombre Cardinal)**

ER_0 = DEM N CARD

$D_{DEM}(ER_0)=$
$D_{DEM}(DEM)$ N CARD + $\alpha(DEM)$ D_{DEM}(N CARD)=
λ N CARD + $\alpha(DEM)$ [$D_{DEM}(N)$ CARD + $\alpha(N)$ D_{DEM}(CARD)]=
N CARD + $\varnothing$ [$\varnothing$ + $\varnothing$]=
N CARD = ER_1

$D_N(ER_0)=$
$D_N(DEM)$ N CARD + $\alpha(DEM)$ + D_N(N CARD)=
$\varnothing$ N CARD + $\alpha(DEM)$ [$D_N(N)$ CARD + $\alpha(N)$ D_N(CARD)]=
$\varnothing$ + $\varnothing$ [λ CARD + $\varnothing$ $\varnothing$] = $\varnothing$

$D_{CARD}(ER_0)=$
$D_{CARD}(DEM)$ N CARD + $\alpha(DEM)$ D_{CARD}(N CARD)=
$\varnothing$ N CARD + $\alpha(DEM)$ [$D_{CARD}(N)$ CARD + $\alpha(N)$ D_{CARD}(CARD)]=
$\varnothing$ + $\varnothing$ [$\varnothing$ CARD + $\varnothing$ λ] = $\varnothing$

$D_{DEM}(ER_1)=$
$D_{DEM}(N)$ CARD + $\alpha(N)$ D_{DEM}(CARD)=
$\varnothing$ CARD + $\varnothing$ $\varnothing$ = $\varnothing$

$D_N(ER_1)=$
$D_N(N)$ CARD + $\alpha(N)$ D_N(CARD)=
λ CARD + $\varnothing$ $\varnothing$ =
CARD = ER_2

$D_{CARD}(ER_1)=$
$D_{CARD}(N)$ CARD + $\alpha(N)$ D_{CARD}(CARD)=
$\varnothing$ CARD + $\varnothing$ λ = $\varnothing$

$D_{DEM}(ER_2)=$
$D_{DEM}(CARD)$ = $\varnothing$

$D_N(ER_2)=$
$D_N(CARD)$ = $\varnothing$

$D_{CARD}(ER_2)=$
$D_{CARD}(CARD)$ = λ

La *Gramática Regular* que reconoce el sintagma SN_{35} es la siguiente:

$G = (\{DEM, N, CARD\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

ER_0 ::= DEM ER_1
ER_1 ::= N ER_2
ER_2 ::= CARD

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$$AF = (\{DEM, N, CARD\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$$

donde la función de transición, f, se define como:

$$f(ER_0, DEM) = ER_1$$
$$f(ER_0, N) = \varnothing$$
$$f(ER_0, CARD) = \varnothing$$
$$f(ER_1, DEM) = \varnothing$$
$$f(ER_1, N) = ER_2$$
$$f(ER_1, CARD) = \varnothing$$
$$f(ER_2, DEM) = \varnothing$$
$$f(ER_2, N) = \varnothing$$
$$f(ER_2, CARD) = F$$
$$f(F, DEM) = \varnothing$$
$$f(F, N) = \varnothing$$
$$f(F, CARD) = \varnothing$$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$$f(q_0, DEM) = q_1$$
$$f(q_1, N) = q_2$$
$$f(q_2, CARD) = q_3$$

El transductor gráfico que reconoce el sintagma SN_{35} es el siguiente (Fig. 6.38):

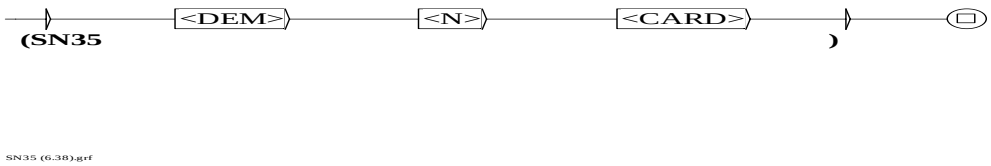


Fig. 6.38: FST gráfico que reconoce la estructura SN_{35}

37. **SN_{36} → DET ORD N (Determinante Ordinal Nombre)**

ER_0 = DET ORD N

$D_{DET}(ER_0)=$
$D_{DET}(DET)$ ORD N + $\alpha(DET)$ $D_{DET}(ORD\ N)=$
λ ORD N + $\alpha(DET)$ [$D_{DET}(ORD)$ N + $\alpha(ORD)$ $D_{DET}(N)$]=
ORD N + $\varnothing$ [$\varnothing$ + $\varnothing$]=
ORD N = ER_1

$D_{ORD}(ER_0)=$
$D_{ORD}(DET)$ ORD N + $\alpha(DET)$ $D_{ORD}(ORD\ N)=$
$\varnothing$ ORD N + $\alpha(DET)$ [$D_{ORD}(ORD)$ N + $\alpha(ORD)$ $D_{ORD}(N)$]=
$\varnothing$ + $\varnothing$ [λ N + $\varnothing$ $\varnothing$] = $\varnothing$

$D_N(ER_0)=$
$D_N(DET)$ ORD N + $\alpha(DET)$ + $D_N(ORD\ N)=$
$\varnothing$ ORD N + $\alpha(DET)$ [$D_N(ORD)$ N + $\alpha(ORD)$ $D_N(N)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ N + $\varnothing$ λ] = $\varnothing$

$D_{DET}(ER_1)=$
$D_{DET}(ORD)$ N + $\alpha(ORD)$ $D_{DET}(N)=$
$\varnothing$ N + $\varnothing$ $\varnothing$ = $\varnothing$

$D_{ORD}(ER_1)=$
$D_{ORD}(ORD)$ N + $\alpha(ORD)$ $D_{ORD}(N)=$
λ N + $\varnothing$ $\varnothing$ =
N = ER_2

$D_N(ER_1)=$
$D_N(ORD)$ N + $\alpha(ORD)$ $D_N(N)=$
$\varnothing$ N + $\varnothing$ λ = $\varnothing$

$D_{DET}(ER_2)=$
$D_{DET}(N)$ = $\varnothing$

$D_{ORD}(ER_2)=$
$D_{ORD}(N)$ = $\varnothing$

$D_N(ER_2)=$
$D_N(N)$ = λ

La *Gramática Regular* que reconoce el sintagma SN_{36} es la siguiente:

$G = (\{DET, ORD, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

ER_0 ::= DET ER_1
ER_1 ::= ORD ER_2
ER_2 ::= N

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DET, ORD, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f , se define como:

$f\ (ER_0, DET) = ER_1$
$f\ (ER_0, ORD) = \varnothing$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_1, DET) = \varnothing$
$f\ (ER_1, ORD) = ER_2$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_2, DET) = \varnothing$
$f\ (ER_2, ORD) = \varnothing$
$f\ (ER_2, N) = F$
$f\ (F, DET) = \varnothing$
$f\ (F, ORD) = \varnothing$
$f\ (F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, DET) = q_1$
$f\ (q_1, ORD) = q_2$
$f\ (q_2, N) = q_3$

El transductor gráfico que reconoce el sintagma SN_{36} es el siguiente (Fig. 6.39):

SN36 (6.39).grf

Fig. 6.39: FST gráfico que reconoce la estructura SN_{36}

38. **SN_{37} → DET N ORD (Determinante Nombre Ordinal)**

ER_0 = DET N ORD

$D_{DET}(ER_0)=$
$D_{DET}(DET)$ N ORD + $\alpha(DET)$ $D_{DET}(N\ ORD)=$
λ N ORD + $\alpha(DET)$ [$D_{DET}(N)$ ORD + $\alpha(N)$ $D_{DET}(ORD)$]=
N ORD + $\varnothing$ [$\varnothing$ + $\varnothing$]=
N ORD = ER_1

$D_N(ER_0)=$
$D_N(DET)$ N ORD + $\alpha(DET)$ + $D_N(N\ ORD)=$
$\varnothing$ N ORD + $\alpha(DET)$ [$D_N(N)$ ORD + $\alpha(N)$ $D_N(ORD)$]=
$\varnothing$ + $\varnothing$ [λ ORD + $\varnothing$ $\varnothing$] = $\varnothing$

$D_{ORD}(ER_0)=$
$D_{ORD}(DET)$ N ORD + $\alpha(DET)$ $D_{ORD}(N\ ORD)=$
$\varnothing$ N ORD + $\alpha(DET)$ [$D_{ORD}(N)$ ORD + $\alpha(N)$ $D_{ORD}(ORD)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ ORD + $\varnothing$ λ] = $\varnothing$

$D_{DET}(ER_1)=$
$D_{DET}(N)$ ORD + $\alpha(N)$ $D_{DET}(ORD)=$
$\varnothing$ ORD + $\varnothing$ $\varnothing$ = $\varnothing$

$D_N(ER_1)=$
$D_N(N)$ ORD + $\alpha(N)$ $D_N(ORD)=$
λ ORD + $\varnothing$ $\varnothing$ =
ORD = ER_2

$D_{ORD}(ER_1)=$
$D_{ORD}(N)$ ORD + $\alpha(N)$ $D_{ORD}(ORD)=$
$\varnothing$ CARD + $\varnothing$ λ = $\varnothing$

$D_{DET}(ER_2)=$
$D_{DET}(ORD)$ = $\varnothing$

$D_N(ER_2)=$
$D_N(ORD)$ = $\varnothing$

$D_{ORD}(ER_2)=$
$D_{ORD}(ORD)=\lambda$

La *Gramática Regular* que reconoce el sintagma SN_{37} es la siguiente:

$G = (\{DET, N, ORD\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= N\ ER_2$
$ER_2 ::= ORD$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DET, N, ORD\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, DET) = ER_1$
$f(ER_0, N) = \varnothing$
$f(ER_0, ORD) = \varnothing$
$f(ER_1, DET) = \varnothing$
$f(ER_1, N) = ER_2$
$f(ER_1, ORD) = \varnothing$
$f(ER_2, DET) = \varnothing$
$f(ER_2, N) = \varnothing$
$f(ER_2, ORD) = F$
$f(F, DET) = \varnothing$
$f(F, N) = \varnothing$
$f(F, ORD) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, DET) = q_1$
$f(q_1, N) = q_2$
$f(q_2, ORD) = q_3$

El transductor gráfico que reconoce el sintagma SN_{37} es el siguiente (Fig. 6.40):

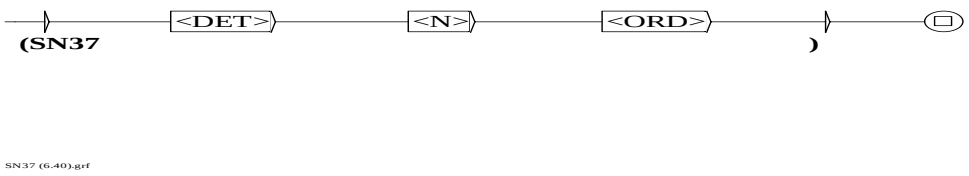


Fig. 6.40: FST gráfico que reconoce la estructura SN_{37}

39. **SN_{38} → POS ORD N (Posesivo Ordinal Nombre)**

ER_0 = POS ORD N

$D_{POS}(ER_0)=$
$D_{POS}(POS)$ ORD N + $\alpha(POS)$ $D_{POS}(ORD\ N)=$
λ ORD N + $\alpha(POS)$ [$D_{POS}(ORD)$ N + $\alpha(ORD)$ $D_{POS}(N)$]=
ORD N + $\varnothing$ [$\varnothing$ + $\varnothing$]=
ORD N = ER_1

$D_{ORD}(ER_0)=$
$D_{ORD}(POS)$ ORD N + $\alpha(POS)$ $D_{ORD}(ORD\ N)=$
$\varnothing$ ORD N + $\alpha(POS)$ [$D_{ORD}(ORD)$ N + $\alpha(ORD)$ $D_{ORD}(N)$]=
$\varnothing$ + $\varnothing$ [λ N + $\varnothing$ $\varnothing$] = $\varnothing$

$D_N(ER_0)=$
$D_N(POS)$ ORD N + $\alpha(POS)$ + $D_N(ORD\ N)=$
$\varnothing$ ORD N + $\alpha(POS)$ [$D_N(ORD)$ N + $\alpha(ORD)$ $D_N(N)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ N + $\varnothing$ λ] = $\varnothing$

$D_{POS}(ER_1)=$
$D_{POS}(CRD)$ N + $\alpha(ORD)$ $D_{POS}(N)=$
$\varnothing$ N + $\varnothing$ $\varnothing$ = $\varnothing$

$D_{ORD}(ER_1)=$
$D_{ORD}(ORD)$ N + $\alpha(ORD)$ $D_{ORD}(N)=$
λ N + $\varnothing$ $\varnothing$ =
N = ER_2

$D_N(ER_1)=$
$D_N(ORD)$ N + $\alpha(ORD)$ $D_N(N)=$
$\varnothing$ N + $\varnothing$ λ = $\varnothing$

$D_{POS}(ER_2)=$
$D_{POS}(N)$ = $\varnothing$

$D_{ORD}(ER_2)=$
$D_{ORD}(N)$ = $\varnothing$

$D_N(ER_2)=$
$D_N(N)$ = λ

La *Gramática Regular* que reconoce el sintagma SN_{38} es la siguiente:

$G = (\{POS, ORD, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= \text{POS}\ ER_1$
$ER_1 ::= \text{ORD}\ ER_2$
$ER_2 ::= \text{N}$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$$AF = (\{\text{POS}, \text{ORD}, \text{N}\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$$

donde la función de transición, f, se define como:

$f(ER_0, \text{POS}) = ER_1$
$f(ER_0, \text{ORD}) = \varnothing$
$f(ER_0, \text{N}) = \varnothing$
$f(ER_1, \text{POS}) = \varnothing$
$f(ER_1, \text{ORD}) = ER_2$
$f(ER_1, \text{N}) = \varnothing$
$f(ER_2, \text{POS}) = \varnothing$
$f(ER_2, \text{ORD}) = \varnothing$
$f(ER_2, \text{N}) = F$
$f(F, \text{POS}) = \varnothing$
$f(F, \text{ORD}) = \varnothing$
$f(F, \text{N}) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, \text{POS}) = q_1$
$f(q_1, \text{ORD}) = q_2$
$f(q_2, \text{N}) = q_3$

El transductor gráfico que reconoce el sintagma SN_{38} es el siguiente (Fig. 6.41):

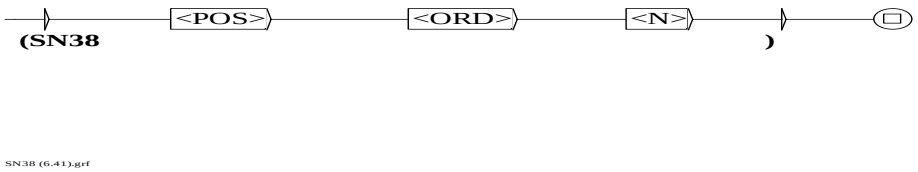


Fig. 6.41: FST gráfico que reconoce la estructura SN_{38}

40. **SN_{39} → POS N ORD (Posesivo Nombre Ordinal)**

ER_0 = POS N ORD

$D_{POS}(ER_0)=$
$D_{POS}(POS)$ N ORD + $\alpha(POS)$ $D_{POS}(N\ ORD)=$
λ N ORD + $\alpha(POS)$ [$D_{POS}(N)$ ORD + $\alpha(N)$ $D_{POS}(ORD)$]=
N ORD + $\varnothing$ [$\varnothing$ + $\varnothing$]=
N ORD = ER_1

$D_N(ER_0)=$
$D_N(POS)$ N ORD + $\alpha(POS)$ + $D_N(N\ ORD)=$
$\varnothing$ N ORD + $\alpha(POS)$ [$D_N(N)$ ORD + $\alpha(N)$ $D_N(ORD)$]=
$\varnothing$ + $\varnothing$ [λ ORD + $\varnothing$ $\varnothing$] = $\varnothing$

$D_{ORD}(ER_0)=$
$D_{ORD}(POS)$ N ORD + $\alpha(POS)$ $D_{ORD}(N\ ORD)=$
$\varnothing$ N ORD + $\alpha(POS)$ [$D_{ORD}(N)$ ORD + $\alpha(N)$ $D_{ORD}(ORD)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ ORD + $\varnothing$ λ] = $\varnothing$

$D_{POS}(ER_1)=$
$D_{POS}(N)$ ORD + $\alpha(N)$ $D_{POS}(ORD)=$
$\varnothing$ ORD + $\varnothing$ $\varnothing$ = $\varnothing$

$D_N(ER_1)=$
$D_N(N)$ ORD + $\alpha(N)$ $D_N(ORD)=$
λ ORD + $\varnothing$ $\varnothing$ =
ORD = ER_2

$D_{ORD}(ER_1)=$
$D_{ORD}(N)$ ORD + $\alpha(N)$ $D_{ORD}(ORD)=$
$\varnothing$ CARD + $\varnothing$ λ = $\varnothing$

$D_{POS}(ER_2)=$
$D_{POS}(ORD) = \varnothing$

$D_{N}(ER_2)=$
$D_{N}(ORD) = \varnothing$

$D_{ORD}(ER_2)=$
$D_{ORD}(ORD) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{39} es la siguiente:

$G = (\{DET, N, ORD\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= N\ ER_2$
$ER_2 ::= ORD$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DET, N, ORD\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, DET) = ER_1$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_0, ORD) = \varnothing$
$f\ (ER_1, DET) = \varnothing$
$f\ (ER_1, N) = ER_2$
$f\ (ER_1, ORD) = \varnothing$
$f\ (ER_2, DET) = \varnothing$
$f\ (ER_2, N) = \varnothing$
$f\ (ER_2, ORD) = F$
$f\ (F, DET) = \varnothing$
$f\ (F, N) = \varnothing$
$f\ (F, ORD) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, POS) = q_1$
$f\ (q_1, N) = q_2$
$f\ (q_2, ORD) = q_3$

El transductor gráfico que reconoce el sintagma SN_{39} es el siguiente (Fig. 6.42):

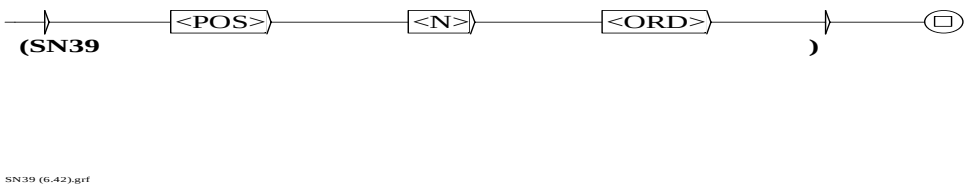


Fig. 6.42: FST gráfico que reconoce la estructura SN_{39}

41. **SN_{40} → DEM ORD N (Demostrativo Ordinal Nombre)**

ER_0 = DEM ORD N

$D_{DEM}(ER_0)=$
$D_{DEM}(DEM)$ ORD N + $\alpha(DEM)$ $D_{DEM}(ORD\ N)=$
λ ORD N + $\alpha(DEM)$ [$D_{DEM}(ORD)$ N + $\alpha(ORD)$ $D_{DEM}(N)$]=
ORD N + $\varnothing$ [$\varnothing$ + $\varnothing$]=
ORD N = ER_1

$D_{ORD}(ER_0)=$
$D_{ORD}(DEM)$ ORD N + $\alpha(DEM)$ $D_{ORD}(ORD\ N)=$
$\varnothing$ ORD N + $\alpha(DEM)$ [$D_{ORD}(ORD)$ N + $\alpha(ORD)$ $D_{ORD}(N)$]=
$\varnothing$ + $\varnothing$ [λ N + $\varnothing$ $\varnothing$] = $\varnothing$

$D_N(ER_0)=$
$D_N(DEM)$ ORD N + $\alpha(DEM)$ + $D_N(ORD\ N)=$
$\varnothing$ ORD N + $\alpha(DEM)$ [$D_N(ORD)$ N + $\alpha(ORD)$ $D_N(N)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ N + $\varnothing$ λ] = $\varnothing$

$D_{DEM}(ER_1) =$
$D_{DEM}(CRD)\ N + \alpha(ORD)\ D_{DEM}(N) =$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_{ORD}(ER_1) =$
$D_{ORD}(ORD)\ N + \alpha(ORD)\ D_{ORD}(N) =$
$\lambda\ N + \varnothing\ \varnothing =$
$N = ER_2$

$D_N(ER_1) =$
$D_N(ORD)\ N + \alpha(ORD)\ D_N(N) =$
$\varnothing\ N + \varnothing\ \lambda = \varnothing$

$D_{DET}(ER_2) =$
$D_{DET}(N) = \varnothing$

$D_{ORD}(ER_2) =$
$D_{ORD}(N) = \varnothing$

$D_N(ER_2) =$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{40} es la siguiente:

$G = (\{DEM, ORD, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DEM\ ER_1$
$ER_1 ::= ORD\ ER_2$
$ER_2 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DEM, ORD, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, DEM) = ER_1$
$f\ (ER_0, ORD) = \varnothing$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_1, DEM) = \varnothing$
$f\ (ER_1, ORD) = ER_2$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_2, DEM) = \varnothing$
$f\ (ER_2, ORD) = \varnothing$
$f\ (ER_2, N) = F$
$f\ (F, DEM) = \varnothing$
$f\ (F, ORD) = \varnothing$
$f\ (F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, DEM) = q_1$
$f\ (q_1, ORD) = q_2$
$f\ (q_2, N) = q_3$

El transductor gráfico que reconoce el sintagma SN_{40} es el siguiente (Fig. 6.43):

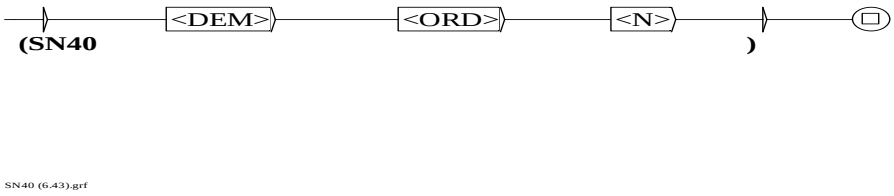


Fig. 6.43: FST gráfico que reconoce la estructura SN_{40}

42. **SN_{41} → DEM N ORD (Demostrativo Nombre Ordinal)**

ER_0 = DEM N ORD

$D_{DEM}(ER_0)=$
$D_{DEM}(DEM)\ N\ ORD + \alpha(DEM)\ D_{DEM}(N\ ORD)=$
$\lambda\ N\ ORD + \alpha(DEM)\ [\ D_{DEM}(N)\ ORD + \alpha(N)\ D_{DEM}(ORD)\ ]=$
$N\ ORD + \varnothing\ [\varnothing + \varnothing]=$
$N\ ORD = ER_1$

$D_N(ER_0)=$
$D_N(DEM)\ N\ ORD + \alpha(DEM) + D_N(N\ ORD)=$
$\varnothing\ N\ ORD + \alpha(DEM)\ [\ D_N(N)\ ORD + \alpha(N)\ D_N(ORD)\ ]=$
$\varnothing + \varnothing\ [\lambda\ ORD + \varnothing\ \varnothing] = \varnothing$

$D_{ORD}(ER_0)=$
$D_{ORD}(DEM)\ N\ ORD + \alpha(DEM)\ D_{ORD}(N\ ORD)=$
$\varnothing\ N\ ORD + \alpha(DEM)\ [\ D_{ORD}(N)\ ORD + \alpha(N)\ D_{ORD}(ORD)\ ]=$
$\varnothing + \varnothing\ [\varnothing\ ORD + \varnothing\ \lambda] = \varnothing$

$D_{DEM}(ER_1)=$
$D_{DEM}(N)\ ORD + \alpha(N)\ D_{DEM}(ORD)=$
$\varnothing\ ORD + \varnothing\ \varnothing = \varnothing$

$D_N(ER_1)=$
$D_N(N)\ ORD + \alpha(N)\ D_N(ORD)=$
$\lambda\ ORD + \varnothing\ \varnothing =$
$ORD = ER_2$

$D_{ORD}(ER_1)=$
$D_{ORD}(N)\ ORD + \alpha(N)\ D_{ORD}(ORD)=$
$\varnothing\ CARD + \varnothing\ \lambda = \varnothing$

$D_{DEM}(ER_2)=$
$D_{DEM}(ORD) = \varnothing$

$D_N(ER_2)=$
$D_N(ORD) = \varnothing$

$D_{ORD}(ER_2)=$
$D_{ORD}(ORD) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{41} es la siguiente:

$G = (\{DEM, N, ORD\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DEM\ ER_1$
$ER_1 ::= N\ ER_2$
$ER_2 ::= ORD$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$$AF = (\{DEM, N, ORD\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$$

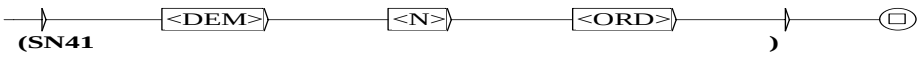
donde la función de transición, f, se define como:

$f(ER_0, DEM) = ER_1$
$f(ER_0, N) = \varnothing$
$f(ER_0, ORD) = \varnothing$
$f(ER_1, DEM) = \varnothing$
$f(ER_1, N) = ER_2$
$f(ER_1, ORD) = \varnothing$
$f(ER_2, DEM) = \varnothing$
$f(ER_2, N) = \varnothing$
$f(ER_2, ORD) = F$
$f(F, DEM) = \varnothing$
$f(F, N) = \varnothing$
$f(F, ORD) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, DET) = q_1$
$f(q_1, N) = q_2$
$f(q_2, ORD) = q_3$

El transductor gráfico que reconoce el sintagma SN_{41} es el siguiente (Fig. 6.44):



SN41 (6.44).grf

Fig. 6.44: FST gráfico que reconoce la estructura SN_{41}

43. **SN_{42} → DET A N A (Determinante Adjetivo Nombre Adjetivo)**

ER_0 = DET A N A

$D_{DET}(ER_0)=$
$D_{DET}(DET)$ A N A + $\alpha(DET)$ D_{DET}(A N A)=
λ A N A + $\alpha(DET)$ D_{DET}(A N A)=
A N A + $\emptyset$ [D_{DET}(A) N A + α(A) D_{DET}(N A)]=
A N A = ER_1

$D_A(ER_0)=$
$D_A(DET)$ A N A + $\alpha(DET)$ + D_A(A N A)=
$\emptyset$ A N A + $\alpha(DET)$ D_A(A N A)=
$\emptyset$ + $\emptyset$ [D_A(A) N A + α(A) D_A(N A)]=$\emptyset$

$D_N(ER_0)=$
$D_N(DET)$ A N A + $\alpha(DET)$ + D_N(A N A)=
$\emptyset$ A N A + $\alpha(DET)$ D_N(A N A)=
$\emptyset$ + $\emptyset$ [D_N(A) N A + α(A) D_N(N A)]=$\emptyset$

$D_{DET}(ER_1)=$
D_{DET}(A) N A + α(A) D_{DET}(N A)=
$\emptyset$ N A + α(A) D_{DET}(N A)=
$\emptyset$ + $\emptyset$ [D_{DET}(N) A + α(N) D_{DET}(A)]=$\emptyset$

$D_A(ER_1)=$
D_A(A) N A + α(A) D_A(N A)=
λ N A + α(A) D_A(N A)=
N A + $\emptyset$ [D_A(N) A + α(N) D_A(A)]=
N A = ER_2

$D_N(ER_1)=$
D_N(A) N A + α(A) D_N(N A)=
$\emptyset$ N A + α(A) D_N(N A)=
$\emptyset$ + $\emptyset$ [D_N(N) A + α(N) D_N(A)]=$\emptyset$

$D_{DET}(ER_2)=$
D_{DET}(N) A + α(N) D_{DET}(A)=
$\emptyset$ A + $\emptyset$ $\emptyset$ = $\emptyset$

$D_A(ER_2)=$
D_A(N) A + α(N) A =
$\emptyset$ A + $\emptyset$ A = $\emptyset$

$D_N(ER_2)=$
D_N(N) A + α(N) D_N(A)=
λ A + $\emptyset$ $\emptyset$ =
A = ER_3

$D_{DET}(ER_3)=$
$D_{DET}(A) = \varnothing$

$D_A(ER_3)=$
$D_A(A)=\lambda$

$D_N(ER_3)=$
$D_N(A)=\varnothing$

La *Gramática Regular* que reconoce el sintagma SN_{42} es la siguiente:

$G = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= A\ ER_2$
$ER_2 ::= N\ ER_3$
$ER_3 ::= A$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3, F\}, f, ER_0, F)$

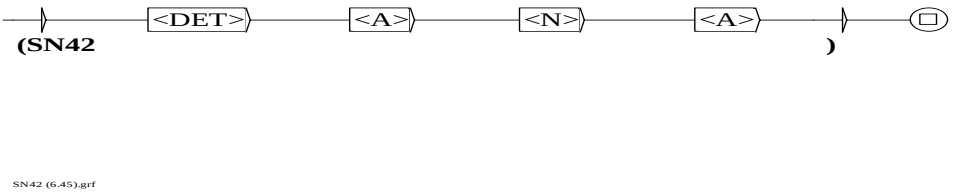
donde la función de transición, f, se define como:

$f\ (ER_0, DET)= ER_1$
$f\ (ER_0, A)=\varnothing$
$f\ (ER_0, N)=\varnothing$
$f\ (ER_1, DET)=\varnothing$
$f\ (ER_1, A) = ER_2$
$f\ (ER_1, N)=\varnothing$
$f\ (ER_2, DET)=\varnothing$
$f\ (ER_2, A)=\varnothing$
$f\ (ER_2, N)= ER_3$
$f\ (ER_3, DET)=\varnothing$
$f\ (ER_3, A)= F$
$f\ (ER_3, N)=\varnothing$
$f\ (F, DET)=\varnothing$
$f\ (F, A) = \varnothing$
$f\ (F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, \text{DET}) = q_1$
$f\ (q_1, \text{A}) = q_2$
$f\ (q_2, \text{N}) = q_3$
$f\ (q_3, \text{A}) = q_4$

El transductor gráfico que reconoce el sintagma SN_{42} es el siguiente (Fig. 6.45):



SN42 (6.45).grf

Fig. 6.45: FST gráfico que reconoce la estructura SN_{42}

44. **SN_{43} → DET PA N PA (Determinante Partic-Adj Nombre Partic-Adj)**

ER_0 = DET PA N PA

$D_{\text{DET}}(\text{ER}_0) =$
$D_{\text{DET}}(\text{DET})$ PA N PA + $\alpha(\text{DET})\ D_{\text{DET}}(\text{PA N PA}) =$
λ PA N PA + $\alpha(\text{DET})\ D_{\text{DET}}(\text{PA N PA}) =$
PA N PA + $\varnothing$ [$D_{\text{DET}}(\text{PA})$ N PA + $\alpha(\text{PA})\ D_{\text{DET}}(\text{N PA})$] =
PA N PA = ER_1

$D_{\text{PA}}(\text{ER}_0) =$
$D_{\text{PA}}(\text{DET})$ PA N PA + $\alpha(\text{DET})$ + $D_{\text{PA}}(\text{PA N PA}) =$
$\varnothing$ PA N PA + $\alpha(\text{DET})\ D_{\text{PA}}(\text{PA N PA}) =$
$\varnothing$ + $\varnothing$ [$D_{\text{PA}}(\text{PA})$ N PA + $\alpha(\text{PA})\ D_{\text{PA}}(\text{N PA})$] = $\varnothing$

$D_{\text{N}}(\text{ER}_0) =$
$D_{\text{N}}(\text{DET})$ PA N PA + $\alpha(\text{DET})$ + $D_{\text{N}}(\text{PA N PA}) =$
$\varnothing$ PA N PA + $\alpha(\text{DET})\ D_{\text{N}}(\text{PA N PA}) =$
$\varnothing$ + $\varnothing$ [$D_{\text{N}}(\text{PA})$ N PA + $\alpha(\text{PA})\ D_{\text{N}}(\text{N PA})$] = $\varnothing$

$D_{DET}(ER_1)=$
$D_{DET}(PA)\ N\ PA + \alpha(PA)\ D_{DET}(N\ PA)=$
$\varnothing\ N\ PA + \alpha(PA)\ D_{DET}(N\ PA)=$
$\varnothing + \varnothing\ [D_{DET}(N)\ PA + \alpha(N)\ D_{DET}(PA)]=\varnothing$

$D_{PA}(ER_1)=$
$D_{PA}(PA)\ N\ PA + \alpha(PA)\ D_{PA}(N\ PA)=$
$\lambda\ N\ PA + \alpha(PA)\ D_{PA}(N\ PA)=$
$N\ PA + \varnothing\ [D_{PA}(N)\ PA + \alpha(N)\ D_{PA}(PA)]=$
$N\ PA = ER_2$

$D_N(ER_1)=$
$D_N(PA)\ N\ PA + \alpha(PA)\ D_N(N\ PA)=$
$\varnothing\ N\ PA + \alpha(PA)\ D_N(N\ PA)=$
$\varnothing + \varnothing\ [D_N(N)\ PA + \alpha(N)\ D_N(PA)]=\varnothing$

$D_{DET}(ER_2)=$
$D_{DET}(N)\ PA + \alpha(N)\ D_{DET}(PA)=$
$\varnothing\ PA + \varnothing\ \varnothing=\varnothing$

$D_{PA}(ER_2)=$
$D_{PA}(N)\ PA + \alpha(N)\ PA =$
$\varnothing\ PA + \varnothing\ PA = \varnothing$

$D_N(ER_2)=$
$D_N(N)\ PA + \alpha(N)\ D_N(PA)=$
$\lambda\ PA + \varnothing\ \varnothing=$
$PA = ER_3$

$D_{DET}(ER_3)=$
$D_{DET}(PA) =\varnothing$

$D_A(ER_3)=$
$D_A(PA)=\lambda$

$D_N(ER_3)=$
$D_N(PA)=\varnothing$

La *Gramática Regular* que reconoce el sintagma SN_{43} es la siguiente:

$G = (\{DET, PA, N\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= PA\ ER_2$
$ER_2 ::= N\ ER_3$
$ER_3 ::= PA$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$$AF = (\{DET, PA, N\}, \{ER_0, ER_1, ER_2, ER_3, F\}, f, ER_0, F)$$

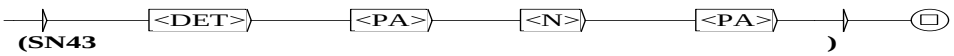
donde la función de transición, f, se define como:

$f(ER_0, DET) = ER_1$
$f(ER_0, PA) = \varnothing$
$f(ER_0, N) = \varnothing$
$f(ER_1, DET) = \varnothing$
$f(ER_1, PA) = ER_2$
$f(ER_1, N) = \varnothing$
$f(ER_2, DET) = \varnothing$
$f(ER_2, PA) = \varnothing$
$f(ER_2, N) = ER_3$
$f(ER_3, DET) = \varnothing$
$f(ER_3, PA) = F$
$f(ER_3, N) = \varnothing$
$f(F, DET) = \varnothing$
$f(F, PA) = \varnothing$
$f(F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, DET) = q_1$
$f(q_1, PA) = q_2$
$f(q_2, N) = q_3$
$f(q_3, PA) = q_4$

El transductor gráfico que reconoce el sintagma SN_{43} es el siguiente (Fig. 6.46):



SN43 (6.46).grf

Fig. 6.46: FST gráfico que reconoce la estructura SN_{43}

45. **SN_{44} → DET A N PA (Determinante Adjetivo Nombre Partic - Adj)**

ER_0 = DET A N PA

$D_{DET}(ER_0)=$
$D_{DET}(DET)$ A N PA + $\alpha(DET)$ $D_{DET}(A\ N\ PA)=$
λ A N PA + $\alpha(DET)$ $D_{DET}(A\ N\ PA)=$
A N PA + $\varnothing$ [$D_{DET}(A)$ N PA + $\alpha(A)$ $D_{DET}(N\ PA)$]=
A N PA = ER_1

$D_A(ER_0)=$
$D_A(DET)$ A N PA + $\alpha(DET)$ + $D_A(A\ N\ PA)=$
$\varnothing$ A N PA + $\alpha(DET)$ $D_A(A\ N\ PA)=$
$\varnothing$ + $\varnothing$ [$D_A(A)$ N PA + $\alpha(A)$ $D_A(N\ PA)$]=$\varnothing$

$D_N(ER_0)=$
$D_N(DET)$ A N PA + $\alpha(DET)$ + $D_N(A\ N\ PA)=$
$\varnothing$ A N PA + $\alpha(DET)$ $D_N(A\ N\ PA)=$
$\varnothing$ + $\varnothing$ [$D_N(A)$ N PA + $\alpha(A)$ $D_N(N\ PA)$]=$\varnothing$

$D_{PA}(ER_0)=$
$D_{PA}(DET)$ A N PA + $\alpha(DET)$ + $D_{PA}(A\ N\ PA)=$
$\varnothing$ A N PA + $\alpha(DET)$ $D_{PA}(A\ N\ PA)=$
$\varnothing$ + $\varnothing$ [$D_{PA}(A)$ N PA + $\alpha(A)$ $D_{PA}(N\ PA)$]=$\varnothing$

$D_{DET}(ER_1)=$
$D_{DET}(A)$ N PA + $\alpha(A)$ $D_{DET}(N\ PA)=$
$\varnothing$ N PA + $\alpha(A)$ $D_{DET}(N\ PA)=$
$\varnothing$ + $\varnothing$ [$D_{DET}(N)$ PA + $\alpha(N)$ $D_{DET}(PA)$]=$\varnothing$

$D_A(ER_1)=$
$D_A(A)$ N PA + $\alpha(A)$ $D_A(N\ PA)=$
λ N PA + $\alpha(A)$ $D_A(N\ PA)=$
N PA + $\varnothing$ [$D_A(N)$ PA + $\alpha(N)$ $D_A(PA)$]=
N PA = ER_2

$D_N(ER_1)=$
$D_N(A)$ N PA + $\alpha(A)$ $D_N(N\ PA)=$
$\varnothing$ N PA + $\alpha(A)$ $D_N(N\ PA)=$
$\varnothing$ + $\varnothing$ [$D_N(N)$ PA + $\alpha(N)$ $D_N(PA)$]=$\varnothing$

$D_{PA}(ER_1)=$
$D_{PA}(A)$ N PA + $\alpha(A)$ $D_{PA}(N\ PA)=$
$\varnothing$ N PA + $\alpha(A)$ $D_{PA}(N\ PA)=$
$\varnothing$ + $\varnothing$ [$D_{PA}(N)$ PA + $\alpha(N)$ $D_{PA}(PA)$]=$\varnothing$

$D_{DET}(ER_2)=$
$D_{DET}(N)\ PA + \alpha(N)\ D_{DET}(PA)=$
$\varnothing\ PA + \varnothing\ \varnothing = \varnothing$

$D_A(ER_2)=$
$D_A(N)\ PA + \alpha(N)\ PA =$
$\varnothing\ PA + \varnothing\ PA = \varnothing$

$D_N(ER_2)=$
$D_N(N)\ PA + \alpha(N)\ D_N(PA)=$
$\lambda\ PA + \varnothing\ \varnothing =$
$PA = ER_3$

$D_{PA}(ER_2)=$
$D_{PA}(N)\ PA + \alpha(N)\ PA =$
$\varnothing\ PA + \varnothing\ PA = \varnothing$

$D_{DET}(ER_3)=$
$D_{DET}(PA) = \varnothing$

$D_A(ER_3)=$
$D_A(PA)=\varnothing$

$D_N(ER_3)=$
$D_N(PA)=\varnothing$

$D_{PA}(ER_3)=$
$D_{PA}(PA)=\lambda$

La *Gramática Regular* que reconoce el sintagma SN_{44} es la siguiente:

$G = (\{DET, A, N, PA\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= A\ ER_2$
$ER_2 ::= N\ ER_3$
$ER_3 ::= PA$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DET, A, N, PA\}, \{ER_0, ER_1, ER_2, ER_3, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, \text{DET}) = ER_1$
$f\ (ER_0, \text{A}) = \varnothing$
$f\ (ER_0, \text{N}) = \varnothing$
$f\ (ER_0, \text{PA}) = \varnothing$
$f\ (ER_1, \text{DET}) = \varnothing$
$f\ (ER_1, \text{A}) = ER_2$
$f\ (ER_1, \text{N}) = \varnothing$
$f\ (ER_1, \text{PA}) = \varnothing$
$f\ (ER_2, \text{DET}) = \varnothing$
$f\ (ER_2, \text{A}) = \varnothing$
$f\ (ER_2, \text{N}) = ER_3$
$f\ (ER_2, \text{PA}) = \varnothing$
$f\ (ER_3, \text{DET}) = \varnothing$
$f\ (ER_3, \text{A}) = \varnothing$
$f\ (ER_3, \text{N}) = \varnothing$
$f\ (ER_3, \text{PA}) = F$
$f\ (F, \text{DET}) = \varnothing$
$f\ (F, \text{A}) = \varnothing$
$f\ (F, \text{N}) = \varnothing$
$f\ (F, \text{PA}) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, \text{DET}) = q_1$
$f\ (q_1, \text{A}) = q_2$
$f\ (q_2, \text{N}) = q_3$
$f\ (q_3, \text{PA}) = q_4$

El transductor gráfico que reconoce el sintagma SN_{44} es el siguiente (Fig. 6.47):

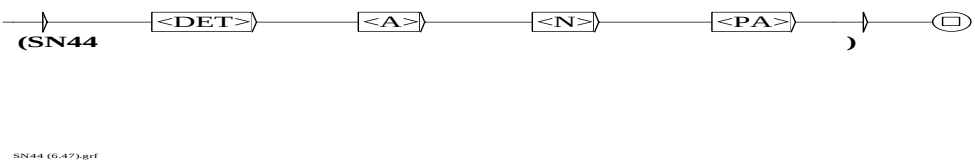


Fig. 6.47: FST gráfico que reconoce la estructura SN_{44}

46. **SN_{45} → DET PA N A (Determinante Partic-Adj Nombre Adjetivo)**

ER_0 = DET PA N A

$D_{DET}(ER_0)=$
$D_{DET}(DET)$ PA N A + $\alpha(DET)$ $D_{DET}(PA\ N\ A)=$
λ PA N A + $\alpha(DET)$ $D_{DET}(PA\ N\ A)=$
PA N A + $\varnothing$ [$D_{DET}(PA)$ N A + $\alpha(PA)$ $D_{DET}(N\ A)$]=
PA N A = ER_1

$D_{PA}(ER_0)=$
$D_{PA}(DET)$ PA N A + $\alpha(DET)$ + $D_{PA}(PA\ N\ A)=$
$\varnothing$ PA N A + $\alpha(DET)$ $D_{PA}(PA\ N\ A)=$
$\varnothing$ + $\varnothing$ [$D_{PA}(PA)$ N A + $\alpha(PA)$ $D_{PA}(N\ A)$]=$\varnothing$

$D_N(ER_0)=$
$D_N(DET)$ PA N A + $\alpha(DET)$ + $D_N(PA\ N\ A)=$
$\varnothing$ PA N A + $\alpha(DET)$ $D_N(PA\ N\ A)=$
$\varnothing$ + $\varnothing$ [$D_N(PA)$ N A + $\alpha(PA)$ $D_N(N\ A)$]=$\varnothing$

$D_A(ER_0)=$
$D_A(DET)$ PA N A + $\alpha(DET)$ + $D_A(PA\ N\ A)=$
$\varnothing$ PA N A + $\alpha(DET)$ $D_A(PA\ N\ A)=$
$\varnothing$ + $\varnothing$ [$D_A(PA)$ N A + $\alpha(PA)$ $D_A(N\ A)$]=$\varnothing$

$D_{DET}(ER_1)=$
$D_{DET}(PA)$ N A + $\alpha(PA)$ $D_{DET}(N\ A)=$
$\varnothing$ N A + $\alpha(PA)$ $D_{DET}(N\ A)=$
$\varnothing$ + $\varnothing$ [$D_{DET}(N)$ A + $\alpha(N)$ $D_{DET}(A)$]=$\varnothing$

$D_{PA}(ER_1)=$
$D_{PA}(PA)$ N A + $\alpha(PA)$ $D_{PA}(N\ A)=$
λ N A + $\alpha(PA)$ $D_{PA}(N\ A)=$
N A + $\varnothing$ [$D_{PA}(N)$ A + $\alpha(N)$ $D_{PA}(A)$]=
N A = ER_2

$D_N(ER_1)=$
$D_N(PA)$ N A + $\alpha(PA)$ $D_N(N\ A)=$
$\varnothing$ N A + $\alpha(PA)$ $D_N(N\ A)=$
$\varnothing$ + $\varnothing$ [$D_N(N)$ A + $\alpha(N)$ $D_N(A)$]=$\varnothing$

$D_A(ER_1)=$
$D_A(PA)$ N A + $\alpha(PA)$ $D_A(N\ A)=$
$\varnothing$ N A + $\alpha(PA)$ $D_A(N\ A)=$
$\varnothing$ + $\varnothing$ [$D_A(N)$ A + $\alpha(N)$ $D_A(A)$]=$\varnothing$

$D_{DET}(ER_2)=$
$D_{DET}(N)\ A + \alpha(N)\ D_{DET}(A)=$
$\varnothing\ A + \varnothing\ \varnothing = \varnothing$

$D_{PA}(ER_2)=$
$D_{PA}(N)\ A + \alpha(N)\ A =$
$\varnothing\ A + \varnothing\ A = \varnothing$

$D_{N}(ER_2)=$
$D_{N}(N)\ A + \alpha(N)\ D_{N}(A)=$
$\lambda\ A + \varnothing\ \varnothing =$
$A = ER_3$

$D_{A}(ER_2)=$
$D_{A}(N)\ A + \alpha(N)\ A =$
$\varnothing\ A + \varnothing\ A = \varnothing$

$D_{DET}(ER_3)=$
$D_{DET}(A) = \varnothing$

$D_{PA}(ER_3)=$
$D_{PA}(A)=\varnothing$

$D_{N}(ER_3)=$
$D_{N}(A)=\varnothing$

$D_{A}(ER_3)=$
$D_{A}(A)=\lambda$

La *Gramática Regular* que reconoce el sintagma SN_{45} es la siguiente:

$G = (\{DET, A, N, PA\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= PA\ ER_2$
$ER_2 ::= N\ ER_3$
$ER_3 ::= A$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DET, A, N, PA\}, \{ER_0, ER_1, ER_2, ER_3, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, DET) = ER_1$
$f\ (ER_0, PA) = \varnothing$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_0, A) = \varnothing$
$f\ (ER_1, DET) = \varnothing$
$f\ (ER_1, PA) = ER_2$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_1, A) = \varnothing$
$f\ (ER_2, DET) = \varnothing$
$f\ (ER_2, PA) = \varnothing$
$f\ (ER_2, N) = ER_3$
$f\ (ER_2, A) = \varnothing$
$f\ (ER_3, DET) = \varnothing$
$f\ (ER_3, PA) = \varnothing$
$f\ (ER_3, N) = \varnothing$
$f\ (ER_3, A) = F$
$f\ (F, DET) = \varnothing$
$f\ (F, PA) = \varnothing$
$f\ (F, N) = \varnothing$
$f\ (F, A) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, DET) = q_1$
$f\ (q_1, PA) = q_2$
$f\ (q_2, N) = q_3$
$f\ (q_3, A) = q_4$

El transductor gráfico que reconoce el sintagma SN_{45} es el siguiente (Fig. 6.48):

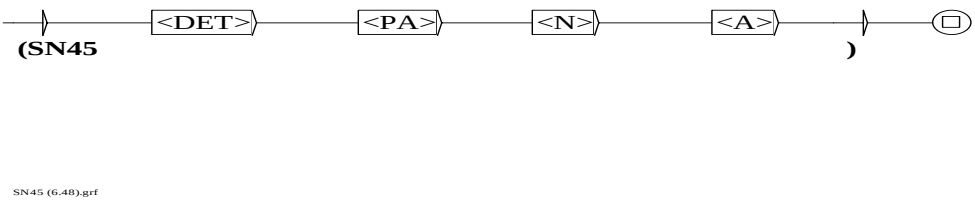


Fig. 6.48: FST gráfico que reconoce la estructura SN_{45}

47. **SN_{46} → CUANT DET N (Cuantificador Determinante Nombre)**

ER_0 = CUANT DET N

$D_{CUANT}(ER_0)=$
$D_{CUANT}(CUANT)$ DET N + $\alpha(CUANT)$ $D_{CUANT}(DET\ N)=$
λ DET N + $\alpha(CUANT)$ [$D_{CUANT}(DET)$ N + $\alpha(DET)$ $D_{CUANT}(N)$]=
DET N + $\varnothing$ [$D_{CUANT}(DET)$ N + $\alpha(DET)$ $D_{CUANT}(N)$]=
DET N = ER_1

$D_{DET}(ER_0)=$
$D_{DET}(CUANT)$ DET N + $\alpha(CUANT)$ $D_{DET}(DET\ N)=$
$\varnothing$ DET N + $\alpha(CUANT)$ [$D_{DET}(DET)$ N + $\alpha(DET)$ $D_{DET}(N)$]=
$\varnothing$ + $\varnothing$ [$D_{DET}(DET)$ N + $\alpha(DET)$ $D_{DET}(N)$]=$\varnothing$

$D_N(ER_0)=$
$D_N(CUANT)$ DET N + $\alpha(CUANT)$ + $D_N(DET\ N)=$
$\varnothing$ DET N + $\alpha(CUANT)$ [$D_N(DET)$ N + $\alpha(DET)$ $D_N(N)$]=
$\varnothing$ + $\varnothing$ [$D_N(DET)$ N + $\alpha(DET)$ $D_N(N)$]=$\varnothing$

$D_{CUANT}(ER_1)=$
$D_{CUANT}(DET)$ N + $\alpha(DET)$ $D_{CUANT}(N)=$
$\varnothing$ N + $\varnothing$ $\varnothing=\varnothing$

$D_{DET}(ER_1)=$
$D_{DET}(DET)$ N + $\alpha(DET)$ $D_{DET}(N)=$
λ N + $\varnothing$ $\varnothing=$
N = ER_2

$D_N(ER_1)=$
$D_N(DET)$ N + $\alpha(DET)$ $D_N(N)=$
$\varnothing$ N + $\varnothing$ $\lambda=\varnothing$

$D_{CUANT}(ER_2)=$
$D_{CUANT}(N)=\varnothing$

$D_{DET}(ER_2)=$
$D_{DET}(N)=\varnothing$

$D_N(ER_2)=$
$D_N(N)=\lambda$

La *Gramática Regular* que reconoce el sintagma SN_{46} es la siguiente:

$G=(\{CUANT, DET, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= \text{CUANT } ER_1$
$ER_1 ::= \text{DET } ER_2$
$ER_2 ::= \text{N}$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$$AF = (\{\text{CUANT, DET, N}\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$$

donde la función de transición, f, se define como:

$f(ER_0, \text{CUANT}) = ER_1$
$f(ER_0, \text{DET}) = \varnothing$
$f(ER_0, \text{N}) = \varnothing$
$f(ER_1, \text{CUANT}) = \varnothing$
$f(ER_1, \text{DET}) = ER_2$
$f(ER_1, \text{N}) = \varnothing$
$f(ER_2, \text{CUANT}) = \varnothing$
$f(ER_2, \text{DET}) = \varnothing$
$f(ER_2, \text{N}) = F$
$f(F, \text{CUANT}) = \varnothing$
$f(F, \text{DET}) = \varnothing$
$f(F, \text{N}) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, \text{CUANT}) = q_1$
$f(q_1, \text{DET}) = q_2$
$f(q_2, \text{N}) = q_3$

El transductor gráfico que reconoce el sintagma SN_{46} es el siguiente (Fig. 6.49):

SN46 (6.49).grf

Fig. 6.49: FST gráfico que reconoce la estructura SN_{46}

48. **SN_{47} → CUANT DEM N (Cuantificador Demostrativo Nombre)**

ER_0 = CUANT DEM N

$D_{CUANT}(ER_0)=$
$D_{CUANT}(CUANT)$ DEM N + $\alpha(CUANT)$ $D_{CUANT}(DEM\ N)=$
λ DEM N + $\alpha(CUANT)$ [$D_{CUANT}(DEM)$ N + $\alpha(DEM)$ $D_{CUANT}(N)$]=
DEM N + $\varnothing$ [$D_{CUANT}(DEM)$ N + $\alpha(DEM)$ $D_{CUANT}(N)$]=
DET N = ER_1

$D_{DEM}(ER_0)=$
$D_{DEM}(CUANT)$ DEM N + $\alpha(CUANT)$ $D_{DEM}(DEM\ N)=$
$\varnothing$ DEM N + $\alpha(CUANT)$ [$D_{DEM}(DEM)$ N + $\alpha(DEM)$ $D_{DEM}(N)$]=
$\varnothing$ + $\varnothing$ [$D_{DEM}(DEM)$ N + $\alpha(DEM)$ $D_{DEM}(N)$]=$\varnothing$

$D_N(ER_0)=$
$D_N(CUANT)$ DEM N + $\alpha(CUANT)$ + $D_N(DEM\ N)=$
$\varnothing$ DEM N + $\alpha(CUANT)$ [$D_N(DEM)$ N + $\alpha(DEM)$ $D_N(N)$]=
$\varnothing$ + $\varnothing$ [$D_N(DEM)$ N + $\alpha(DEM)$ $D_N(N)$]=$\varnothing$

$D_{CUANT}(ER_1)=$
$D_{CUANT}(DEM)$ N + $\alpha(DEM)$ $D_{CUANT}(N)=$
$\varnothing$ N + $\varnothing$ $\varnothing=\varnothing$

$D_{DEM}(ER_1)=$
$D_{DEM}(DEM)$ N + $\alpha(DEM)$ $D_{DEM}(N)=$
λ N + $\varnothing$ $\varnothing=$
$N=ER_2$

$D_N(ER_1)=$
$D_N(DEM)$ N + $\alpha(DEM)$ $D_N(N)=$
$\varnothing$ N + $\varnothing$ $\lambda=\varnothing$

$D_{CUANT}(ER_2)=$
$D_{CUANT}(N)=\varnothing$

$D_{DEM}(ER_2)=$
$D_{DEM}(N)=\varnothing$

$D_N(ER_2)=$
$D_N(N)=\lambda$

La *Gramática Regular* que reconoce el sintagma SN_{47} es la siguiente:

$$G = (\{\text{CUANT}, \text{DEM}, \text{N}\}, \{\text{ER}_0, \text{ER}_1, \text{ER}_2\}, \text{ER}_0, P)$$

donde las *reglas de producción*, P, se definen como:

$$\text{ER}_0 ::= \text{CUANT ER}_1$$
$$\text{ER}_1 ::= \text{DEM ER}_2$$
$$\text{ER}_2 ::= \text{N}$$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$$AF = (\{\text{CUANT}, \text{DEM}, \text{N}\}, \{\text{ER}_0, \text{ER}_1, \text{ER}_2, F\}, f, \text{ER}_0, F)$$

donde la función de transición, f, se define como:

$$f(\text{ER}_0, \text{CUANT}) = \text{ER}_1$$
$$f(\text{ER}_0, \text{DEM}) = \varnothing$$
$$f(\text{ER}_0, \text{N}) = \varnothing$$
$$f(\text{ER}_1, \text{CUANT}) = \varnothing$$
$$f(\text{ER}_1, \text{DEM}) = \text{ER}_2$$
$$f(\text{ER}_1, \text{N}) = \varnothing$$
$$f(\text{ER}_2, \text{CUANT}) = \varnothing$$
$$f(\text{ER}_2, \text{DEM}) = \varnothing$$
$$f(\text{ER}_2, \text{N}) = F$$
$$f(F, \text{CUANT}) = \varnothing$$
$$f(F, \text{DEM}) = \varnothing$$
$$f(F, \text{N}) = \varnothing$$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$$f(q_0, \text{CUANT}) = q_1$$
$$f(q_1, \text{DEM}) = q_2$$
$$f(q_2, \text{N}) = q_3$$

El transductor gráfico que reconoce el sintagma SN_{47} es el siguiente (Fig. 6.50):

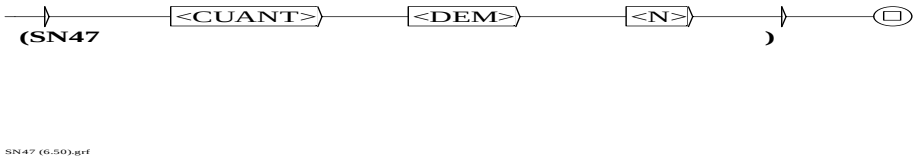


Fig. 6.50: FST gráfico que reconoce la estructura SN_{47}

49. **SN_{48} → CUANT POS ORD N (Cuantificador Posesivo Ordinal Nombre)**

ER_0 = CUANT POS ORD N

$D_{CUANT}(ER_0)=$
$D_{CUANT}(CUANT)$ POS ORD N + $\alpha(CUANT)$ D_{CUANT}(POS ORD N)=
λ POS ORD N + $\alpha(CUANT)$ [$D_{CUANT}(POS)$ ORD N + $\alpha(POS)$ D_{CUANT}(ORD N)]=
POS ORD N + $\varnothing$ [$D_{CUANT}(POS)$ ORD N + $\alpha(POS)$ D_{CUANT}(ORD N)]=
POS ORD N= ER_1

$D_{POS}(ER_0)=$
$D_{POS}(CUANT)$ POS ORD N + $\alpha(CUANT)$ D_{POS}(POS ORD N)=
$\varnothing$ POS ORD N + $\alpha(CUANT)$ [$D_{POS}(POS)$ ORD N + $\alpha(POS)$ D_{POS}(ORD N)]=
$\varnothing$ + $\varnothing$ [$D_{POS}(POS)$ ORD N + $\alpha(POS)$ D_{POS}(ORD N)]=$\varnothing$

$D_{ORD}(ER_0)=$
$D_{ORD}(CUANT)$ POS ORD N + $\alpha(CUANT)$ D_{ORD}(POS ORD N)=
$\varnothing$ POS ORD N + $\alpha(CUANT)$ [$D_{ORD}(POS)$ ORD N + $\alpha(POS)$ D_{ORD}(ORD N)]=
$\varnothing$ + $\varnothing$ [$D_{ORD}(POS)$ ORD N + $\alpha(POS)$ D_{ORD}(ORD N)]=$\varnothing$

$D_N(ER_0)=$
$D_N(CUANT)$ POS ORD N + $\alpha(CUANT)$ + D_N(POS ORD N)=
$\varnothing$ POS ORD N + $\alpha(CUANT)$ [$D_N(POS)$ ORD N + $\alpha(POS)$ D_N(ORD N)]=
$\varnothing$ + $\varnothing$ [$D_N(POS)$ ORD N + $\alpha(POS)$ D_N(ORD N)]=$\varnothing$

$D_{CUANT}(ER_1) =$
$D_{CUANT}(POS)\ ORD\ N + \alpha(POS)\ D_{CUANT}(ORD\ N) =$
$\varnothing\ ORD\ N + \varnothing\ [D_{CUANT}(ORD)\ N + \alpha(ORD)\ D_{CUANT}(N)] = \varnothing$

$D_{POS}(ER_1) =$
$D_{POS}(POS)\ ORD\ N + \alpha(POS)\ D_{POS}(ORD\ N) =$
$\lambda\ ORD\ N + \varnothing\ [D_{POS}(ORD)\ N + \alpha(ORD)\ D_{POS}(N)] =$
$ORD\ N = ER_2$

$D_{ORD}(ER_1) =$
$D_{ORD}(POS)\ ORD\ N + \alpha(POS)\ D_{ORD}(ORD\ N) =$
$\varnothing\ ORD\ N + \varnothing\ [D_{ORD}(ORD)\ N + \alpha(ORD)\ D_{ORD}(N)] = \varnothing$

$D_N(ER_1) =$
$D_N(POS)\ ORD\ N + \alpha(POS)\ D_N(ORD\ N) =$
$\varnothing\ ORD\ N + \varnothing\ [D_N(ORD)\ N + \alpha(ORD)\ D_N(N)] = \varnothing$

$D_{CUANT}(ER_2) =$
$D_{CUANT}(ORD)\ N + \alpha(ORD)\ D_{CUANT}(N) =$
$\varnothing\ \ N + \varnothing = \varnothing$

$D_{POS}(ER_2) =$
$D_{POS}(ORD)\ N + \alpha(ORD)\ D_{POS}(N) =$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_{ORD}(ER_2) =$
$D_{ORD}(ORD)\ N + \alpha(ORD)\ D_{ORD}(N) =$
$\lambda\ N + \varnothing\ \varnothing =$
$N = ER_3$

$D_N(ER_2) =$
$D_N(ORD)\ N + \alpha(ORD)\ D_N(N) =$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_{CUANT}(ER_3) =$
$D_{CUANT}(N) = \varnothing$

$D_{POS}(ER_3) =$
$D_{POS}(N) = \varnothing$

$D_{ORD}(ER_3) =$
$D_{ORD}(N) = \varnothing$

$D_N(ER_3) =$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{48} es la siguiente:

$G = (\{CUANT, POS, ORD, N\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= \text{CUANT } ER_1$
$ER_1 ::= \text{POS } ER_2$
$ER_2 ::= \text{ORD } ER_3$
$ER_3 ::= \text{N}$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$$AF = (\{\text{CUANT, POS, ORD, N}\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$$

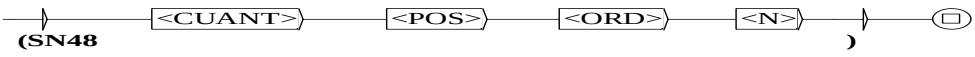
donde la función de transición, f, se define como:

$f(ER_0, \text{CUANT}) = ER_1$
$f(ER_0, \text{POS}) = \varnothing$
$f(ER_0, \text{ORD}) = \varnothing$
$f(ER_0, \text{N}) = \varnothing$
$f(ER_1, \text{CUANT}) = \varnothing$
$f(ER_1, \text{POS}) = ER_2$
$f(ER_1, \text{ORD}) = \varnothing$
$f(ER_1, \text{N}) = \varnothing$
$f(ER_2, \text{CUANT}) = \varnothing$
$f(ER_2, \text{POS}) = \varnothing$
$f(ER_2, \text{ORD}) = ER_3$
$f(ER_2, \text{N}) = \varnothing$
$f(ER_3, \text{CUANT}) = \varnothing$
$f(ER_3, \text{POS}) = \varnothing$
$f(ER_3, \text{ORD}) = \varnothing$
$f(ER_3, \text{N}) = F$
$f(F, \text{CUANT}) = \varnothing$
$f(F, \text{POS}) = \varnothing$
$f(F, \text{ORD}) = \varnothing$
$f(F, \text{N}) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, \text{CUANT}) = q_1$
$f(q_1, \text{POS}) = q_2$
$f(q_2, \text{ORD}) = q_3$
$f(q_3, \text{N}) = q_4$

El transductor gráfico que reconoce el sintagma SN_{48} es el siguiente (Fig. 6.51):



SN48 (6.51).grf

Fig. 6.51: FST gráfico que reconoce la estructura SN_{48}

50. **SN_{49} → CUANT POS N ORD (Cuantificador Posesivo Nombre Ordinal)**

ER_0 = CUANT POS N ORD

$D_{CUANT}(ER_0)=$
$D_{CUANT}(CUANT)$ POS N ORD + $\alpha(CUANT)$ $D_{CUANT}(POS\ N\ ORD)=$
λ POS N ORD + $\alpha(CUANT)$ [$D_{CUANT}(POS)$ N ORD + $\alpha(POS)$ $D_{CUANT}(N\ ORD)$]=
POS N ORD + $\varnothing$ [$D_{CUANT}(POS)$ N ORD + $\alpha(POS)$ $D_{CUANT}(N\ ORD)$]=
POS N ORD= ER_1

$D_{POS}(ER_0)=$
$D_{POS}(CUANT)$ POS N ORD + $\alpha(CUANT)$ $D_{POS}(POS\ N\ ORD)=$
$\varnothing$ POS N ORD + $\alpha(CUANT)$ [$D_{POS}(POS)$ N ORD + $\alpha(POS)$ $D_{POS}(N\ ORD)$]=
$\varnothing$ + $\varnothing$ [$D_{POS}(POS)$ N ORD + $\alpha(POS)$ $D_{POS}(N\ ORD)$]=$\varnothing$

$D_N(ER_0)=$
$D_N(CUANT)$ POS N ORD + $\alpha(CUANT)$ + $D_N(POS\ N\ ORD)=$
$\varnothing$ POS N ORD + $\alpha(CUANT)$ [$D_N(POS)$ N ORD + $\alpha(POS)$ $D_N(N\ ORD)$]=
$\varnothing$ + $\varnothing$ [$D_N(POS)$ N ORD + $\alpha(POS)$ $D_N(N\ ORD)$]=$\varnothing$

$D_{ORD}(ER_0)=$
$D_{ORD}(CUANT)$ POS N ORD + $\alpha(CUANT)$ $D_{ORD}(POS\ N\ ORD)=$
$\varnothing$ POS N ORD + $\alpha(CUANT)$ [$D_{ORD}(POS)$ N ORD + $\alpha(POS)$ $D_{ORD}(N\ ORD)$]=
$\varnothing$ + $\varnothing$ [$D_{ORD}(POS)$ N ORD + $\alpha(POS)$ $D_{ORD}(N\ ORD)$]=$\varnothing$

$D_{CUANT}(ER_1) =$
$D_{CUANT}(POS)\ N\ ORD + \alpha(POS)\ D_{CUANT}(N\ ORD) =$
$\varnothing\ N\ ORD + \varnothing\ [D_{CUANT}(N)\ ORD + \alpha(N)\ D_{CUANT}(ORD)] = \varnothing$

$D_{POS}(ER_1) =$
$D_{POS}(POS)\ N\ ORD + \alpha(POS)\ D_{POS}(N\ ORD) =$
$\lambda\ N\ ORD + \varnothing\ [D_{POS}(N)\ ORD + \alpha(N)\ D_{POS}(ORD)] =$
$N\ ORD = ER_2$

$D_N(ER_1) =$
$D_N(POS)\ N\ ORD + \alpha(POS)\ D_N(N\ ORD) =$
$\varnothing\ N\ ORD + \varnothing\ [D_N(N)\ ORD + \alpha(N)\ D_N(ORD)] = \varnothing$

$D_{ORD}(ER_1) =$
$D_{ORD}(POS)\ N\ ORD + \alpha(POS)\ D_{ORD}(N\ ORD) =$
$\varnothing\ N\ ORD + \varnothing\ [D_{ORD}(N)\ ORD + \alpha(N)\ D_{ORD}(ORD)] = \varnothing$

$D_{CUANT}(ER_2) =$
$D_{CUANT}(N)\ ORD + \alpha(N)\ D_{CUANT}(ORD) =$
$\varnothing\ \ ORD + \varnothing = \varnothing$

$D_{POS}(ER_2) =$
$D_{POS}(N)\ ORD + \alpha(N)\ D_{POS}(ORD) =$
$\varnothing\ ORD + \varnothing\ \varnothing = \varnothing$

$D_N(ER_2) =$
$D_N(N)\ ORD + \alpha(N)\ D_N(ORD) =$
$\lambda\ ORD + \varnothing\ \varnothing =$
$ORD = ER_3$

$D_{ORD}(ER_2) =$
$D_{ORD}(N)\ ORD + \alpha(N)\ D_{ORD}(ORD) =$
$\varnothing\ ORD + \varnothing\ \varnothing = \varnothing$

$D_{CUANT}(ER_3) =$
$D_{CUANT}(ORD) = \varnothing$

$D_{POS}(ER_3) =$
$D_{POS}(ORD) = \varnothing$

$D_N(ER_3) =$
$D_N(ORD) = \varnothing$

$D_{ORD}(ER_3) =$
$D_{ORD}(ORD) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{49} es la siguiente:

$G = (\{CUANT, POS, N, ORD\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= \text{CUANT } ER_1$
$ER_1 ::= \text{POS } ER_2$
$ER_2 ::= \text{N } ER_3$
$ER_3 ::= \text{ORD}$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$$AF = (\{\text{CUANT}, \text{POS}, \text{N}, \text{ORD}\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$$

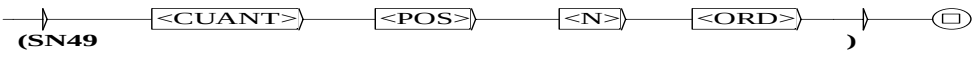
donde la función de transición, f, se define como:

$f(ER_0, \text{CUANT}) = ER_1$
$f(ER_0, \text{POS}) = \varnothing$
$f(ER_0, \text{N}) = \varnothing$
$f(ER_0, \text{ORD}) = \varnothing$
$f(ER_1, \text{CUANT}) = \varnothing$
$f(ER_1, \text{POS}) = ER_2$
$f(ER_1, \text{N}) = \varnothing$
$f(ER_1, \text{ORD}) = \varnothing$
$f(ER_2, \text{CUANT}) = \varnothing$
$f(ER_2, \text{POS}) = \varnothing$
$f(ER_2, \text{N}) = ER_3$
$f(ER_2, \text{ORD}) = \varnothing$
$f(ER_3, \text{CUANT}) = \varnothing$
$f(ER_3, \text{POS}) = \varnothing$
$f(ER_3, \text{N}) = \varnothing$
$f(ER_3, \text{ORD}) = F$
$f(F, \text{CUANT}) = \varnothing$
$f(F, \text{POS}) = \varnothing$
$f(F, \text{N}) = \varnothing$
$f(F, \text{ORD}) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, \text{CUANT}) = q_1$
$f(q_1, \text{POS}) = q_2$
$f(q_2, \text{N}) = q_3$
$f(q_3, \text{ORD}) = q_4$

El transductor gráfico que reconoce el sintagma SN_{49} es el siguiente (Fig. 6.52):



SN49 (6.52).grf

Fig. 6.52: FST gráfico que reconoce la estructura SN_{49}

51. **SN_{50} → CUANT POS CARD N (Cuantificador Posesivo Cardinal Nombre)**

ER_0 = CUANT POS CARD N

$D_{CUANT}(ER_0)=$
$D_{CUANT}(CUANT)$ POS CARD N + $\alpha(CUANT)$ $D_{CUANT}(POS\ CARD\ N)=$
λ POS CARD N + $\alpha(CUANT)$ [$D_{CUANT}(POS)$ CARD N + $\alpha(POS)$ $D_{CUANT}(CARD\ N)$]=
POS CARD N + $\varnothing$ [$D_{CUANT}(POS)$ CARD N + $\alpha(POS)$ $D_{CUANT}(CARD\ N)$]=
POS CARD N= ER_1

$D_{POS}(ER_0)=$
$D_{POS}(CUANT)$ POS CARD N + $\alpha(CUANT)$ $D_{POS}(POS\ CARD\ N)=$
$\varnothing$ POS CARD N + $\alpha(CUANT)$ [$D_{POS}(POS)$ CARD N + $\alpha(POS)$ $D_{POS}(CARD\ N)$]=
$\varnothing$ + $\varnothing$ [$D_{POS}(POS)$ CARD N + $\alpha(POS)$ $D_{POS}(CARD\ N)$]=$\varnothing$

$D_{ORD}(ER_0)=$
$D_{ORD}(CUANT)$ POS CARD N + $\alpha(CUANT)$ $D_{ORD}(POS\ CARD\ N)=$
$\varnothing$ POS CARD N + $\alpha(CUANT)$ [$D_{ORD}(POS)$ CARD N + $\alpha(POS)$ $D_{ORD}(CARD\ N)$]=
$\varnothing$ + $\varnothing$ [$D_{ORD}(POS)$ CARD N + $\alpha(POS)$ $D_{ORD}(CARD\ N)$]=$\varnothing$

$D_N(ER_0)=$
$D_N(CUANT)$ POS CARD N + $\alpha(CUANT)$ + $D_N(POS\ CARD\ N)=$
$\varnothing$ POS CARD N + $\alpha(CUANT)$ [$D_N(POS)$ CARD N + $\alpha(POS)$ $D_N(CARD\ N)$]=
$\varnothing$ + $\varnothing$ [$D_N(POS)$ CARD N + $\alpha(POS)$ $D_N(CARD\ N)$]=$\varnothing$

$D_{CUANT}(ER_1) =$
$D_{CUANT}(POS)\ CARD\ N + \alpha(POS)\ D_{CUANT}(CARD\ N) =$
$\varnothing\ CARD\ N + \varnothing\ [D_{CUANT}(CARD)\ N + \alpha(CARD)\ D_{CUANT}(N)] = \varnothing$

$D_{POS}(ER_1) =$
$D_{POS}(POS)\ CARD\ N + \alpha(POS)\ D_{POS}(CARD\ N) =$
$\lambda\ CARD\ N + \varnothing\ [D_{POS}(CARD)\ N + \alpha(CARD)\ D_{POS}(N)] =$
$CARD\ N = ER_2$

$D_{CARD}(ER_1) =$
$D_{CARD}(POS)\ CARD\ N + \alpha(POS)\ D_{CARD}(CARD\ N) =$
$\varnothing\ CARD\ N + \varnothing\ [D_{CARD}(CARD)\ N + \alpha(CARD)\ D_{CARD}(N)] = \varnothing$

$D_N(ER_1) =$
$D_N(POS)\ CARD\ N + \alpha(POS)\ D_N(CARD\ N) =$
$\varnothing\ CARD\ N + \varnothing\ [D_N(CARD)\ N + \alpha(CARD)\ D_N(N)] = \varnothing$

$D_{CUANT}(ER_2) =$
$D_{CUANT}(CARD)\ N + \alpha(CARD)\ D_{CUANT}(N) =$
$\varnothing\ \ N + \varnothing = \varnothing$

$D_{POS}(ER_2) =$
$D_{POS}(CARD)\ N + \alpha(CARD)\ D_{POS}(N) =$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_{CARD}(ER_2) =$
$D_{CARD}(CARD)\ N + \alpha(CARD)\ D_{CARD}(N) =$
$\lambda\ N + \varnothing\ \varnothing =$
$N = ER_3$

$D_N(ER_2) =$
$D_N(CARD)\ N + \alpha(CARD)\ D_N(N) =$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_{CUANT}(ER_3) =$
$D_{CUANT}(N) = \varnothing$

$D_{POS}(ER_3) =$
$D_{POS}(N) = \varnothing$

$D_{CARD}(ER_3) =$
$D_{ORD}(N) = \varnothing$

$D_N(ER_3) =$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{50} es la siguiente:

$G = (\{CUANT, POS, CARD, N\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= \text{CUANT } ER_1$
$ER_1 ::= \text{POS } ER_2$
$ER_2 ::= \text{CARD } ER_3$
$ER_3 ::= \text{N}$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$$AF = (\{\text{CUANT, POS, CARD, N}\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$$

donde la función de transición, f, se define como:

$f(ER_0, \text{CUANT}) = ER_1$
$f(ER_0, \text{POS}) = \varnothing$
$f(ER_0, \text{CARD}) = \varnothing$
$f(ER_0, \text{N}) = \varnothing$
$f(ER_1, \text{CUANT}) = \varnothing$
$f(ER_1, \text{POS}) = ER_2$
$f(ER_1, \text{CARD}) = \varnothing$
$f(ER_1, \text{N}) = \varnothing$
$f(ER_2, \text{CUANT}) = \varnothing$
$f(ER_2, \text{POS}) = \varnothing$
$f(ER_2, \text{CARD}) = ER_3$
$f(ER_2, \text{N}) = \varnothing$
$f(ER_3, \text{CUANT}) = \varnothing$
$f(ER_3, \text{POS}) = \varnothing$
$f(ER_3, \text{CARD}) = \varnothing$
$f(ER_3, \text{N}) = F$
$f(F, \text{CUANT}) = \varnothing$
$f(F, \text{POS}) = \varnothing$
$f(F, \text{CARD}) = \varnothing$
$f(F, \text{N}) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, \text{CUANT}) = q_1$
$f(q_1, \text{POS}) = q_2$
$f(q_2, \text{CARD}) = q_3$
$f(q_3, \text{N}) = q_4$

El transductor gráfico que reconoce el sintagma SN_{50} es el siguiente (Fig. 6.53):

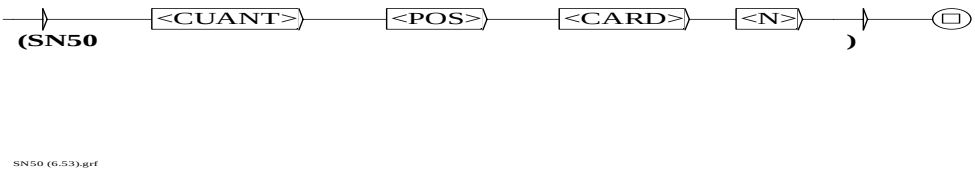


Fig. 6.53: FST gráfico que reconoce la estructura SN_{50}

52. **SN_{51} → DET ADV A N (Determinante Adverbio Adjetivo Nombre)**

ER_0 = DET ADV A N

$D_{DET}(ER_0)=$
$D_{DET}(DET)$ ADV A N + $\alpha(DET)$ $D_{DET}(ADV\ A\ N)=$
λ ADV A N + $\alpha(DET)$ [$D_{DET}(ADV)$ A N + $\alpha(ADV)$ $D_{DET}(A\ N)$]=
ADV A N + $\varnothing$ [$D_{DET}(ADV)$ A N + $\alpha(ADV)$ $D_{DET}(A\ N)$]=
ADV A N = ER_1

$D_{ADV}(ER_0)=$
$D_{ADV}(DET)$ ADV A N + $\alpha(DET)$ $D_{ADV}(ADV\ A\ N)=$
$\varnothing$ ADV A N + $\alpha(DET)$ [$D_{ADV}(ADV)$ A N + $\alpha(ADV)$ $D_{ADV}(A\ N)$]=
$\varnothing$ + $\varnothing$ [$D_{ADV}(ADV)$ A N + $\alpha(ADV)$ $D_{ADV}(A\ N)$]=$\varnothing$

$D_A(ER_0)=$
$D_A(DET)$ ADV A N + $\alpha(DET)$ $D_A(ADV\ A\ N)=$
$\varnothing$ ADV A N + $\alpha(DET)$ [$D_A(ADV)$ A N + $\alpha(ADV)$ $D_A(A\ N)$]=
$\varnothing$ + $\varnothing$ [$D_A(ADV)$ A N + $\alpha(ADV)$ $D_A(A\ N)$]=$\varnothing$

$D_N(ER_0)=$
$D_N(DET)$ ADV A N + $\alpha(DET)$ + $D_N(ADV\ A\ N)=$
$\varnothing$ ADV A N + $\alpha(DET)$ [$D_N(ADV)$ A N + $\alpha(ADV)$ $D_N(A\ N)$]=
$\varnothing$ + $\varnothing$ [$D_N(ADV)$ A N + $\alpha(ADV)$ $D_N(A\ N)$]=$\varnothing$

$D_{DET}(ER_1)=$
$D_{DET}(ADV)\ A\ N + \alpha(ADV)\ D_{DET}(A\ N)=$
$\varnothing\ A\ N + \varnothing\ [D_{DET}(A)\ N + \alpha(A)\ D_{DET}(N)]=\varnothing$

$D_{ADV}(ER_1)=$
$D_{ADV}(ADV)\ A\ N + \alpha(ADV)\ D_{ADV}(A\ N)=$
$\lambda\ A\ N + \varnothing\ [D_{ADV}(A)\ N + \alpha(A)\ D_{ADV}(N)]=$
$A\ N = ER_2$

$D_A(ER_1)=$
$D_A(ADV)\ A\ N + \alpha(ADV)\ D_A(A\ N)=$
$\varnothing\ A\ N + \varnothing\ [D_A(A)\ N + \alpha(A)\ D_A(N)]=\varnothing$

$D_N(ER_1)=$
$D_N(ADV)\ A\ N + \alpha(ADV)\ D_N(A\ N)=$
$\varnothing\ A\ N + \varnothing\ [D_N(A)\ N + \alpha(A)\ D_N(N)]=\varnothing$

$D_{DET}(ER_2)=$
$D_{DET}(A)\ N + \alpha(A)\ D_{DET}(N)=$
$\varnothing\ \ N + \varnothing = \varnothing$

$D_{ADV}(ER_2)=$
$D_{ADV}(A)\ N + \alpha(A)\ D_{ADV}(N)=$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_A(ER_2)=$
$D_A(A)\ N + \alpha(A)\ D_A(N)=$
$\lambda\ N + \varnothing\ \varnothing =$
$N = ER_3$

$D_N(ER_2)=$
$D_N(A)\ N + \alpha(A)\ D_N(N)=$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_{DET}(ER_3)=$
$D_{DET}(N) = \varnothing$

$D_{ADV}(ER_3)=$
$D_{ADV}(N) = \varnothing$

$D_A(ER_3)=$
$D_A(N)=\varnothing$

$D_N(ER_3)=$
$D_N(N)=\lambda$

La *Gramática Regular* que reconoce el sintagma SN_{51} es la siguiente:

$G = (\{DET, ADV, A, N\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

ER_0 ::= DET ER_1
ER_1 ::= ADV ER_2
ER_2 ::= A ER_3
ER_3 ::= N

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DET, ADV, A, N\}, \{ER_0, ER_1, ER_2, ER_3, F\}, f, ER_0, F)$

donde la función de transición, f , se define como:

$f(ER_0, DET) = ER_1$
$f(ER_0, ADV) = \varnothing$
$f(ER_0, A) = \varnothing$
$f(ER_0, N) = \varnothing$
$f(ER_1, DET) = \varnothing$
$f(ER_1, ADV) = ER_2$
$f(ER_1, A) = \varnothing$
$f(ER_1, N) = \varnothing$
$f(ER_2, DET) = \varnothing$
$f(ER_2, ADV) = \varnothing$
$f(ER_2, A) = ER_3$
$f(ER_2, N) = \varnothing$
$f(ER_3, DET) = \varnothing$
$f(ER_3, ADV) = \varnothing$
$f(ER_3, A) = \varnothing$
$f(ER_3, N) = F$
$f(F, DET) = \varnothing$
$f(F, ADV) = \varnothing$
$f(F, A) = \varnothing$
$f(F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, DET) = q_1$
$f(q_1, ADV) = q_2$
$f(q_2, A) = q_3$
$f(q_3, N) = q_4$

El transductor gráfico que reconoce el sintagma SN_{51} es el siguiente (Fig. 6.54):

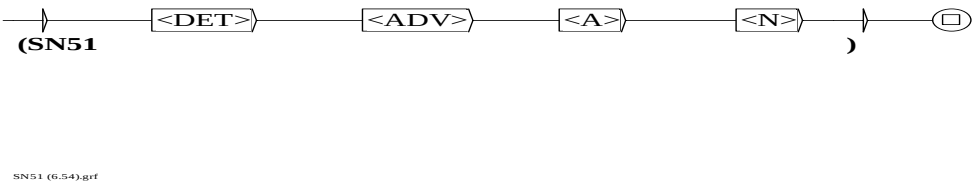


Fig. 6.54: FST gráfico que reconoce la estructura SN_{51}

53. **SN_{52} → DET ADV PA N (Determinante Adverbio Participio Nombre)**

ER_0 = DET ADV PA N

$D_{DET}(ER_0)=$
$D_{DET}(DET)$ ADV PA N + $\alpha(DET)$ $D_{DET}(ADV$ PA N$)=$
λ ADV PA N + $\alpha(DET)$ [$D_{DET}(ADV)$ PA N + $\alpha(ADV)$ $D_{DET}(PA$ N$)$]=
ADV PA N + $\varnothing$ [$D_{DET}(ADV)$ PA N + $\alpha(ADV)$ $D_{DET}(PA$ N$)$]=
ADV PA N = ER_1

$D_{ADV}(ER_0)=$
$D_{ADV}(DET)$ ADV PA N + $\alpha(DET)$ $D_{ADV}(ADV$ PA N$)=$
$\varnothing$ ADV PA N + $\alpha(DET)$ [$D_{ADV}(ADV)$ PA N + $\alpha(ADV)$ $D_{ADV}(PA$ N$)$]=
$\varnothing$ + $\varnothing$ [$D_{ADV}(ADV)$ PA N + $\alpha(ADV)$ $D_{ADV}(PA$ N$)$]=$\varnothing$

$D_{PA}(ER_0)=$
$D_{PA}(DET)$ ADV PA N + $\alpha(DET)$ $D_{PA}(ADV$ PA N$)=$
$\varnothing$ ADV PA N + $\alpha(DET)$ [$D_{PA}(ADV)$ PA N + $\alpha(ADV)$ $D_{PA}(PA$ N$)$]=
$\varnothing$ + $\varnothing$ [$D_{PA}(ADV)$ PA N + $\alpha(ADV)$ $D_{PA}(PA$ N$)$]=$\varnothing$

$D_N(ER_0)=$
$D_N(DET)$ ADV PA N + $\alpha(DET)$ + $D_N(ADV$ PA N$)=$
$\varnothing$ ADV PA N + $\alpha(DET)$ [$D_N(ADV)$ PA N + $\alpha(ADV)$ $D_N(PA$ N$)$]=
$\varnothing$ + $\varnothing$ [$D_N(ADV)$ PA N + $\alpha(ADV)$ $D_N(PA$ N$)$]=$\varnothing$

$D_{DET}(ER_1)=$
$D_{DET}(ADV)\ PA\ N + \alpha(ADV)\ D_{DET}(PA\ N)=$
$\varnothing\ PA\ N + \varnothing\ [D_{DET}(PA)\ N + \alpha(PA)\ D_{DET}(N)]=\varnothing$

$D_{ADV}(ER_1)=$
$D_{ADV}(ADV)\ PA\ N + \alpha(ADV)\ D_{ADV}(PA\ N)=$
$\lambda\ PA\ N + \varnothing\ [D_{ADV}(PA)\ N + \alpha(PA)\ D_{ADV}(N)]=$
$PA\ N = ER_2$

$D_{PA}(ER_1)=$
$D_{PA}(ADV)\ PA\ N + \alpha(ADV)\ D_{PA}(PA\ N)=$
$\varnothing\ PA\ N + \varnothing\ [D_{PA}(PA)\ N + \alpha(PA)\ D_{PA}(N)]=\varnothing$

$D_N(ER_1)=$
$D_N(ADV)\ PA\ N + \alpha(ADV)\ D_N(PA\ N)=$
$\varnothing\ PA\ N + \varnothing\ [D_N(PA)\ N + \alpha(PA)\ D_N(N)]=\varnothing$

$D_{DET}(ER_2)=$
$D_{DET}(PA)\ N + \alpha(PA)\ D_{DET}(N)=$
$\varnothing\ \ N + \varnothing = \varnothing$

$D_{ADV}(ER_2)=$
$D_{ADV}(PA)\ N + \alpha(PA)\ D_{ADV}(N)=$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_{PA}(ER_2)=$
$D_{PA}(PA)\ N + \alpha(PA)\ D_{PA}(N)=$
$\lambda\ N + \varnothing\ \varnothing =$
$N = ER_3$

$D_N(ER_2)=$
$D_N(PA)\ N + \alpha(PA)\ D_N(N)=$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_{DET}(ER_3)=$
$D_{DET}(N) = \varnothing$

$D_{ADV}(ER_3)=$
$D_{ADV}(N) = \varnothing$

$D_{PA}(ER_3)=$
$D_{PA}(N)=\varnothing$

$D_N(ER_3)=$
$D_N(N)=\lambda$

La *Gramática Regular* que reconoce el sintagma SN_{52} es la siguiente:

$G = (\{DET, ADV, PA, N\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

ER_0 ::= DET ER_1
ER_1 ::= ADV ER_2
ER_2 ::= PA ER_3
ER_3 ::= N

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{DET, ADV, PA, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

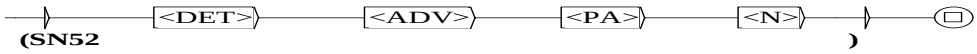
donde la función de transición, f , se define como:

$f\ (ER_0, DET) = ER_1$
$f\ (ER_0, ADV) = \varnothing$
$f\ (ER_0, PA) = \varnothing$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_1, DET) = \varnothing$
$f\ (ER_1, ADV) = ER_2$
$f\ (ER_1, PA) = \varnothing$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_2, DET) = \varnothing$
$f\ (ER_2, ADV) = \varnothing$
$f\ (ER_2, PA) = ER_3$
$f\ (ER_2, N) = \varnothing$
$f\ (ER_3, DET) = \varnothing$
$f\ (ER_3, ADV) = \varnothing$
$f\ (ER_3, PA) = \varnothing$
$f\ (ER_3, N) = F$
$f\ (F, DET) = \varnothing$
$f\ (F, ADV) = \varnothing$
$f\ (F, PA) = \varnothing$
$f\ (F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f\ (q_0, DET) = q_1$
$f\ (q_1, ADV) = q_2$
$f\ (q_2, PA) = q_3$
$f\ (q_3, N) = q_4$

El transductor gráfico que reconoce el sintagma SN_{52} es el siguiente (Fig. 6.55):



SN52 (6.55).grf

Fig. 6.55: FST gráfico que reconoce la estructura SN_{52}

54. **SN_{53} → CARD ORD N (Cardinal Ordinal Nombre)**

ER_0 = CARD ORD N

$D_{CARD}(ER_0)=$
$D_{CARD}(CARD)$ ORD N + $\alpha(CARD)$ $D_{CARD}(ORD\ N)=$
λ ORD N + $\alpha(CARD)$ [$D_{CARD}(ORD)$ N + $\alpha(ORD)$ $D_{CARD}(N)$]=
ORD N + $\varnothing$ [$\varnothing$ + $\varnothing$]=
ORD N = ER_1

$D_{ORD}(ER_0)=$
$D_{ORD}(CARD)$ ORD N + $\alpha(CARD)$ $D_{ORD}(ORD\ N)=$
$\varnothing$ ORD N + $\alpha(CARD)$ [$D_{ORD}(ORD)$ N + $\alpha(ORD)$ $D_{ORD}(N)$]=
$\varnothing$ + $\varnothing$ [λ N + $\varnothing$ $\varnothing$] = $\varnothing$

$D_N(ER_0)=$
$D_N(CARD)$ ORD N + $\alpha(CARD)$ + $D_N(ORD\ N)=$
$\varnothing$ ORD N + $\alpha(CARD)$ [$D_N(ORD)$ N + $\alpha(ORD)$ $D_N(N)$]=
$\varnothing$ + $\varnothing$ [$\varnothing$ N + $\varnothing$ λ] = $\varnothing$

$D_{CARD}(ER_1)=$
$D_{CARD}(ORD)$ N + $\alpha(ORD)$ $D_{CARD}(N)=$
$\varnothing$ N + $\varnothing$ $\varnothing$ = $\varnothing$

$D_{ORD}(ER_1)=$
$D_{ORD}(ORD)$ N + $\alpha(ORD)$ $D_{ORD}(N)=$
λ N + $\varnothing$ $\varnothing$ =
N = ER_2

$D_N(ER_1)=$
$D_N(ORD)$ N + $\alpha(ORD)$ $D_N(N)=$
$\varnothing$ N + $\varnothing$ λ = $\varnothing$

$D_{CARD}(ER_2) =$
$D_{CARD}(N) = \varnothing$

$D_{ORD}(ER_2) =$
$D_{ORD}(N) = \varnothing$

$D_{N}(ER_2) =$
$D_{N}(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{53} es la siguiente:

$G = (\{CARD, ORD, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= CARD\ ER_1$
$ER_1 ::= ORD\ ER_2$
$ER_2 ::= N$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$AF = (\{CARD, ORD, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, CARD) = ER_1$
$f(ER_0, ORD) = \varnothing$
$f(ER_0, N) = \varnothing$
$f(ER_1, CARD) = \varnothing$
$f(ER_1, ORD) = ER_2$
$f(ER_1, N) = \varnothing$
$f(ER_2, CARD) = \varnothing$
$f(ER_2, ORD) = \varnothing$
$f(ER_2, N) = F$
$f(F, CARD) = \varnothing$
$f(F, ORD) = \varnothing$
$f(F, N) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, CARD) = q_1$
$f(q_1, ORD) = q_2$
$f(q_2, N) = q_3$

El transductor gráfico que reconoce el sintagma SN_{53} es el siguiente (Fig. 6.56):

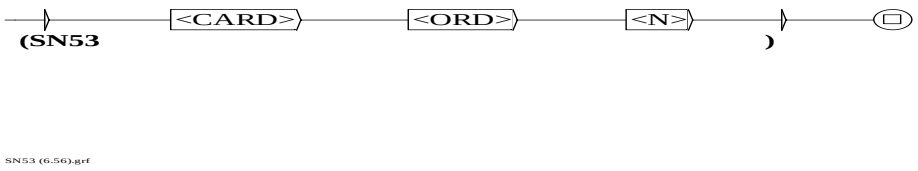


Fig. 6.56: FST gráfico que reconoce la estructura SN_{53}

55. **SN_{54} → DET CARD ORD N (Determinante Cardinal Ordinal Nombre)**

ER_0 = DET CARD ORD N

$D_{DET}(ER_0)=$
$D_{DET}(DET)$ CARD ORD N + $\alpha(DET)$ $D_{DET}(CARD\ ORD\ N)=$
λ CARD ORD N + $\alpha(DET)$ [$D_{DET}(CARD)$ ORD N + $\alpha(CARD)$ $D_{DET}(ORD\ N)$]=
CARD ORD N + $\varnothing$ [$D_{DET}(CARD)$ ORD N + $\alpha(CARD)$ $D_{DET}(ORD\ N)$]=
CARD ORD N = ER_1

$D_{CARD}(ER_0)=$
$D_{CARD}(DET)$ POS ORD N + $\alpha(DET)$ $D_{CARD}(POS\ ORD\ N)=$
$\varnothing$ POS ORD N + $\alpha(DET)$ [$D_{CARD}(POS)$ ORD N + $\alpha(POS)$ $D_{CARD}(ORD\ N)$]=
$\varnothing$ + $\varnothing$ [$D_{CARD}(POS)$ ORD N + $\alpha(POS)$ $D_{CARD}(ORD\ N)$]= $\varnothing$

$D_{ORD}(ER_0)=$
$D_{ORD}(DET)$ CARD ORD N + $\alpha(DET)$ $D_{ORD}(CARD\ ORD\ N)=$
$\varnothing$ CARD ORD N + $\alpha(DET)$ [$D_{ORD}(CARD)$ ORD N + $\alpha(CARD)$ $D_{ORD}(ORD\ N)$]=
$\varnothing$ + $\varnothing$ [$D_{ORD}(CARD)$ ORD N + $\alpha(CARD)$ $D_{ORD}(ORD\ N)$]= $\varnothing$

$D_N(ER_0)=$
$D_N(DET)$ CARD ORD N + $\alpha(DET)$ + $D_N(CARD\ ORD\ N)=$
$\varnothing$ CARD ORD N + $\alpha(DET)$ [$D_N(CARD)$ ORD N + $\alpha(CARD)$ $D_N(ORD\ N)$]=
$\varnothing$ + $\varnothing$ [$D_N(CARD)$ ORD N + $\alpha(CARD)$ $D_N(ORD\ N)$]= $\varnothing$

$D_{DET}(ER_1) =$
$D_{DET}(CARD)\ ORD\ N + \alpha(CARD)\ D_{DET}(ORD\ N) =$
$\varnothing\ ORD\ N + \varnothing\ [D_{DET}(ORD)\ N + \alpha(ORD)\ D_{DET}(N)] = \varnothing$

$D_{CARD}(ER_1) =$
$D_{CARD}(CARD)\ ORD\ N + \alpha(CARD)\ D_{CARD}(ORD\ N) =$
$\lambda\ ORD\ N + \varnothing\ [D_{CARD}(ORD)\ N + \alpha(ORD)\ D_{CARD}(N)] =$
$ORD\ N = ER_2$

$D_{ORD}(ER_1) =$
$D_{ORD}(CARD)\ ORD\ N + \alpha(CARD)\ D_{ORD}(ORD\ N) =$
$\varnothing\ ORD\ N + \varnothing\ [D_{ORD}(ORD)\ N + \alpha(ORD)\ D_{ORD}(N)] = \varnothing$

$D_N(ER_1) =$
$D_N(CARD)\ ORD\ N + \alpha(CARD)\ D_N(ORD\ N) =$
$\varnothing\ ORD\ N + \varnothing\ [D_N(ORD)\ N + \alpha(ORD)\ D_N(N)] = \varnothing$

$D_{DET}(ER_2) =$
$D_{DET}(ORD)\ N + \alpha(ORD)\ D_{DET}(N) =$
$\varnothing\ \ N + \varnothing = \varnothing$

$D_{CARD}(ER_2) =$
$D_{CARD}(ORD)\ N + \alpha(ORD)\ D_{CARD}(N) =$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_{ORD}(ER_2) =$
$D_{ORD}(ORD)\ N + \alpha(ORD)\ D_{ORD}(N) =$
$\lambda\ N + \varnothing\ \varnothing =$
$N = ER_3$

$D_N(ER_2) =$
$D_N(ORD)\ N + \alpha(ORD)\ D_N(N) =$
$\varnothing\ N + \varnothing\ \varnothing = \varnothing$

$D_{DET}(ER_3) =$
$D_{DET}(N) = \varnothing$

$D_{CARD}(ER_3) =$
$D_{CARD}(N) = \varnothing$

$D_{ORD}(ER_3) =$
$D_{ORD}(N) = \varnothing$

$D_N(ER_3) =$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el sintagma SN_{54} es la siguiente:

$G = (\{DET, CARD, ORD, N\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= \text{DET } ER_1$
$ER_1 ::= \text{CARD } ER_2$
$ER_2 ::= \text{ORD } ER_3$
$ER_3 ::= \text{N}$

El *Autómata Finito*, que reconoce el lenguaje generado por la gramática, es:

$$AF = (\{\text{DET, CARD, ORD, N}\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$$

donde la función de transición, f, se define como:

$f(ER_0, \text{DET}) = ER_1$
$f(ER_0, \text{CARD}) = \varnothing$
$f(ER_0, \text{ORD}) = \varnothing$
$f(ER_0, \text{N}) = \varnothing$
$f(ER_1, \text{DET}) = \varnothing$
$f(ER_1, \text{CARD}) = ER_2$
$f(ER_1, \text{ORD}) = \varnothing$
$f(ER_1, \text{N}) = \varnothing$
$f(ER_2, \text{DET}) = \varnothing$
$f(ER_2, \text{CARD}) = \varnothing$
$f(ER_2, \text{ORD}) = ER_3$
$f(ER_2, \text{N}) = \varnothing$
$f(ER_3, \text{DET}) = \varnothing$
$f(ER_3, \text{CARD}) = \varnothing$
$f(ER_3, \text{ORD}) = \varnothing$
$f(ER_3, \text{N}) = F$
$f(F, \text{DET}) = \varnothing$
$f(F, \text{CARD}) = \varnothing$
$f(F, \text{ORD}) = \varnothing$
$f(F, \text{N}) = \varnothing$

Se eliminan las transiciones vacías, se renombran los estados y se redefine la función de transición del AF Mínimo:

$f(q_0, \text{DET}) = q_1$
$f(q_1, \text{CARD}) = q_2$
$f(q_2, \text{ORD}) = q_3$
$f(q_3, \text{N}) = q_4$

El transductor gráfico que reconoce el sintagma SN_{54} es el siguiente (Fig. 6.57):

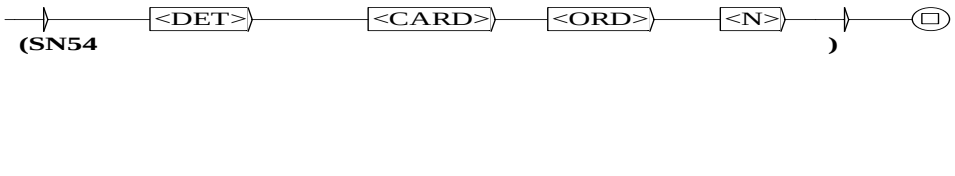


Fig. 6.57: FST gráfico que reconoce la estructura SN_{54}

Además, para agrupar las secuencias sintagmáticas anteriores y sus variantes relacionándolas con construcciones sintácticas canónicas se va a emplear el *operador de unión* de Expresiones Regulares, y a continuación se va a construir el transductor gráfico que encargará de relacionar y etiquetar las distintas construcciones sintagmáticas con un SN normalizado (Fig. 6.58):

SN_{55} → N
N N
DET N
CUANT N
POS N
ORD N
DEM N
CARD N
CUANT POS N
CUANT DET N
CUANT DEM N
DEM ORD N
...

ER_0 = N + N N + DET N + CUANT N + POS N + ORD N + DEM N + CARD N + CUANT POS N + CUANT DET N + CUANT DEM N + DEM ORD N + DEM CARD N + DET ORD N + DET CARD N + POS ORD N + POS CARD N + CUANT DEM N + CUANT DEM CARD N + CUANT DET ORD N + CUANT DET CARD N + CUANT POS ORD N + CUANT POS CARD N + CARD ORD N + DET CARD ORD N

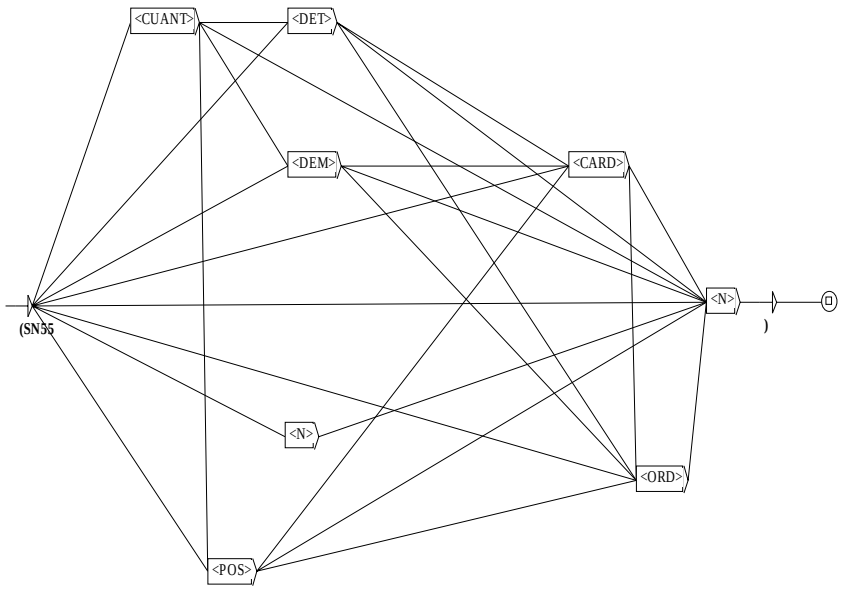


Fig. 6.58: FST gráfico que reconoce las variantes de la estructura SN_{55}

El lenguaje que genera, reconoce y etiqueta el transductor gráfico anterior es:

<N> => (SN55)
<POS> <N> => (SN55)
<N> <N> => (SN55)
<ORD> <N> => (SN55)
<DET> <N> => (SN55)
<DEM> <N> => (SN55)
<CARD> <N> => (SN55)
<CUANT> <N> => (SN55)
<CUANT> <POS> <N> => (SN55)
<CUANT> <DET> <N> => (SN55)
<CUANT> <DEM> <N> => (SN55)
<CARD> <ORD> <N> => (SN55)

<DEM> <ORD> <N> => (SN55)
<DEM> <CARD> <N> => (SN55)
<DET> <ORD> <N> => (SN55)
<DET> <CARD> <N> => (SN55)
<POS> <ORD> <N> => (SN55)
<POS> <CARD> <N> => (SN55)
<POS> <CARD> <ORD> <N> => (SN55)
<DET> <CARD> <ORD> <N> => (SN55)
<DEM> <CARD> <ORD> <N> => (SN55)
<CUANT> <DEM> <ORD> <N> => (SN55)
<CUANT> <DEM> <CARD> <N> => (SN55)
<CUANT> <DET> <ORD> <N> => (SN55)
<CUANT> <DET> <CARD> <N> => (SN55)
<CUANT> <POS> <ORD> <N> => (SN55)
<CUANT> <POS> <CARD> <N> => (SN55)
<CUANT> <POS> <CARD> <ORD> <N> => (SN55)
<CUANT> <DET> <CARD> <ORD> <N> => (SN55)
<CUANT> <DEM> <CARD> <ORD> <N> => (SN55)

De la misma forma, se desarrolla un transductor gráfico que agrupa las variantes sintagmática en las que aparece un *nombre* junto a un *adjetivo*, o *participio-adjetivo*, en posición prenominal, o posnominal (Fig. 6.59):

SN_{56} → N A
A N
N PA
PA N
DET A N
DET N A
DET PA N
DET N PA
CUANT A N
CUANT N A
CUANT PA N
CUANT N PA
DEM A N
...

ER_0 = N A + A N + N PA + PA N + DET A N + DET N A + DET PA N + DET N PA + CUANT A N + CUANT N A + CUANT PA N + CUANT N PA + DEM A N + DEM N A + DEM PA N + DEM N PA + A N A + A N PA + PA N A + PA N PA + POS N A + POS A N + POS N PA + POS A N A + POS A N PA + POS PA N A + POS PA N PA + DET A N A + DET A N PA + DET PA N A + DET PA N PA + DEM A N A + DEM A N PA + DEM PA N PA + CUANT A N A + CUANT A N PA + CUANT PA N A + CUANT PA N PA + DET ADV A N + DET ADV PA N + DET ADV A N A + DET ADV A N PA + DET ADV PA N A

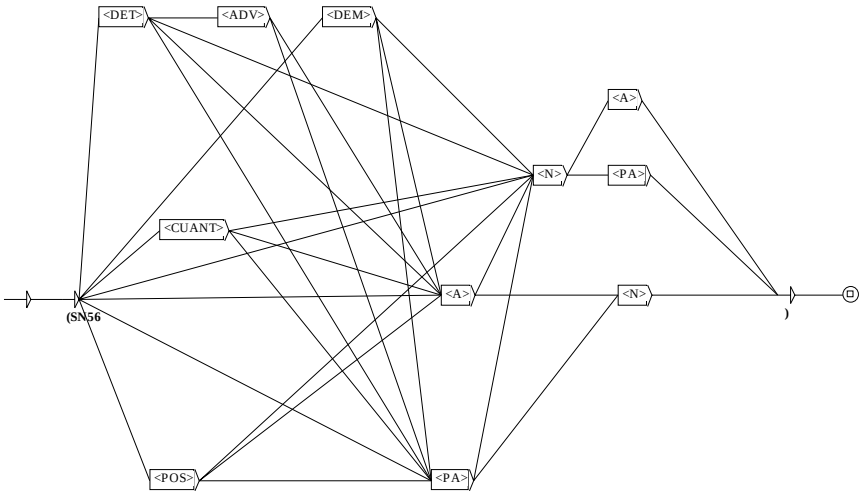


Fig. 6.59: FST gráfico que reconoce las variantes de la estructura SN_{56}

El lenguaje que genera, reconoce y etiqueta el transductor anterior es:

<PA> <N> => (SN56)
<N> <A> => (SN56)
<N> <PA> => (SN56)
<A> <N> => (SN56)
<CUANT> <PA> <N> => (SN56)
<CUANT> <N> <A> => (SN56)
<CUANT> <N> <PA> => (SN56)
<CUANT> <A> <N> => (SN56)
<DEM> <PA> <N> => (SN56)
<DEM> <N> <A> => (SN56)
<DEM> <N> <PA> => (SN56)
<DEM> <A> <N> => (SN56)
<DET> <PA> <N> => (SN56)
<DET> <N> <A> => (SN56)
<DET> <N> <PA> => (SN56)
<DET> <A> <N> => (SN56)
<A> <N> <A> => (SN56)
<A> <N> <PA> => (SN56)
<PA> <N> <A> => (SN56)

<PA> <N> <PA> => (SN56)
<POS> <PA> <N> => (SN56)
<POS> <N> <A> => (SN56)
<POS> <N> <PA> => (SN56)
<POS> <A> <N> => (SN56)
<POS> <A> <N> <A> => (SN56)
<POS> <A> <N> <PA> => (SN56)
<POS> <PA> <N> <A> => (SN56)
<POS> <PA> <N> <PA> => (SN56)
<DET> <ADV> <PA> <N> => (SN56)
<DET> <ADV> <A> <N> => (SN56)
<DET> <A> <N> <A> => (SN56)
<DET> <A> <N> <PA> => (SN56)
<DET> <PA> <N> <A> => (SN56)
<DET> <PA> <N> <PA> => (SN56)
<DEM> <A> <N> <A> => (SN56)
<DEM> <A> <N> <PA> => (SN56)
<DEM> <PA> <N> <A> => (SN56)
<DEM> <PA> <N> <PA> => (SN56)
<CUANT> <A> <N> <A> => (SN56)
<CUANT> <A> <N> <PA> => (SN56)
<CUANT> <PA> <N> <A> => (SN56)
<CUANT> <PA> <N> <PA> => (SN56)
<DET> <ADV> <A> <N> <A> => (SN56)
<DET> <ADV> <A> <N> <PA> => (SN56)
<DET> <ADV> <PA> <N> <A> => (SN56)
<DET> <ADV> <PA> <N> <PA> => (SN56)

Por último, para reconocer una selección de SSNN se desarrolla un único transductor gráfico que vincula un grupo de sintagmas imbricados y es capaz de reconocerlos de forma conjunta en los textos (Fig. 6.60).

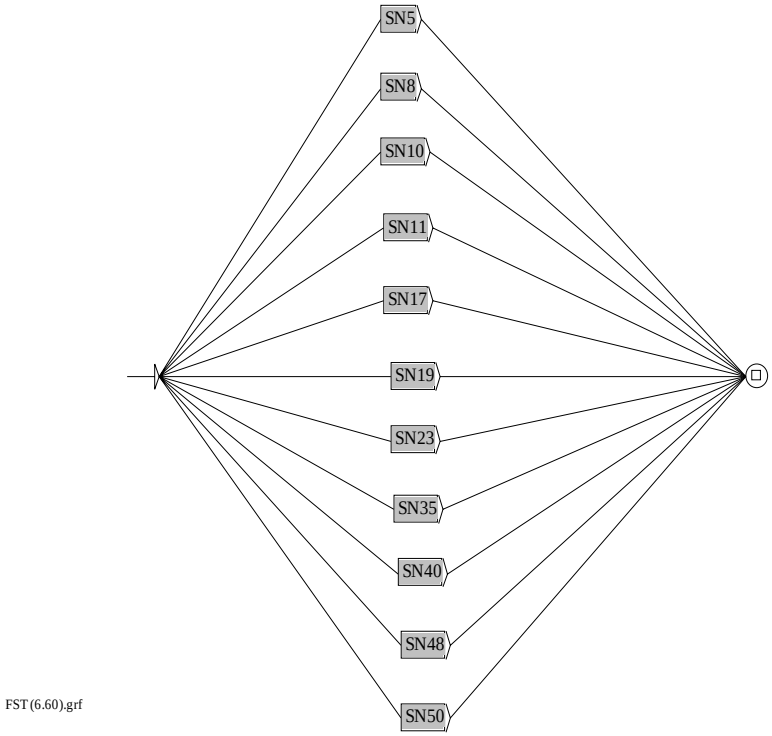


Fig. 6.60: FST gráfico que vincula un grupo de SSNN

6.1.2. SSNN de Estructura Simple con iteración de constituyentes

Con el objetivo de desarrollar las gramáticas que sean capaces de generar las estructuras simples de los SSNN en las que se produce iteración de constituyentes vamos a aplicar la misma metodología que hemos desarrollado en el apartado anterior. En primer lugar, se van a especificar las estructuras de los SSNN; en segundo lugar, se van a definir en términos de *Expresiones Regulares* por medio de la *convención de Kleene*; en tercer lugar, se van a obtener las *Gramáticas Parciales* que sean capaces de generarlas; y en cuarto lugar, se van a desarrollar los autómatas y transductores gráficos que se encarguen de reconocer e insertar etiquetas a los SSNN identificados por medio del

editor *FSGraph*. Para evitar que el desarrollo de las derivaciones de las Expresiones Regulares se extienda demasiado, debido a que la iteración de constituyentes provoca una proliferación de construcciones sintagmáticas, se van a agrupar las expresiones similares del modo siguiente:

1. **SN_{57} → DET N^+ (Determinante iter. Nombre)**

ER_0 = DET N^+
N $^+$
CUANT N^+
POS N^+
DEM N^+
CARD N^+
ORD N^+
DET CARD N^+
DET ORD N^+
DEM CARD N^+
DEM ORD N^+
POS CARD N^+
POS ORD N^+

ER_0 = CUANT DET N^+
CUANT DEM N^+
CUANT POS N^+
CUANT DET CARD N^+
CUANT DET ORD N^+
CUANT DEM CARD N^+
CUANT DEM ORD N^+
CUANT POS CARD N^+
CUANT POS ORD N^+

$$D_{DET}(ER_0) =$$
$$\left[D_{DET}(DET)\ N^+ + \alpha(DET)\ D_{DET}\left(N^+\right) \right] =$$
$$\lambda\ N^+ + \varnothing = N^+ = ER_1$$

$$D_N(ER_0) =$$
$$\left[D_N(DET)\ N^+ + \alpha(DET)\ D_N\left(N^+\right) \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_{DET}(ER_1) =$$
$$D_{DET}\left(N^+\right) =$$
$$D_{DET}\left(N\ N^*\right) =$$
$$\left[D_{DET}(N)\ N^* + \alpha(N)\ D_{DET}\left(N^*\right) \right] =$$
$$\varnothing\ N^* + \varnothing = \varnothing$$

$$D_N(ER_1) =$$
$$D_N\left(N^+\right) =$$
$$D_N\left(N\ N^*\right) =$$
$$\left[D_N(N)\ N^* + \alpha(N)\ D_N\left(N^*\right) \right] =$$
$$\lambda\ N^* + \varnothing =$$
$$N^* = ER_2$$

$$D_{DET}(ER_2) =$$
$$D_{DET}(N)\ N^* = \varnothing$$

$$D_N(ER_2) =$$
$$D_N(N)\ N^* =$$
$$\lambda\ N^* = ER_2$$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_{57} es la siguiente:

$$G = (\{DET, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$$

donde las *reglas de producción*, P, se definen como:

$$ER_0 ::= DET\ ER_1$$
$$ER_1 ::= N\ ER_2 \mid N$$
$$ER_2 ::= N\ ER_2 \mid N$$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$$AF = (\{DET, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$$

donde la función de transición, f, se define como:

$f\ (\mathrm{ER}_0, \mathrm{DET}) = \mathrm{ER}_1$
$f\ (\mathrm{ER}_0, \mathrm{N}) = \varnothing$
$f\ (\mathrm{ER}_1, \mathrm{DET}) = \varnothing$
$f\ (\mathrm{ER}_1, \mathrm{N}) = \{\mathrm{ER}_2, F\}$
$f\ (\mathrm{ER}_2, \mathrm{DET}) = \varnothing$
$f\ (\mathrm{ER}_2, \mathrm{N}) = \{\mathrm{ER}_2, F\}$
$f\ (F, \mathrm{DET}) = \varnothing$
$f\ (F, \mathrm{N}) = \varnothing$

se renombran los estados:

$f\ (q_0, \mathrm{DET}) = q_1$
$f\ (q_0, \mathrm{N}) = \varnothing$
$f\ (q_1, \mathrm{DET}) = \varnothing$
$f\ (q_1, \mathrm{N}) = q_2$
$f\ (q_2, \mathrm{DET}) = \varnothing$
$f\ (q_2, \mathrm{N}) = q_2$

y se eliminan las transiciones vacías:

$f\ (q_0, \mathrm{DET}) = q_1$
$f\ (q_1, \mathrm{N}) = q_2$
$f\ (q_2, \mathrm{N}) = q_2$

El autómata obtenido no se puede reducir y se considera el *Autómata Finito Determinista Mínimo*. A su vez, el AFD se puede representa en un *diagrama de transiciones* (Fig. 6.61), o en una *tabla de transiciones* (Fig. 6.62).

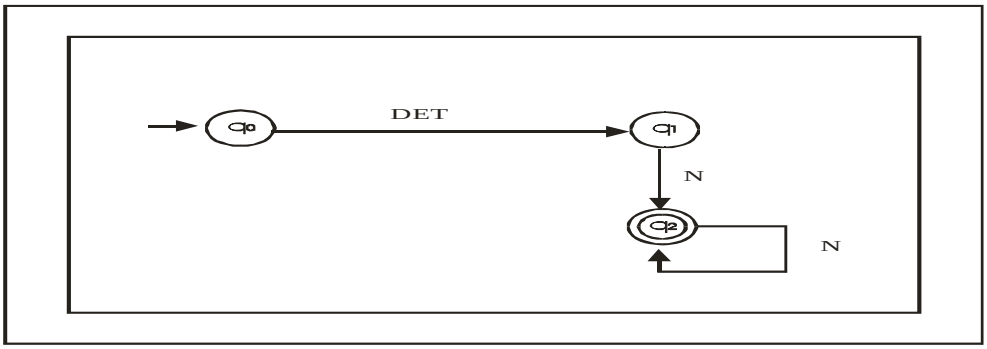


Fig. 6.61 : Diagrama de transiciones del AFD que reconoce la estructura SN_{57}

f	D E T	N
→ q_0	q_1	∅
q_1	∅	q_2
* q_2	∅	q_2

Fig. 6.62: Tabla de transiciones del AFD que reconoce la estructura SN_{57}

Por último, el autómata se va a representar en un transductor gráfico que se encarga de asignar marcas al SN identificado (Fig. 6.63).

SN57 (6.63).grf

Fig. 6.63: FST gráfico que reconoce la estructura SN_{57}

A partir de ahora la fase de representación de los autómatas en *diagramas de transiciones* y *tablas de transiciones* se va a excluir con el objetivo de reducir todo el proceso. De tal forma que, una vez obtenidas las gramáticas que generen los SSNN, los autómatas se van a representar directamente en FST gráficos. Además, para evitar que el desarrollo de las gramáticas y de los autómatas sea demasiado extenso, hemos decido agrupar los SSNN con estructura similares e incorporarlos directamente al editor *FSGraph*. El transductor gráfico obtenido nos va a permitir identificar y agrupar todas las variantes de lo que consideramos una misma estructura sintagmática (Fig. 6.64).

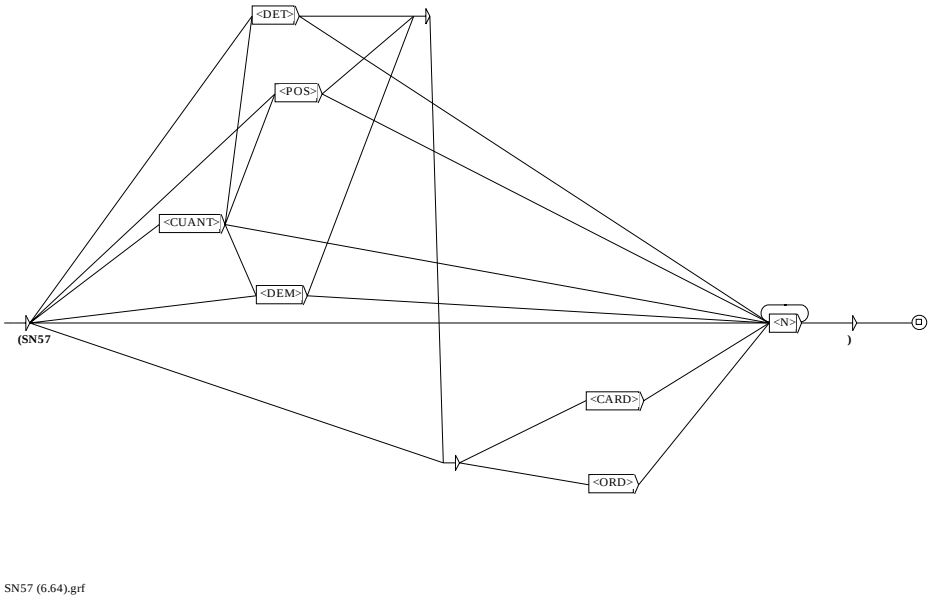


Fig. 6.64: FST gráfico que agrupa las variantes de la estructura SN_{57}

2. **SN_{58} → DET N^+ A (Determinante iter. Nombre Adjetivo)**

ER_0 = DET N^+ A
N^+ A
N^+ PA
N^+ CARD
N^+ ORD
CUANT N^+ A
POS N^+ A
DEM N^+ A
DET N^+ PA
CUANT N^+ PA
POS N^+ PA
DEM N^+ PA

$$ER_0 = \text{DET } N^+ \text{ CARD}$$
$$\text{CUANT } N^+ \text{ CARD}$$
$$\text{POS } N^+ \text{ CARD}$$
$$\text{DEM } N^+ \text{ CARD}$$
$$\text{DET } N^+ \text{ ORD}$$
$$\text{CUANT } N^+ \text{ ORD}$$
$$\text{POS } N^+ \text{ ORD}$$
$$\text{DEM } N^+ \text{ ORD}$$

$$D_{DET}(ER_0) =$$
$$\left[D_{DET}(DET)\ N^+\ A + \alpha(DET)\ D_{DET}\left(N^+\right)\ A \right] =$$
$$\lambda\ N^+\ A + \varnothing = N^+\ A = ER_1$$

$$D_N(ER_0) =$$
$$\left[D_N(DET)\ N^+\ A + \alpha(DET)\ D_N\left(N^+\right)\ A \right] =$$
$$\varnothing\ +\ \varnothing = \varnothing$$

$$D_A(ER_0) =$$
$$\left[D_A(DET)\ N^+\ A + \alpha(DET)\ D_A\left(N^+\right)\ A \right] =$$
$$\varnothing\ +\ \varnothing = \varnothing$$

$D_{DET}(ER_1) =$
$D_{DET}(N^+)\ A =$
$D_{DET}(N\ N^*)\ A =$
$[\ D_{DET}(N)\ N^*\ A + \alpha(N)\ D_{DET}(N^*\ A)\ ] =$
$\varnothing\ N^*\ A + \varnothing = \varnothing$

$D_N(ER_1) =$
$D_N(N^+\ A) =$
$D_N(N\ N^*\ A) =$
$[\ D_N(N)\ N^*\ A + \alpha(N)\ D_N(N^*\ A)\ ] =$
$\lambda\ N^*\ A + \varnothing =$
$N^*\ A = ER_2$

$D_A(ER_1) =$
$D_A(N^+\ A) =$
$D_A(N\ N^*\ A) =$
$[\ D_A(N)\ N^*\ A + \alpha(N)\ D_A(N^*\ A)\ ] =$
$\varnothing\ N^*\ A + \varnothing = \varnothing$

$D_{DET}(ER_2) =$
$D_{DET}(N^*)\ A + \alpha(N^*)\ D_{DET}(A) =$
$D_{DET}(N)\ N^*A + \lambda\ \varnothing =$
$\varnothing\ N^*\ A + \varnothing = \varnothing$

$D_N(ER_2) =$
$D_N(N^*)\ A + \alpha(N^*)\ D_N(A) =$
$D_N(N)\ N^*A + \lambda\ \varnothing =$
$\lambda\ N^*\ A = ER_2$

$D_A(ER_2) =$
$D_A(N^*)\ A + \alpha(N^*)\ D_A(A) =$
$D_A(N)\ N^*A + \lambda\ \lambda =$
$\varnothing\ N^*A + \lambda = \lambda$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_{58} es la siguiente:

$G = (\{DET, N, A\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= \text{DET } ER_1$
$ER_1 ::= \text{N } ER_2$
$ER_2 ::= \text{N } ER_2 \mid \text{A}$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$AF = (\{\text{DET, N, A}\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, \text{DET}) = ER_1$
$f(ER_0, \text{N}) = \varnothing$
$f(ER_0, \text{A}) = \varnothing$
$f(ER_1, \text{DET}) = \varnothing$
$f(ER_1, \text{N}) = ER_2$
$f(ER_1, \text{A}) = \varnothing$
$f(ER_2, \text{DET}) = \varnothing$
$f(ER_2, \text{N}) = ER_2$
$f(ER_2, \text{A}) = F$
$f(F, \text{DET}) = \varnothing$
$f(F, \text{N}) = \varnothing$
$f(F, \text{A}) = \varnothing$

se renombran los estados:

$f(q_0, \text{DET}) = q_1$
$f(q_0, \text{N}) = \varnothing$
$f(q_0, \text{A}) = \varnothing$
$f(q_1, \text{DET}) = \varnothing$
$f(q_1, \text{N}) = q_2$
$f(q_1, \text{A}) = \varnothing$
$f(q_2, \text{DET}) = \varnothing$
$f(q_2, \text{N}) = q_2$
$f(q_2, \text{A}) = q_3$

y se eliminan las transiciones vacías:

$f(q_0, \text{DET}) = q_1$
$f(q_1, \text{N}) = q_2$
$f(q_2, \text{N}) = q_2$
$f(q_2, \text{A}) = q_3$

A continuación, el autómata obtenido se representa en un transductor gráfico que se encarga de asignar marcas a las variantes de los SSNN identificados (Fig. 6.65).

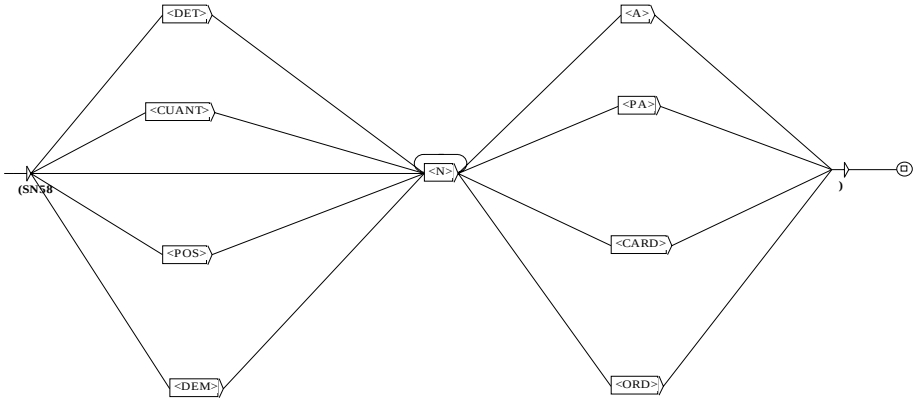


Fig. 6.65: FST gráfico que agrupa las variantes de la estructura SN_{58}

3. **SN_{59} → DET N A^+ (Determinante Nombre iter. Adjetivo)**

ER_0 = DET N A^+
N A^+
N PA^+
CUANT N A^+
POS N A^+
DEM N A^+
DET N PA^+
CUANT N PA^+
POS N PA^+
DEM N PA^+

$$D_{DET}(ER_0) =$$
$$\left[D_{DET}(DET)\ N\ A^+ + \alpha(DET)\ D_{DET}(N)\ A^+ \right] =$$
$$\lambda\ N\ A^+ + \varnothing = N\ A^+ = ER_1$$

$$D_N(ER_0) =$$
$$\left[D_N(DET)\ N\ A^+ + \alpha(DET)\ D_N(N)\ A^+ \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_A(ER_0) =$$
$$\left[D_A(DET)\ N\ A^+ + \alpha(DET)\ D_A(N)\ A^+ \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_{DET}(ER_1) =$$
$$D_{DET}(N)\ A^+ =$$
$$D_{DET}(N)\ A^+ =$$
$$\varnothing\ \ A^+ = \varnothing$$

$$D_N(ER_1) =$$
$$D_N(N)\ A^+ =$$
$$\lambda\ A^+ = A^+ = ER_2$$

$$D_A(ER_1) =$$
$$D_A(N)\ A^+ =$$
$$\varnothing\ A^+ = \varnothing$$

$$D_{DET}(ER_2) =$$
$$D_{DET}\left(A^+\right) =$$
$$D_{DET}\left(A\ A^*\right) =$$
$$\left[D_{DET}(A)\ A^* + \alpha(A)\ D_{DET}\left(A^*\right) \right] =$$
$$\varnothing\ A^* + \varnothing = \varnothing$$

$$D_N(ER_2) =$$
$$D_N\left(A^+\right) =$$
$$D_N\left(A\ A^*\right) =$$
$$\left[D_N(A)\ A^* + \alpha(A)\ D_N\left(A^*\right) \right] =$$
$$\varnothing\ A^* + \varnothing = \varnothing$$

$$D_A(ER_2) =$$
$$D_A\left(A^+\right) =$$
$$D_A\left(A\ A^*\right)$$
$$\left[D_A(A)\ A^* + \alpha(A)\ D_A\left(A^*\right) \right] =$$
$$\lambda\ A^* + \varnothing =$$
$$A^* = ER_3$$

$D_{DET}(ER_3) =$
$D_{DET}(A^*) =$
$D_{DET}(A)\ A^* =$
$\varnothing\ A^* = \varnothing$

$D_N(ER_3) =$
$D_N(A^*) =$
$D_N(A)\ A^* =$
$\varnothing\ A^* = \varnothing$

$D_A(ER_3) =$
$D_A(A^*) =$
$D_A(A)\ A^* =$
$\lambda\ A^* = ER_3$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_{59} es la siguiente:

$G = (\{DET, N, A\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= N\ ER_2$
$ER_2 ::= A\ ER_3 \mid A$
$ER_3 ::= A\ ER_3 \mid A$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$AF = (\{DET, N, A\}, \{ER_0, ER_1, ER_2, ER_3, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, DET) = ER_1$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_0, A) = \varnothing$
$f\ (ER_1, DET) = \varnothing$
$f\ (ER_1, N) = ER_2$
$f\ (ER_1, A) = \varnothing$
$f\ (ER_2, DET) = \varnothing$
$f\ (ER_2, N) = \varnothing$
$f\ (ER_2, A) = \{ER_3, F\}$
$f\ (ER_3, DET) = \varnothing$
$f\ (ER_3, N) = \varnothing$
$f\ (ER_3, A) = \{ER_3, F\}$
$f\ (F, DET) = \varnothing$
$f\ (F, N) = \varnothing$
$f\ (F, A) = \varnothing$

se renombran los estados:

$f\ (q_0, DET) = q_1$
$f\ (q_0, N) = \varnothing$
$f\ (q_0, A) = \varnothing$
$f\ (q_1, DET) = \varnothing$
$f\ (q_1, N) = q_2$
$f\ (q_1, A) = \varnothing$
$f\ (q_2, DET) = \varnothing$
$f\ (q_2, N) = \varnothing$
$f\ (q_2, A) = q_3$
$f\ (q_3, DET) = \varnothing$
$f\ (q_3, N) = \varnothing$
$f\ (q_3, A) = q_3$

y se eliminan las transiciones vacías:

$f\ (q_0, DET) = q_1$
$f\ (q_1, N) = q_2$
$f\ (q_2, A) = q_3$
$f\ (q_3, A) = q_3$

El autómata obtenido se representa en un transductor gráfico que se encarga de asignar marcas a las variantes de los SSNN identificados (Fig. 6.66).

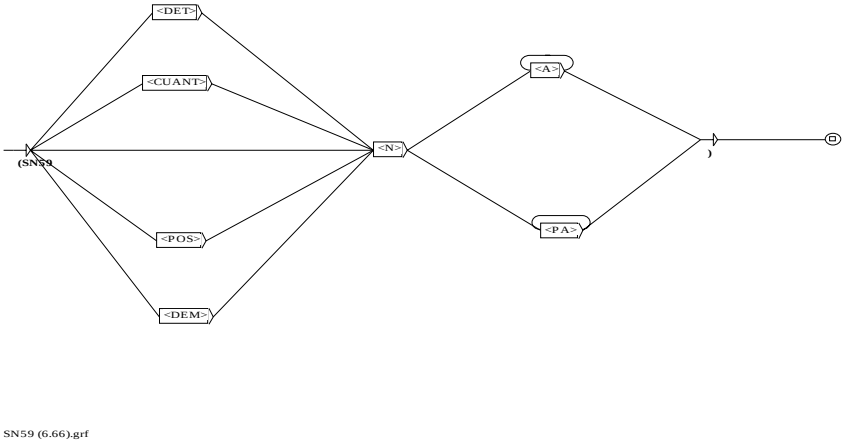


Fig. 6.66: FST gráfico que agrupa las variantes de la estructura SN_{59}

4. **SN_{60} → DET N^+ A^+** (**Determinante iter. Nombre iter. Adjetivo**)

ER_0 = DET N^+ A^+
N^+ A^+
N^+ PA^+
CUANT N^+ A^+
POS N^+ A^+
DEM N^+ A^+
DET N^+ PA^+
CUANT N^+ PA^+
POS N^+ PA^+
DEM N^+ PA^+

$$D_{DET}(ER_0) =$$
$$\left[D_{DET}(DET)\ N^+\ A^+ + \alpha(DET)\ D_{DET}\left(N^+\right)\ A^+ \right] =$$
$$\lambda\ N^+\ A^+ + \varnothing = N^+\ A^+ = ER_1$$

$$D_N(ER_0) =$$
$$\left[D_N(DET)\ N^+\ A^+ + \alpha(DET)\ D_N\left(N^+\right)\ A^+ \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_A(ER_0) =$$
$$\left[D_A(DET)\ N^+\ A^+ + \alpha(DET)\ D_A\left(N^+\right)\ A^+ \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_{DET}(ER_1) =$$
$$D_{DET}\left(N^+\ A^+\right) =$$
$$D_{DET}\left(N\ N^*\ A^+\right) =$$
$$\left[D_{DET}(N)\ N^*\ A^+ + \alpha(N)\ D_{DET}\left(N^*\ A\right) \right] =$$
$$\varnothing\ N^*\ A^+ + \varnothing = \varnothing$$

$$D_N(ER_1) =$$
$$D_N\left(N^+\ A^+\right) =$$
$$D_N\left(N\ N^*\ A^+\right) =$$
$$\left[D_N(N)\ N^*\ A^+ + \alpha(N)\ D_N\left(N^*\ A\right) \right] =$$
$$\lambda\ N^*\ A^+ + \varnothing =$$
$$N^*\ A^+ = ER_2$$

$$D_A(ER_1) =$$
$$D_A(N)\ A^+ =$$
$$\varnothing\ A^+ = \varnothing$$

$$D_{DET}(ER_2) =$$
$$D_{DET}\left(N^*\ A^+\right) =$$
$$\left[D_{DET}\left(N^*\right) A^+ + \alpha\left(N^*\right) D_{DET}\left(A^+\right)\right] =$$
$$\varnothing\ A^+ + \lambda\ D_{DET}\left(A^+\right) = \varnothing$$

$$D_N(ER_2) =$$
$$D_N\left(N^*\ A^+\right) =$$
$$\left[D_N\left(N^*\right) A^+ + \alpha\left(N^*\right) D_N\left(A^+\right)\right] =$$
$$D_N(N)\ N^*\ A^+ + \lambda\ D_N\left(A\ A^*\right) =$$
$$\lambda\ N^*\ A^+ = ER_2$$

$$D_A(ER_2) =$$
$$D_A\left(N^*\ A^+\right) =$$
$$\left[D_A\left(N^*\right) A^+ + \alpha\left(N^*\right) D_A\left(A^+\right)\right] =$$
$$\left[D_A(N)\ N^*\ A^+ + \alpha\left(N^*\right) D_A\left(A\ A^*\right)\right] =$$
$$\varnothing\ N^*\ A^+ + \lambda\ \left[D_A(A)\ A^* + \alpha(A)\ D_A\left(A^*\right)\right] =$$
$$\lambda\left[\lambda\ A^* + \varnothing\ D_A\left(A^*\right)\right] =$$
$$A^* = ER_3$$

$$D_{DET}(ER_3) =$$
$$D_{DET}(A)\ A^* = \varnothing$$

$$D_N(ER_3) =$$
$$D_N(A)\ A^* = \varnothing$$

$$D_A(ER_3) =$$
$$D_A(A)\ A^* =$$
$$\lambda\ A^* = ER_3$$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_{60} es la siguiente:

$$G = \left(\{DET, N, A\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P\right)$$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= N\ ER_2$
$ER_2 ::= N\ ER_2 \mid A\ ER_3 \mid A$
$ER_3 ::= A\ ER_3 \mid A$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$$AF = (\{DET, N, A\}, \{ER_0, ER_1, ER_2, ER_3, F\}, f, ER_0, F)$$

donde la función de transición, f, se define como:

$f(ER_0, DET) = ER_1$
$f(ER_0, N) = \varnothing$
$f(ER_0, A) = \varnothing$
$f(ER_1, DET) = \varnothing$
$f(ER_1, N) = ER_2$
$f(ER_1, A) = \varnothing$
$f(ER_2, DET) = \varnothing$
$f(ER_2, N) = ER_2$
$f(ER_2, A) = \{ER_3, F\}$
$f(ER_3, DET) = \varnothing$
$f(ER_3, N) = \varnothing$
$f(ER_3, A) = \{ER_3, F\}$
$f(F, DET) = \varnothing$
$f(F, N) = \varnothing$
$f(F, A) = \varnothing$

se renombran los estados:

$f(q_0, DET) = q_1$
$f(q_0, N) = \varnothing$
$f(q_0, A) = \varnothing$
$f(q_1, DET) = \varnothing$
$f(q_1, N) = q_2$
$f(q_1, A) = \varnothing$
$f(q_2, DET) = \varnothing$
$f(q_2, N) = q_2$
$f(q_2, A) = q_3$
$f(q_3, DET) = \varnothing$
$f(q_3, N) = \varnothing$
$f(q_3, A) = q_3$

y se eliminan las transiciones vacías:

$f(q_0, \text{DET}) = q_1$
$f(q_1, \text{N}) = q_2$
$f(q_2, \text{N}) = q_2$
$f(q_2, \text{A}) = q_3$
$f(q_3, \text{A}) = q_3$

El autómata obtenido se representa en un transductor gráfico que se encarga de asignar marcas a las variantes de los SSNN identificados (Fig. 6.67).

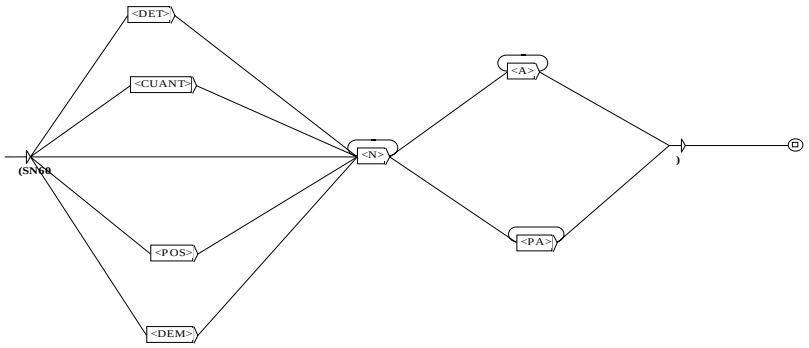


Fig. 6.67: FST gráfico que agrupa las variantes de la estructura SN_{60}

5. **SN_{61} → DET A N^+** (**Determinante Adjetivo iter. Nombre**)

ER_0 = DET A N^+
N N^+
PA N^+
CUANT A N^+
POS A N^+
DEM A N^+
DET PA N^+
CUANT PA N^+
POS PA N^+
DEM PA N^+

$$ER_0 = \text{DET CARD } N^+$$
$$\text{POS CARD } N^+$$
$$\text{DEM CARD } N^+$$
$$\text{DET ORD } N^+$$
$$\text{POS ORD } N^+$$
$$\text{DEM ORD } N^+$$
$$\text{DET ADV A } N^+$$
$$\text{CUANT ADV A } N^+$$
$$\text{POS ADV A } N^+$$
$$\text{DEM ADV A } N^+$$
$$\text{DET ADV PA } N^+$$
$$\text{CUANT ADV PA } N^+$$
$$\text{POS ADV PA } N^+$$
$$\text{DEM ADV PA } N^+$$

$$D_{DET}(ER_0) =$$
$$\left[D_{DET}(\text{DET}) \text{ A } N^+ + \alpha(\text{DET}) D_{DET}(\text{A}) N^+ \right] =$$
$$\lambda \text{ A } N^+ + \varnothing = \text{A } N^+ = ER_1$$

$$D_A(ER_0) =$$
$$\left[D_A(\text{DET}) \text{ A } N^+ + \alpha(\text{DET}) D_A(\text{A}) N^+ \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_N(ER_0) =$$
$$\left[D_N(\text{DET}) \text{ A } N^+ + \alpha(\text{DET}) D_N(\text{A}) N^+ \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_{DET}(ER_1) =$$
$$D_{DET}(A)\ N^+ =$$
$$D_{DET}(A)\ N^+ =$$
$$\varnothing \quad N^+ = \varnothing$$

$$D_A(ER_1) =$$
$$D_A(A)\ N^+ =$$
$$\lambda\ N^+ = N^+ = ER_2$$

$$D_N(ER_1) =$$
$$D_N(A)\ N^+ =$$
$$\varnothing\ N^+ = \varnothing$$

$$D_{DET}(ER_2) =$$
$$D_{DET}\left(N^+\right) =$$
$$D_{DET}\left(N\ N^*\right) =$$
$$\left[D_{DET}(N)\ N^* + \alpha(N)\ D_{DET}\left(N^*\right)\right] =$$
$$\varnothing\ N^* + \varnothing = \varnothing$$

$$D_A(ER_2) =$$
$$D_A\left(N^+\right) =$$
$$D_A\left(N\ N^*\right) =$$
$$\left[D_A(N)\ N^* + \alpha(N)\ D_A\left(N^*\right)\right] =$$
$$\varnothing\ N^* + \varnothing = \varnothing$$

$$D_N(ER_2) =$$
$$D_N\left(N^+\right) =$$
$$D_N\left(N\ N^*\right)$$
$$\left[D_N(N)\ N^* + \alpha(N)\ D_N\left(N^*\right)\right] =$$
$$\lambda\ N^* + \varnothing =$$
$$N^* = ER_3$$

$D_{DET}(ER_3) =$
$D_{DET}(N^*) =$
$D_{DET}(N)\ N^* =$
$\varnothing\ N^* = \varnothing$

$D_A(ER_3) =$
$D_A(N^*) =$
$D_A(N)\ N^* =$
$\varnothing\ N^* = \varnothing$

$D_N(ER_3) =$
$D_N(N^*) =$
$D_N(N)\ N^* =$
$\lambda\ N^* = ER_3$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_{61} es la siguiente:

$G = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= A\ ER_2$
$ER_2 ::= N\ ER_3 \mid N$
$ER_3 ::= N\ ER_3 \mid N$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$AF = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, DET) = ER_1$
$f\ (ER_0, A) = \varnothing$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_1, DET) = \varnothing$
$f\ (ER_1, A) = ER_2$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_2, DET) = \varnothing$
$f\ (ER_2, A) = \varnothing$
$f\ (ER_2, N) = \{ER_3, F\}$
$f\ (ER_3, DET) = \varnothing$
$f\ (ER_3, A) = \varnothing$
$f\ (ER_3, N) = \{ER_3, F\}$
$f\ (F, DET) = \varnothing$
$f\ (F, A) = \varnothing$
$f\ (F, N) = \varnothing$

se renombran los estados:

$f\ (q_0, DET) = q_1$
$f\ (q_0, A) = \varnothing$
$f\ (q_0, N) = \varnothing$
$f\ (q_1, DET) = \varnothing$
$f\ (q_1, A) = q_2$
$f\ (q_1, N) = \varnothing$
$f\ (q_2, DET) = \varnothing$
$f\ (q_2, A) = \varnothing$
$f\ (q_2, N) = q_3$
$f\ (q_3, DET) = \varnothing$
$f\ (q_3, A) = \varnothing$
$f\ (q_3, N) = q_3$

y se eliminan las transiciones vacías:

$f\ (q_0, DET) = q_1$
$f\ (q_1, A) = q_2$
$f\ (q_2, N) = q_3$
$f\ (q_3, N) = q_3$

El autómata obtenido se representa en un transductor gráfico que se encarga de asignar marcas a las variantes de los SSNN identificados (Fig. 6.68).

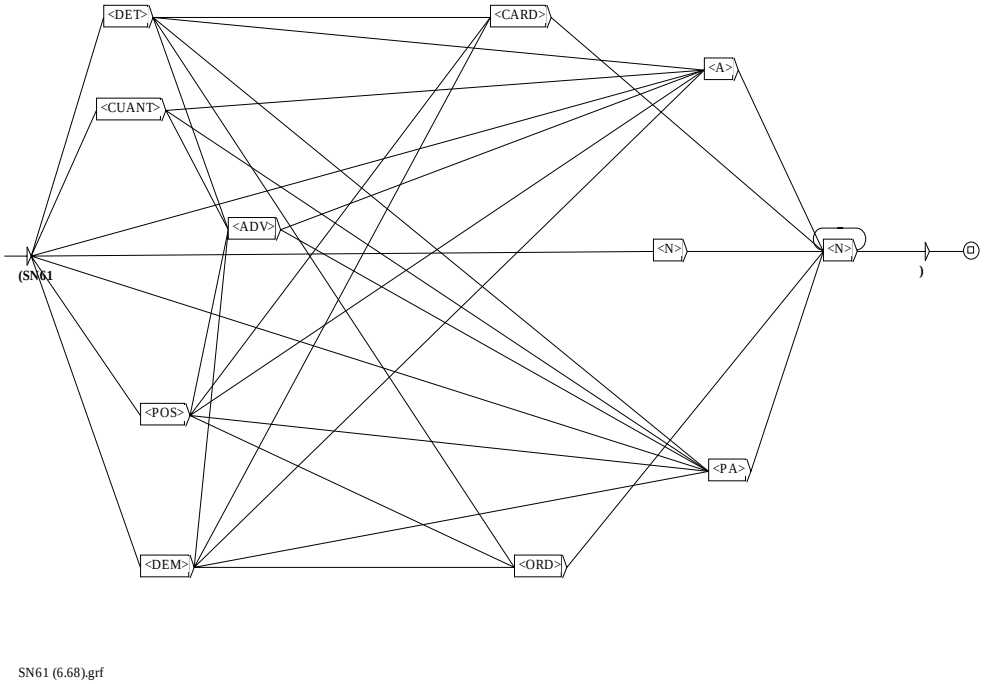


Fig. 6.68: FST gráfico que agrupa las variantes de la estructura SN_{61}

6. **SN_{62} → DET A^+ N** (**Determinante iter. Adjetivo Nombre**)

ER_0 = DET A^+ N
A^+ N
PA$^+$ N
CUANT A^+ N
POS A^+ N
DEM A^+ N
DET PA$^+$ N
CUANT PA$^+$ N
POS PA$^+$ N
DEM PA$^+$ N

$$ER_0 = \text{DET ADV } A^+ \text{ N}$$
$$\text{CUANT ADV } A^+ \text{ N}$$
$$\text{POS ADV } A^+ \text{ N}$$
$$\text{DEM ADV } A^+ \text{ N}$$
$$\text{DET ADV PA}^+ \text{ N}$$
$$\text{CUANT ADV PA}^+ \text{ N}$$
$$\text{POS ADV PA}^+ \text{ N}$$
$$\text{DEM ADV PA}^+ \text{ N}$$

$$D_{DET}(ER_0) =$$
$$\left[D_{DET}(DET)\ A^+ \text{ N} + \alpha(DET)\ D_{DET}\left(A^+\right) \text{ N} \right] =$$
$$\lambda\ A^+ \text{ N} + \varnothing = A^+ \text{ N} = ER_1$$

$$D_A(ER_0) =$$
$$\left[D_A(DET)\ A^+ \text{ N} + \alpha(DET)\ D_A\left(A^+\right) \text{ N} \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_N(ER_0) =$$
$$\left[D_N(DET)\ A^+ \text{ N} + \alpha(DET)\ D_A\left(A^+\right) \text{ N} \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_{DET}(ER_1) =$$
$$D_{DET}\left(A^+\right) \text{ N} =$$
$$D_{DET}\left(A\ A^*\right) \text{ N} =$$
$$\left[D_{DET}(A)\ A^* \text{ N} + \alpha(A)\ D_{DET}\left(A^* \text{ N}\right) \right] =$$
$$\varnothing\ A^* \text{ N} + \varnothing = \varnothing$$

$$D_A(ER_1) =$$
$$D_A\left(A^+ \text{ N}\right) =$$
$$D_A\left(A\ A^* \text{ N}\right) =$$
$$\left[D_A(A)\ A^* \text{ N} + \alpha(A)\ D_A\left(A^* \text{ N}\right) \right] =$$
$$\lambda\ A^* \text{ N} + \varnothing =$$
$$A^* \text{ N} = ER_2$$

$$D_N(ER_1) =$$
$$D_N\left(A^+ \text{ N}\right) =$$
$$D_N\left(A\ A^* \text{ N}\right) =$$
$$\left[D_N(A)\ A^* \text{ N} + \alpha(A)\ D_N\left(A^* \text{ N}\right) \right] =$$
$$\varnothing\ A^* \text{ N} + \varnothing = \varnothing$$

$D_{DET}(ER_2) =$
$D_{DET}(A^*)\ N + \alpha(A^*)\ D_{DET}(N) =$
$D_{DET}(A)\ A^*\ N + \lambda\ \varnothing =$
$\varnothing\ A^*\ N + \varnothing = \varnothing$

$D_A(ER_2) =$
$D_A(A^*)\ N + \alpha(A^*)\ D_A(N) =$
$D_A(A)\ A^*\ N + \lambda\ \varnothing =$
$\lambda\ A^*\ N = ER_2$

$D_N(ER_2) =$
$D_N(A^*)\ N + \alpha(A^*)\ D_N(N) =$
$D_N(A)\ A^*\ N + \lambda\ \lambda =$
$\varnothing\ A^*\ N + \lambda = \lambda$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_{62} es la siguiente:

$G = (\{DET, A, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= A\ ER_2$
$ER_2 ::= A\ ER_2 \mid N$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$AF = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (ER_0, DET) = ER_1$
$f\ (ER_0, A) = \varnothing$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_1, DET) = \varnothing$
$f\ (ER_1, A) = ER_2$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_2, DET) = \varnothing$
$f\ (ER_2, A) = ER_2$
$f\ (ER_2, N) = F$
$f\ (F, DET) = \varnothing$
$f\ (F, A) = \varnothing$
$f\ (F, N) = \varnothing$

se renombran los estados:

$f(q_0, \texttt{DET}) = q_1$
$f(q_0, \texttt{A}) = \varnothing$
$f(q_0, \texttt{N}) = \varnothing$
$f(q_1, \texttt{DET}) = \varnothing$
$f(q_1, \texttt{A}) = q_2$
$f(q_1, \texttt{N}) = \varnothing$
$f(q_2, \texttt{DET}) = \varnothing$
$f(q_2, \texttt{A}) = q_2$
$f(q_2, \texttt{N}) = q_3$

y se eliminan las transiciones vacías:

$f(q_0, \texttt{DET}) = q_1$
$f(q_1, \texttt{N}) = q_2$
$f(q_2, \texttt{N}) = q_2$
$f(q_2, \texttt{A}) = q_3$

A continuación, el autómata obtenido se representa en un transductor gráfico que se encarga de asignar marcas a las variantes de los SSNN identificados (Fig. 6.69).

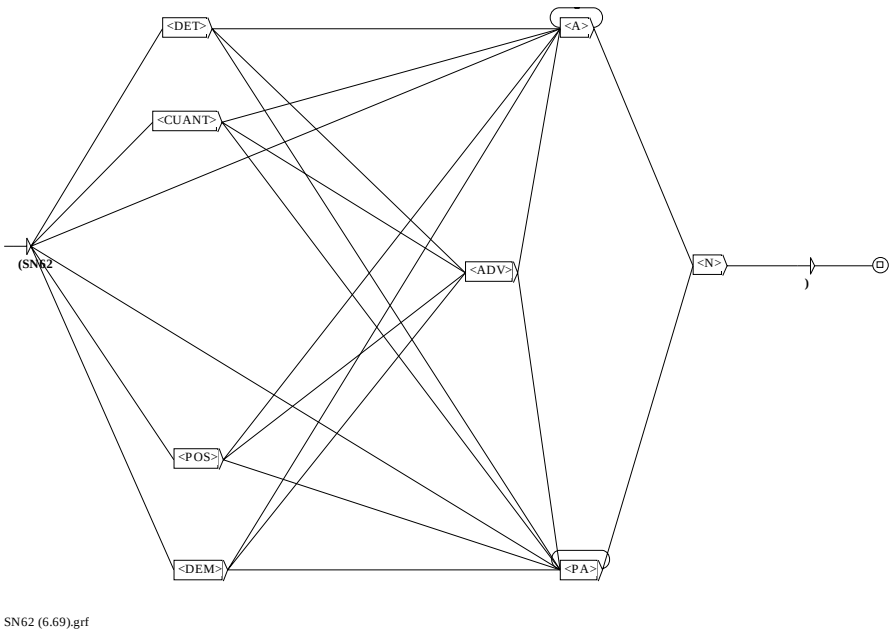


Fig. 6.69: FST gráfico que agrupa las variantes de la estructura $\texttt{SN}_{62}$

7. $\mathbf{SN_{63} \rightarrow DET\ A^{+}\ N^{+}}$ (**Determinante iter. Adjetivo iter. Nombre**)

ER_0 = DET A^{+} N^{+}
A^{+} N^{+}
PA^{+} N^{+}
CUANT A^{+} N^{+}
POS A^{+} N^{+}
DEM A^{+} N^{+}
DET PA^{+} N^{+}
CUANT PA^{+} N^{+}
POS PA^{+} N^{+}
DEM PA^{+} N^{+}

ER_0 = DET ADV A^{+} N^{+}
CUANT ADV A^{+} N^{+}
POS ADV A^{+} N^{+}
DEM ADV A^{+} N^{+}
DET ADV PA^{+} N^{+}
CUANT ADV PA^{+} N^{+}
POS ADV PA^{+} N^{+}
DEM ADV PA^{+} N^{+}

$$D_{DET}(ER_0) =$$
$$\left[D_{DET}(DET)\ A^{+}\ N^{+} + \alpha(DET)\ D_{DET}\left(A^{+}\right)\ N^{+} \right] =$$
$$\lambda\ A^{+}\ N^{+} + \varnothing = A^{+}\ N^{+} = ER_1$$

$$D_{A}(ER_0) =$$
$$\left[D_{A}(DET)\ A^{+}\ N^{+} + \alpha(DET)\ D_{A}\left(A^{+}\right)\ N^{+} \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_{N}(ER_0) =$$
$$\left[D_{N}(DET)\ A^{+}\ N^{+} + \alpha(DET)\ D_{N}\left(A^{+}\right)\ N^{+} \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_{DET}(ER_1) =$$
$$D_{DET}\left(A^+ \; N^+\right) =$$
$$D_{DET}\left(A \; A^* \; N^+\right) =$$
$$\left[D_{DET}(A) \; A^* \; N^+ + \alpha(A) \; D_{DET}\left(A^* \; N^+\right)\right] =$$
$$\left[\varnothing \; A^* \; N^+ + \varnothing\right] = \varnothing$$

$$D_A(ER_1) =$$
$$D_A\left(A^+ \; N^+\right) =$$
$$D_A\left(A \; A^* \; N^+\right) =$$
$$\left[D_A(A) \; A^* \; N^+ + \alpha(A) \; D_A\left(A^* \; N^+\right)\right] =$$
$$\left[\lambda \; A^* \; N^+ + \varnothing\right] =$$
$$A^* \; N^+ = ER_2$$

$$D_N(ER_1) =$$
$$D_N\left(A^+ \; N^+\right) =$$
$$D_N\left(A \; A^* \; N^+\right)$$
$$\left[D_N(A) \; A^* \; N^+ + \alpha(N) \; D_N\left(A^* \; N^+\right)\right] =$$
$$\left[\varnothing \; A^* \; N^+ + \varnothing\right] = \varnothing$$

$$D_{DET}(ER_2) =$$
$$D_{DET}\left(A^* \; N^+\right) \; =$$
$$\left[D_{DET}\left(A^*\right) \; N^+ + \alpha\left(A^*\right) \; D_{DET}\left(N^+\right)\right] =$$
$$\varnothing \; N^+ + \lambda \; D_{DET}\left(N^+\right) = \varnothing$$

$$D_A(ER_2) =$$
$$D_A\left(A^* \; N^+\right) =$$
$$\left[D_A\left(A^*\right) \; N^+ + \alpha\left(A^*\right) \; D_A\left(N^+\right)\right] =$$
$$D_A(A) \; A^* \; N^+ \; + \lambda \; D_A\left(N \; N^*\right) =$$
$$\lambda \; A^* \; N^+ = ER_2$$

$$D_N(ER_2) =$$
$$D_N\left(A^* \; N^+\right) =$$
$$\left[D_N\left(A^*\right) \; N^+ + \alpha\left(A^*\right) \; D_N\left(N^+\right)\right] =$$
$$\left[D_N(A) \; A^* \; N^+ + \alpha\left(A^*\right) \; D_N\left(N \; N^*\right)\right] =$$
$$\left[\varnothing \; A^* \; N^+ + \lambda \left[D_N(N) \; N^* + \alpha(N) \; D_N\left(N^*\right)\right]\right] =$$
$$\left[\lambda \left[\lambda \; N^* + \varnothing \; D_N\left(N^*\right)\right]\right] =$$
$$N^* = ER_3$$

$$D_{DET}(ER_3) =$$
$$D_{DET}(N) \; N^* = \varnothing$$

$$D_A(ER_3) =$$
$$D_A(N) \; N^* = \varnothing$$

$$D_N(ER_3) =$$
$$D_N(N) \; N^* =$$
$$\lambda \; N^* = ER_3$$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_{63} es la siguiente:

$$G = \left(\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P\right)$$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= A\ ER_2$
$ER_2 ::= A\ ER_2 \mid A\ ER_3 \mid N$
$ER_3 ::= N\ ER_3 \mid N$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$$AF = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3, F\}, f, ER_0, F)$$

donde la función de transición, f, se define como:

$f(ER_0, DET) = ER_1$
$f(ER_0, A) = \varnothing$
$f(ER_0, N) = \varnothing$
$f(ER_1, DET) = \varnothing$
$f(ER_1, A) = ER_2$
$f(ER_1, N) = \varnothing$
$f(ER_2, DET) = \varnothing$
$f(ER_2, A) = ER_2$
$f(ER_2, N) = \{ER_3, F\}$
$f(ER_3, DET) = \varnothing$
$f(ER_3, A) = \varnothing$
$f(ER_3, N) = \{ER_3, F\}$
$f(F, DET) = \varnothing$
$f(F, A) = \varnothing$
$f(F, N) = \varnothing$

se renombran los estados:

$f(q_0, DET) = q_1$
$f(q_0, A) = \varnothing$
$f(q_0, N) = \varnothing$
$f(q_1, DET) = \varnothing$
$f(q_1, A) = q_2$
$f(q_1, N) = \varnothing$
$f(q_2, DET) = \varnothing$
$f(q_2, A) = q_2$
$f(q_2, N) = q_3$
$f(q_3, DET) = \varnothing$
$f(q_3, A) = \varnothing$
$f(q_3, N) = q_3$

y se eliminan las transiciones vacías:

$f(q_0, DET) = q_1$
$f(q_1, A) = q_2$
$f(q_2, A) = q_2$
$f(q_2, N) = q_3$
$f(q_3, N) = q_3$

El autómata obtenido se representa en un transductor gráfico que se encarga de asignar marcas a las variantes de los SSNN identificados (Fig. 6.70).

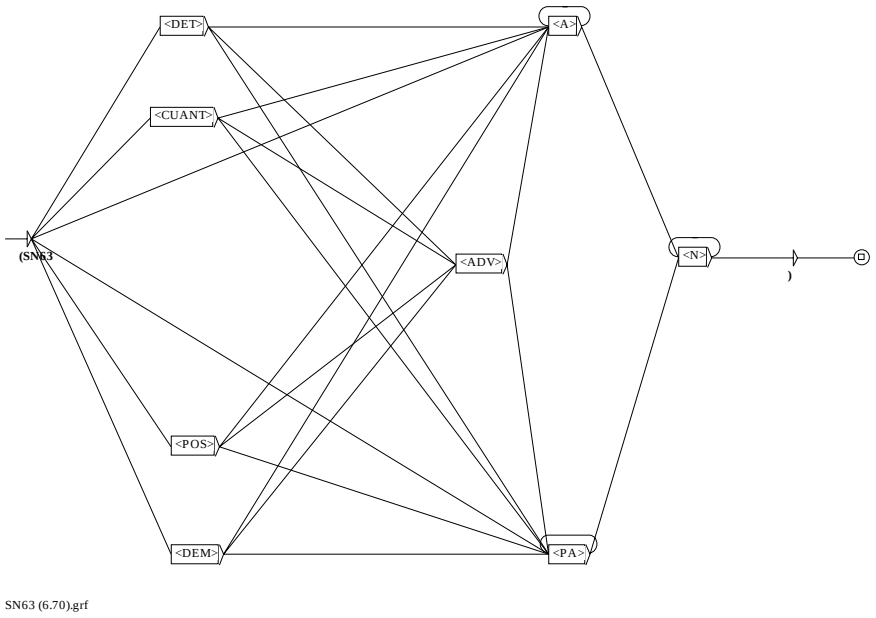


Fig. 6.70: FST gráfico que agrupa las variantes de la estructura SN_{63}

8. **SN_{64} → DET A N^+ A** (**Determinante Adjetivo iter. Nombre Adjetivo**)

ER_0 = DET A N^+ A
A N^+ A
A N^+ PA
PA N^+ A
PA N^+ PA
DET A N^+ PA
DET PA N^+ A
DET PA N^+ PA
POS A N^+ A
POS A N^+ PA
POS PA N^+ A
POS PA N^+ PA
DEM A N^+ A
DEM A N^+ PA
DEM PA N^+ A
DEM PA N^+ PA
CUANT A N^+ A
CUANT A N^+ PA
CUANT PA N^+ A
CUANT PA N^+ PA

ER_0 = DET ADV A N^+ A
DET ADV A N^+ PA
DET ADV PA N^+ A
DET ADV PA N^+ PA
POS ADV A N^+ A
POS ADV A N^+ PA
POS ADV PA N^+ A
POS ADV PA N^+ PA
DEM ADV A N^+ A
DEM ADV A N^+ PA
DEM ADV PA N^+ A
DEM ADV PA N^+ PA
CUANT ADV A N^+ A
CUANT ADV A N^+ PA
CUANT ADV PA N^+ A
CUANT ADV PA N^+ PA

$$D_{DET}(ER_0) =$$
$$\left[D_{DET}(DET)\ A\ N^+\ A + \alpha(DET)\ D_{DET}\left(A\ N^+\ A\right) \right] =$$
$$\left[\lambda\ A\ N^+\ A + \varnothing\right] = A\ N^+\ A = ER_1$$

$$D_A(ER_0) =$$
$$\left[D_A(DET)\ A\ N^+\ A + \alpha(DET)\ D_A\left(A\ N^+\ A\right) \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_N(ER_0) =$$
$$\left[D_N(DET)\ A\ N^+\ A + \alpha(DET)\ D_N\left(A\ N^+\ A\right) \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_{DET}(ER_1) =$$
$$D_{DET}\left(A\ N^+\ A\right) =$$
$$\left[D_{DET}(A)\ N^+\ A + \alpha(A)\ D_{DET}\left(N^+\ A\right)\right] =$$
$$\left[\varnothing\ \ N^+A + \varnothing\right] = \varnothing$$

$$D_A(ER_1) =$$
$$D_A\left(A\ N^+\ A\right) =$$
$$\left[D_A(A)\ N^+\ A + \alpha(A)\ D_A\left(N^+\ A\right)\right]$$
$$\left[\lambda\ N^+\ A + \varnothing\right] = N^+\ A = ER_2$$

$$D_N(ER_1) =$$
$$D_N\left(A\ N^+\ A\right) =$$
$$\left[D_N(A)\ N^+\ A + \alpha(A)\ D_N\left(N^+\ A\right)\right]$$
$$\left[\varnothing\ N^+\ A + \varnothing\right] = \varnothing$$

$$D_{DET}(ER_2) =$$
$$D_{DET}\left(N^{+}\ A\right) =$$
$$D_{DET}\left(N\ N^{*}\ A\right) =$$
$$\left[D_{DET}(N)\ N^{*}\ A + \alpha(N)\ D_{DET}\left(N^{*}\ A\right)\right] =$$
$$\left[\varnothing\ N^{*} + \varnothing\right] = \varnothing$$

$$D_{A}(ER_2) =$$
$$D_{A}\left(N^{+}\ A\right) =$$
$$D_{A}\left(N\ N^{*}\ A\right) =$$
$$\left[D_{A}(N)\ N^{*}\ A + \alpha(N)\ D_{A}\left(N^{*}\ A\right)\right] =$$
$$\left[\varnothing\ N^{*}\ A + \varnothing\right] = \varnothing$$

$$D_{N}(ER_2) =$$
$$D_{N}\left(N^{+}\ A\right) =$$
$$D_{N}\left(N\ N^{*}\ A\right)$$
$$\left[D_{N}(N)\ N^{*}\ A + \alpha(N)\ D_{N}\left(N^{*}\ A\right)\right] =$$
$$\left[\lambda\ N^{*}\ A + \varnothing\right] =$$
$$N^{*}\ A = ER_3$$

$$D_{DET}(ER_3) =$$
$$D_{DET}\left(N^{*}\ A\right) =$$
$$\left[D_{DET}\left(N^{*}\right)\ A + \alpha\left(N^{*}\right)\ D_{DET}(A)\right] =$$
$$\left[D_{DET}(N)\ N^{*}\ A + \lambda\ \varnothing\right] =$$
$$\varnothing\ N^{*}\ A = \varnothing$$

$$D_{A}(ER_3) =$$
$$D_{A}\left(N^{*}\ A\right) =$$
$$\left[D_{A}\left(N^{*}\right)\ A + \alpha\left(N^{*}\right)\ D_{A}(A)\right] =$$
$$\left[D_{A}(N)\ N^{*}\ A + \lambda\ \lambda\ \right] =$$
$$\varnothing\ N^{*}\ A + \lambda = \lambda$$

$$D_{N}(ER_3) =$$
$$\left[D_{N}\left(N^{*}\right)\ A + \alpha\left(N^{*}\right)\ D_{N}(A)\right] =$$
$$\left[D_{N}(N)\ N^{*}\ A + \lambda\ \varnothing\right] =$$
$$\lambda\ N^{*}\ A = ER_3$$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_{64} es la siguiente:

$G = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= A\ ER_2$
$ER_2 ::= N\ ER_3$
$ER_3 ::= N\ ER_3 \mid A$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$AF = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, DET) = ER_1$
$f(ER_0, A) = \varnothing$
$f(ER_0, N) = \varnothing$
$f(ER_1, DET) = \varnothing$
$f(ER_1, A) = ER_2$
$f(ER_1, N) = \varnothing$
$f(ER_2, DET) = \varnothing$
$f(ER_2, A) = \varnothing$
$f(ER_2, N) = ER_3$
$f(ER_3, DET) = \varnothing$
$f(ER_3, A) = F$
$f(ER_3, N) = ER_3$
$f(F, DET) = \varnothing$
$f(F, A) = \varnothing$
$f(F, N) = \varnothing$

se renombran los estados:

$f\ (q_0, \texttt{DET}) = q_1$
$f\ (q_0, \texttt{A}) = \varnothing$
$f\ (q_0, \texttt{N}) = \varnothing$
$f\ (q_1, \texttt{DET}) = \varnothing$
$f\ (q_1, \texttt{A}) = q_2$
$f\ (q_1, \texttt{N}) = \varnothing$
$f\ (q_2, \texttt{DET}) = \varnothing$
$f\ (q_2, \texttt{A}) = q_4$
$f\ (q_2, \texttt{N}) = q_3$
$f\ (q_3, \texttt{DET}) = \varnothing$
$f\ (q_3, \texttt{A}) = \varnothing$
$f\ (q_3, \texttt{N}) = q_3$

y se eliminan las transiciones vacías:

$f\ (q_0, \texttt{DET}) = q_1$
$f\ (q_1, \texttt{A}) = q_2$
$f\ (q_2, \texttt{N}) = q_3$
$f\ (q_3, \texttt{A}) = q_4$
$f\ (q_3, \texttt{N}) = q_3$

El autómata obtenido se representa en un transductor gráfico que se encarga de asignar marcas a las variantes de los SSNN identificados (Fig. 6.71).

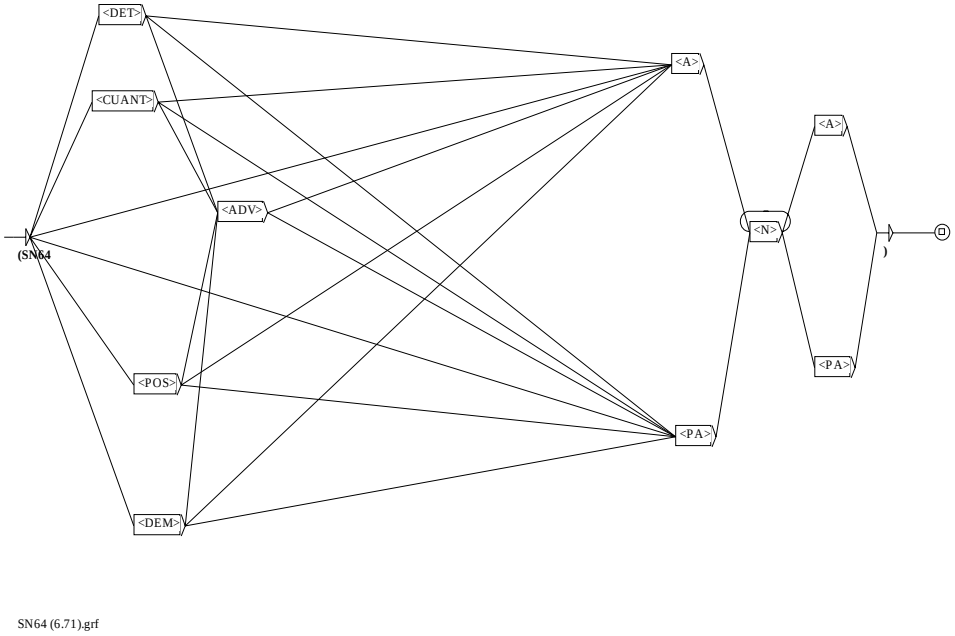


Fig. 6.71: FST gráfico que agrupa las variantes de la estructura $\texttt{SN}_{64}$

9. **$\texttt{SN}_{65}$ → DET A$^+$ N A** (**Determinante iter. Adjetivo Nombre Adjetivo**)

ER_0 = DET A^+ N A
A$^+$ N A
A$^+$ N PA
PA$^+$ N A
PA$^+$ N PA
DET A$^+$ N PA
DET PA$^+$ N A
DET PA$^+$ N PA
POS A$^+$ N A
POS A$^+$ N PA
POS PA$^+$ N A
POS PA$^+$ N PA
DEM A$^+$ N A
DEM A$^+$ N PA
DEM PA$^+$ N A
DEM PA$^+$ N PA
CUANT A$^+$ N A
CUANT A$^+$ N PA
CUANT PA$^+$ N A
CUANT PA$^+$ N PA

ER_0 = DET ADV A$^+$ N A
DET ADV A$^+$ N PA
DET ADV PA$^+$ N A
DET ADV PA$^+$ N PA
POS ADV A$^+$ N A
POS ADV A$^+$ N PA
POS ADV PA$^+$ N A
POS ADV PA$^+$ N PA
DEM ADV A$^+$ N A
DEM ADV A$^+$ N PA
DEM ADV PA$^+$ N A
DEM ADV PA$^+$ N PA
CUANT ADV A$^+$ N A
CUANT ADV A$^+$ N PA
CUANT ADV PA$^+$ N A
CUANT ADV PA$^+$ N PA

$$D_{DET}(ER_0) =$$
$$\left[D_{DET}(DET)\ A^+\ N\ A + \alpha(DET)\ D_{DET}\left(A^+\ N\ A\right)\right] =$$
$$\left[\lambda\ A^+\ N\ A + \varnothing\right] = A^+\ N\ A = ER_1$$

$$D_A(ER_0) =$$
$$\left[D_A(DET)\ A^+\ N\ A + \alpha(DET)\ D_A\left(A^+\ N\ A\right)\right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_N(ER_0) =$$
$$\left[D_N(DET)\ A^+\ N\ A + \alpha(DET)\ D_A\left(A^+\ N\ A\right)\right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_{DET}(ER_1) =$$
$$D_{DET}\left(A^+\ N\ A\right) =$$
$$D_{DET}\left(A\ A^*\ N\ A\right) =$$
$$\left[D_{DET}(A)\ A^*\ N\ A + \alpha(A)\ D_{DET}\left(A^*\ N\ A\right)\ \right] =$$
$$\left[\varnothing\ A^*\ N\ A + \varnothing\right] = \varnothing$$

$$D_A(ER_1) =$$
$$D_A\left(A^+\ N\ A\right) =$$
$$D_A\left(A\ A^*\ N\ A\right) =$$
$$\left[D_A(A)\ A^*\ N\ A + \alpha(A)\ D_A\left(A^*\ N\ A\right)\ \right] =$$
$$\left[\lambda\ A^*\ N\ A + \varnothing\right] =$$
$$A^*\ N\ A = ER_2$$

$$D_N(ER_1) =$$
$$D_N\left(A^+\ N\ A\right) =$$
$$D_N\left(A\ A^*\ N\ A\right) =$$
$$\left[D_N(A)\ A^*\ N\ \ A + \alpha(A)\ D_N\left(A^*\ N\ A\right)\ \right] =$$
$$\left[\varnothing\ A^*\ N\ A + \varnothing\right] = \varnothing$$

$$D_{DET}(ER_2) =$$
$$D_{DET}(A^* \; N \; A) =$$
$$\left[D_{DET}(A^*) \; N \; A + \alpha(A^*) \; D_{DET}(N \; A)\right] =$$
$$\left[D_{DET}(A) \; A^* \; N \; A + \lambda \; \varnothing\right] =$$
$$\left[\varnothing \; A^* \; N \; A + \varnothing\right] = \varnothing$$

$$D_A(ER_2) =$$
$$D_A(A^* \; N \; A) =$$
$$\left[D_A(A^*) \; N \; A + \alpha(A^*) \; D_A(N \; A)\right] =$$
$$\left[D_A(A) \; A^* \; N \; A + \lambda \; \varnothing\right] =$$
$$\lambda \; A^* \; N \; A = ER_2$$

$$D_N(ER_2) =$$
$$D_N(A^* \; N \; A) =$$
$$\left[D_N(A^*) \; N \; A + \alpha(A^*) \; D_N(N \; A)\right] =$$
$$\left[D_N(A) \; A^* \; N \; A + \lambda \; \left[D_N(N) \; A + \alpha(N) \; D_N(A)\right]\right] =$$
$$\left[\varnothing \; A^* \; N + \lambda \; \left[\lambda \;\; A + \varnothing\right]\right] =$$
$$\lambda \; A = A = ER_3$$

$$D_{DET}(ER_3) =$$
$$D_{DET}(A) = \varnothing$$

$$D_A(ER_3) =$$
$$D_A(A) = \lambda$$

$$D_N(ER_3) =$$
$$D_N(A) = \varnothing$$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_{65} es la siguiente:

$$G = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$$

donde las *reglas de producción*, P, se definen como:

$$ER_0 ::= DET \; ER_1$$
$$ER_1 ::= A \; ER_2$$
$$ER_2 ::= A \; ER_2$$
$$ER_2 ::= N \; ER_3$$
$$ER_3 ::= A$$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$$AF = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3, F\}, f, ER_0, F)$$

donde la función de transición, f, se define como:

$$f(ER_0, DET) = ER_1$$
$$f(ER_0, A) = \varnothing$$
$$f(ER_0, N) = \varnothing$$
$$f(ER_1, DET) = \varnothing$$
$$f(ER_1, A) = ER_2$$
$$f(ER_1, N) = \varnothing$$
$$f(ER_2, DET) = \varnothing$$
$$f(ER_2, A) = ER_2$$
$$f(ER_2, N) = ER_3$$
$$f(ER_3, DET) = \varnothing$$
$$f(ER_3, A) = F$$
$$f(ER_3, N) = \varnothing$$
$$f(F, DET) = \varnothing$$
$$f(F, A) = \varnothing$$
$$f(F, N) = \varnothing$$

se renombran los estados:

$$f(q_0, DET) = q_1$$
$$f(q_0, A) = \varnothing$$
$$f(q_0, N) = \varnothing$$
$$f(q_1, DET) = \varnothing$$
$$f(q_1, A) = q_2$$
$$f(q_1, N) = \varnothing$$
$$f(q_2, DET) = \varnothing$$
$$f(q_2, A) = q_2$$
$$f(q_2, N) = q_3$$
$$f(q_3, DET) = \varnothing$$
$$f(q_3, A) = q_4$$
$$f(q_3, N) = \varnothing$$

y se eliminan las transiciones vacías:

$$f(q_0, DET) = q_1$$
$$f(q_1, A) = q_2$$
$$f(q_2, A) = q_2$$
$$f(q_2, N) = q_3$$
$$f(q_3, A) = q_4$$

A continuación, el autómata obtenido se representa en un transductor gráfico que se encarga de asignar marcas a las variantes de los SSNN identificados (Fig. 6.72).

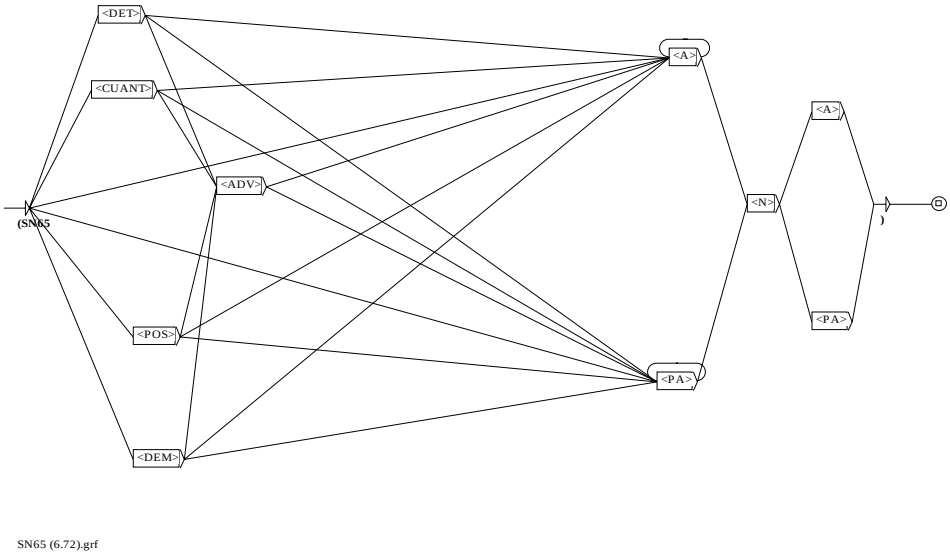


Fig. 6.72: FST gráfico que agrupa las variantes de la estructura $\texttt{SN}_{65}$

10. **$\texttt{SN}_{66}$ → DET A N $\texttt{A}^{+}$** (**Determinante Adjetivo Nombre iter. Adjetivo**)

ER_0 = DET A N A^+
A N A^+
A N PA^+
PA N A^+
PA N PA^+
DET A N PA^+
DET PA N A^+
DET PA N PA^+
POS A N A^+
POS A N PA^+
POS PA N A^+
POS PA N PA^+
DEM A N A^+
DEM A N PA^+
DEM PA N A^+
DEM PA N PA^+
CUANT A N A^+
CUANT A N PA^+
CUANT PA N A^+
CUANT PA N PA^+

ER_0 = DET ADV A N A^+
DET ADV A N PA^+
DET ADV PA N A^+
DET ADV PA N PA^+
POS ADV A N A^+
POS ADV A N PA^+
POS ADV PA N A^+
POS ADV PA N PA^+
DEM ADV A N A^+
DEM ADV A N PA^+
DEM ADV PA N A^+
DEM ADV PA N PA^+
CUANT ADV A N A^+
CUANT ADV A N PA^+
CUANT ADV PA N A^+
CUANT ADV PA N PA^+

$$D_{DET}(ER_0) =$$
$$\left[D_{DET}(DET)\ A\ N\ A^{+} + \alpha(DET)\ D_{DET}\left(A\ N\ A^{+}\right)\right] =$$
$$\left[\lambda\ A\ N\ A^{+} + \varnothing\right] = A\ N\ A^{+} = ER_1$$

$$D_A(ER_0) =$$
$$\left[D_A(DET)\ A\ N\ A^{+} + \alpha(DET)\ D_A\left(A\ N\ A^{+}\right)\right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_N(ER_0) =$$
$$\left[D_N(DET)\ A\ N\ A^{+} + \alpha(DET)\ D_A\left(A\ N\ A^{+}\right)\right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_{DET}(ER_1) =$$
$$D_{DET}\left(A\ N\ A^{+}\right) =$$
$$D_{DET}\left(A\ N\ A\ A^{*}\right) =$$
$$\left[D_{DET}(A)\ N\ A\ A^{*} + \alpha(A)\ D_{DET}\left(N\ A\ A^{*}\right)\ \right] =$$
$$\left[\varnothing\ \ N\ A\ A^{*} + \varnothing\right] = \varnothing$$

$$D_A(ER_1) =$$
$$D_A\left(A\ N\ A^{+}\right) =$$
$$D_A\left(A\ N\ A\ A^{*}\right) =$$
$$\left[D_A(A)\ N\ A\ A^{*} + \alpha(A)\ D_A\left(N\ A\ A^{*}\right)\ \right] =$$
$$\left[\lambda\ N\ A\ A^{*} + \varnothing\right] =$$
$$N\ A\ A^{*} =$$
$$N\ A^{+} = ER_2$$

$$D_N(ER_1) =$$
$$D_N\left(A\ N\ A^{+}\right) =$$
$$D_N\left(A\ N\ A\ A^{*}\right) =$$
$$\left[D_N(A)\ N\ \ A\ A^{*} + \alpha(A)\ D_N\left(N\ A\ A^{*}\right)\ \right] =$$
$$\left[\varnothing\ N\ A\ A^{*} + \varnothing\right] = \varnothing$$

$$D_{DET}(ER_2) =$$
$$D_{DET}\left(N\ A^{+}\right) =$$
$$D_{DET}\left(N\ A\ A^{*}\right) =$$
$$\left[D_{DET}(N)\ A\ A^{*} + \alpha(N)\ D_{DET}\left(A\ A^{*}\right)\right] =$$
$$\left[\varnothing\ A\ A^{*} + \varnothing\right] = \varnothing$$

$$D_{A}(ER_2) =$$
$$D_{A}\left(N\ A^{+}\right) =$$
$$D_{A}\left(N\ A\ A^{*}\right) =$$
$$\left[D_{A}(N)\ A\ A^{*} + \alpha(N)\ D_{A}\left(A\ A^{*}\right)\right] =$$
$$\left[\varnothing\ A\ A^{*} + \varnothing\right] = \varnothing$$

$$D_{N}(ER_2) =$$
$$D_{N}\left(N\ A^{+}\right) =$$
$$D_{N}\left(N\ A\ A^{*}\right) =$$
$$\left[D_{N}(N)\ A\ A^{*} + \alpha(N)\ D_{N}\left(A\ A^{*}\right)\right] =$$
$$\left[\lambda\ A\ A^{*} + \varnothing\right] =$$
$$A\ A^{*} = A^{+} = ER_3$$

$$D_{DET}(ER_3) =$$
$$D_{DET}(A^+) =$$
$$D_{DET}(A\ A^*) =$$
$$\left[D_{DET}(A)\ A^* + \alpha(A)\ D_{DET}(A^*)\right] =$$
$$\left[\varnothing\ A^* + \varnothing\right] = \varnothing$$

$$D_A(ER_3) =$$
$$D_A(A^+) =$$
$$D_A(A\ A^*) =$$
$$\left[D_A(A)\ A^* + \alpha(A)\ D_{DET}(A^*)\right] =$$
$$\left[\lambda\ A^* + \varnothing\right] =$$
$$A^* = ER_4$$

$$D_N(ER_3) =$$
$$D_N(A^+) =$$
$$D_N(A\ A^*) =$$
$$\left[D_N(A)\ A^* + \alpha(A)\ D_N(A^*)\right] =$$
$$\left[\varnothing\ A^* + \varnothing\right] = \varnothing$$

$$D_{DET}(ER_4) =$$
$$D_{DET}(A^*) =$$
$$D_{DET}(A)\ A^* =$$
$$\varnothing\ A^* = \varnothing$$

$$D_A(ER_4) =$$
$$D_A(A^*) =$$
$$D_A(A)\ A^* =$$
$$\lambda\ A^* = ER_4$$

$$D_N(ER_4) =$$
$$D_N(A^*) =$$
$$D_N(A)\ A^* =$$
$$\varnothing\ A^* = \varnothing$$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_{66} es la siguiente:

$$G = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3, ER_4\}, ER_0, P)$$

donde las *reglas de producción*, P, se definen como:

$$ER_0 ::= DET\ ER_1$$
$$ER_1 ::= A\ ER_2$$
$$ER_2 ::= N\ ER_3$$
$$ER_3 ::= A\ ER_4 \mid A$$
$$ER_4 ::= A\ ER_4 \mid A$$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$$AF = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3, ER_4, F\}, f, ER_0, F)$$

donde la función de transición, f, se define como:

$$f(ER_0, DET) = ER_1$$
$$f(ER_0, A) = \varnothing$$
$$f(ER_0, N) = \varnothing$$
$$f(ER_1, DET) = \varnothing$$
$$f(ER_1, A) = ER_2$$
$$f(ER_1, N) = \varnothing$$
$$f(ER_2, DET) = \varnothing$$
$$f(ER_2, A) = \varnothing$$
$$f(ER_2, N) = ER_3$$
$$f(ER_3, DET) = \varnothing$$
$$f(ER_3, A) = \{ER_4, F\}$$
$$f(ER_3, N) = \varnothing$$
$$f(ER_4, DET) = \varnothing$$
$$f(ER_4, A) = \{ER_4, F\}$$
$$f(ER_4, N) = \varnothing$$
$$f(F, DET) = \varnothing$$
$$f(F, A) = \varnothing$$
$$f(F, N) = \varnothing$$

se renombran los estados:

$f(q_0, \texttt{DET}) = q_1$
$f(q_0, \texttt{A}) = \varnothing$
$f(q_0, \texttt{N}) = \varnothing$
$f(q_1, \texttt{DET}) = \varnothing$
$f(q_1, \texttt{A}) = q_2$
$f(q_1, \texttt{N}) = \varnothing$
$f(q_2, \texttt{DET}) = \varnothing$
$f(q_2, \texttt{A}) = \varnothing$
$f(q_2, \texttt{N}) = q_3$
$f(q_3, \texttt{DET}) = \varnothing$
$f(q_3, \texttt{A}) = q_4$
$f(q_3, \texttt{N}) = \varnothing$
$f(q_4, \texttt{DET}) = \varnothing$
$f(q_4, \texttt{A}) = q_4$
$f(q_4, \texttt{N}) = \varnothing$

y se eliminan las transiciones vacías:

$f(q_0, \texttt{DET}) = q_1$
$f(q_1, \texttt{A}) = q_2$
$f(q_2, \texttt{N}) = q_3$
$f(q_3, \texttt{A}) = q_4$
$f(q_4, \texttt{A}) = q_4$

A continuación, el autómata obtenido se representa en un transductor gráfico que se encarga de asignar marcas a las variantes de los SSNN identificados (Fig. 6.73).

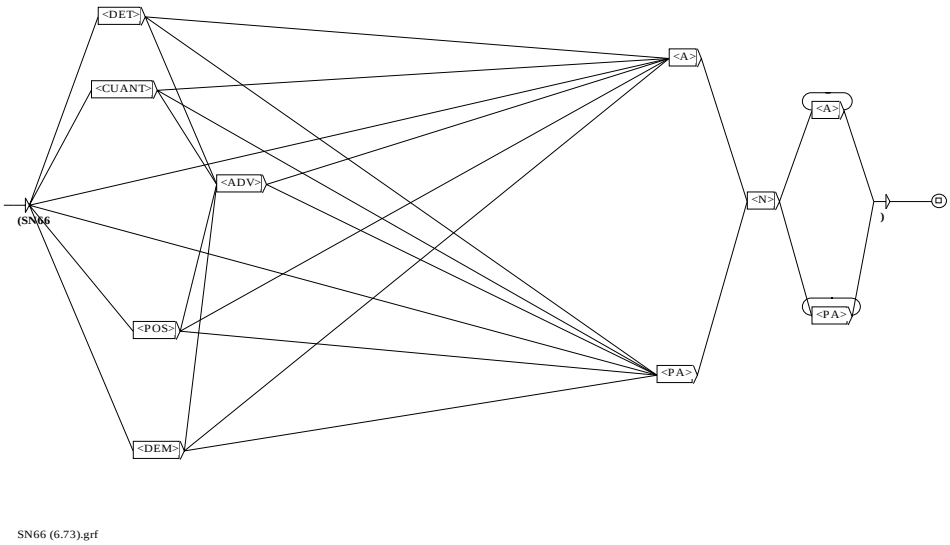


Fig. 6.73: FST gráfico que agrupa las variantes de la estructura $\texttt{SN}_{66}$

11. **SN_{67} → DET A^+ N^+ A** (**Determinante iter. Adjetivo iter. Nombre Adjetivo**)

ER_0 = DET A^+ N^+ A
A$^+$ N$^+$ A
A$^+$ N$^+$ PA
PA$^+$ N$^+$ A
PA$^+$ N$^+$ PA
DET A$^+$ N$^+$ PA
DET PA$^+$ N$^+$ A
DET PA$^+$ N$^+$ PA
POS A$^+$ N$^+$ A
POS A$^+$ N$^+$ PA
POS PA$^+$ N$^+$ A
POS PA$^+$ N$^+$ PA
DEM A$^+$ N$^+$ A
DEM A$^+$ N$^+$ PA
DEM PA$^+$ N$^+$ A
DEM PA$^+$ N$^+$ PA
CUANT A$^+$ N$^+$ A
CUANT A$^+$ N$^+$ PA
CUANT PA$^+$ N$^+$ A
CUANT PA$^+$ N$^+$ PA

ER_0 = DET ADV A^+ N^+ A
DET ADV A$^+$ N$^+$ PA
DET ADV PA$^+$ N$^+$ A
DET ADV PA$^+$ N$^+$ PA
POS ADV A$^+$ N$^+$ A
POS ADV A$^+$ N$^+$ PA
POS ADV PA$^+$ N$^+$ A
POS ADV PA$^+$ N$^+$ PA
DEM ADV A$^+$ N$^+$ A
DEM ADV A$^+$ N$^+$ PA
DEM ADV PA$^+$ N$^+$ A
DEM ADV PA$^+$ N$^+$ PA
CUANT ADV A$^+$ N$^+$ A
CUANT ADV A$^+$ N$^+$ PA
CUANT ADV PA$^+$ N$^+$ A
CUANT ADV PA$^+$ N$^+$ PA

$$D_{DET}(ER_0) =$$
$$\left[D_{DET}(DET)\ A^+\ N^+\ A + \alpha(DET)\ D_{DET}\left(A^+\ N^+\ A\right) \right] =$$
$$\left[\lambda\ A^+\ N^+\ A + \varnothing\right] =$$
$$A^+\ N^+\ A = ER_1$$

$$D_A(ER_0) =$$
$$\left[D_A(DET)\ A^+\ N^+\ A + \alpha(DET)\ D_A\left(A^+\ N^+\ A\right) \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_N(ER_0) =$$
$$\left[D_N(DET)\ A^+\ N^+\ A + \alpha(DET)\ D_N\left(A^+\ N^+\ A\right) \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_{DET}(ER_1) =$$
$$D_{DET}\left(A^+\ N^+\ A\right) =$$
$$D_{DET}\left(A\ A^*\ N\ N^*\ A\right) =$$
$$\left[D_{DET}(A)\ A^*\ N\ N^*\ A + \alpha(A)\ D_{DET}\left(A^*\ N\ N^*\ A\right)\right] =$$
$$\left[\varnothing\ A^*\ N\ N^*\ A + \varnothing\right] = \varnothing$$

$$D_A(ER_1) =$$
$$D_A\left(A^+\ N^+\ A\right) =$$
$$D_A\left(A\ A^*\ N\ N^*\ A\right) =$$
$$\left[D_A(A)\ A^*\ N\ N^*\ A + \alpha(A)\ D_A\left(A^*\ N\ N^*\ A\right)\right] =$$
$$\left[\lambda\ A^*\ N\ N^*\ A + \varnothing\right] =$$
$$A^*\ N^+\ A = ER_2$$

$$D_N(ER_1) =$$
$$D_N\left(A^+\ N^+\ A\right) =$$
$$\left[D_N(A)\ A^*\ N\ N^*\ A + \alpha(A)\ D_N\left(A^*\ N\ N^*\ A\right)\right] =$$
$$\left[\varnothing\ A^*\ N\ N^*\ A + \varnothing\right] = \varnothing$$

$$D_{DET}(ER_2) =$$
$$D_{DET}\left(A^* N^+ A\right) =$$
$$\left[D_{DET}\left(A^*\right) N^+ A + \alpha\left(A^*\right) D_{DET}\left(N^+ A\right)\right] =$$
$$\left[D_{DET}(A) A^* N^+ A + \lambda D_{DET}\left(N N^* A\right)\right] =$$
$$\varnothing + \lambda \varnothing = \varnothing$$

$$D_A(ER_2) =$$
$$D_A\left(A^* N^+ A\right) =$$
$$\left[D_A\left(A^*\right) N^+ A + \alpha\left(A^*\right) D_A\left(N^+ A\right)\right] =$$
$$\left[D_A(A) A^* N^+ A + \lambda D_A\left(N N^* A\right)\right] =$$
$$\left[\lambda A^* N^+ A + \lambda \left[D_A(N) N^* A + \alpha(N) D_A\left(N^* A\right)\right]\right] =$$
$$A^* N^+ A + \lambda \left[\varnothing N^* A + \varnothing D_A\left(N^* A\right)\right]$$
$$\lambda A^* N^+ A = ER_2$$

$$D_N(ER_2) =$$
$$D_N\left(A^* N^+ A\right) =$$
$$\left[D_N\left(A^*\right) N^+ A + \alpha\left(A^*\right) D_N\left(N^+ A\right)\right] =$$
$$\left[D_N(A) A^* N^+ A + \alpha\left(A^*\right) D_N\left(N N^* A\right)\right] =$$
$$\left[\varnothing A^* N^+ + \lambda \left[D_N(N) N^* A + \alpha(N) D_N\left(N^* A\right)\right]\right] =$$
$$\lambda \left[\lambda N^* A + \varnothing D_N\left(N^* A\right)\right] =$$
$$N^* A = ER_3$$

$D_{DET}(ER_3) =$

$D_{DET}(N^* \ A) =$

$D_{DET}(N^*) \ A + \alpha(N^*) \ D_{DET}(A) =$

$D_{DET}(N) \ N^* \ A + \lambda \ \varnothing =$

$\varnothing \ N^* + \varnothing = \varnothing$

$D_A(ER_3) =$

$D_A(N^* \ A) =$

$D_A(N^*) \ A + \alpha(N^*) \ D_A(A) =$

$D_A(N) \ N^* \ A + \lambda \ \lambda =$

$\varnothing \ N^* + \lambda = \lambda$

$D_N(ER_3) =$

$D_N(N^* \ A) =$

$D_N(N^*) \ A + \alpha(N^*) \ D_N(A) =$

$D_N(N) \ N^* \ A + \lambda \ \varnothing =$

$\lambda \ N^* \ A = ER_3$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_{67} es la siguiente:

$G = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET \ ER_1$
$ER_1 ::= A \ ER_2$
$ER_2 ::= A \ ER_2 \mid N \ ER_3$
$ER_3 ::= N \ ER_3 \mid A$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$AF = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f(ER_0, DET) = ER_1$
$f(ER_0, A) = \varnothing$
$f(ER_0, N) = \varnothing$
$f(ER_1, DET) = \varnothing$
$f(ER_1, A) = ER_2$
$f(ER_1, N) = \varnothing$
$f(ER_2, DET) = \varnothing$
$f(ER_2, A) = ER_2$
$f(ER_2, N) = ER_3$
$f(ER_3, DET) = \varnothing$
$f(ER_3, A) = F$
$f(ER_3, N) = ER_3$
$f(F, DET) = \varnothing$
$f(F, A) = \varnothing$
$f(F, N) = \varnothing$

se renombran los estados:

$f(q_0, DET) = q_1$
$f(q_0, A) = \varnothing$
$f(q_0, N) = \varnothing$
$f(q_1, DET) = \varnothing$
$f(q_1, A) = q_2$
$f(q_1, N) = \varnothing$
$f(q_2, DET) = \varnothing$
$f(q_2, A) = q_2$
$f(q_2, N) = q_3$
$f(q_3, DET) = \varnothing$
$f(q_3, A) = q_4$
$f(q_3, N) = q_3$

y se eliminan las transiciones vacías:

$f(q_0, DET) = q_1$
$f(q_1, A) = q_2$
$f(q_2, A) = q_2$
$f(q_2, N) = q_3$
$f(q_3, a) = q_4$
$f(q_3, N) = q_3$

El autómata obtenido se representa en un transductor gráfico que se encarga de asignar marcas a las variantes de los SSNN identificados (Fig. 6.74).

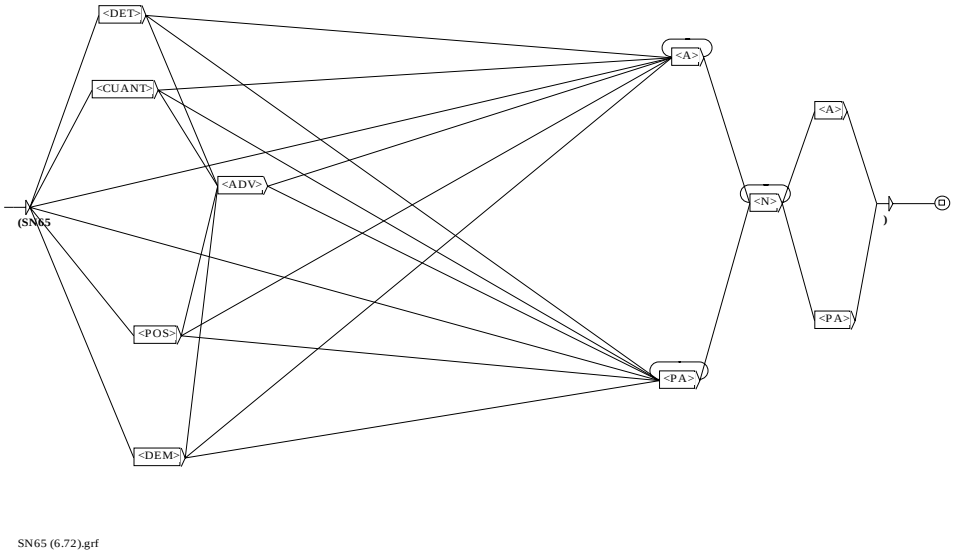


Fig. 6.74: FST gráfico que agrupa las variantes de la estructura $\texttt{SN}_{67}$

12. $\textbf{SN}_{\textbf{68}} \rightarrow \textbf{DET A N}^{+}\ \textbf{A}^{+}$ (**Determinante Adjetivo iter. Nombre iter. Adjetivo**)

ER_0 = DET A N^+ A^+
A N^+ A^+
A N^+ PA^+
PA N^+ A^+
PA N^+ PA^+
DET A N^+ PA^+
DET PA N^+ A^+
DET PA N^+ PA^+
POS A N^+ A^+
POS A N^+ PA^+
POS PA N^+ A^+
POS PA N^+ PA^+
DEM A N^+ A^+
DEM A N^+ PA^+
DEM PA N^+ A^+
DEM PA N^+ PA^+
CUANT A N^+ A^+
CUANT A N^+ PA^+
CUANT PA N^+ A^+
CUANT PA N^+ PA^+

ER_0 = DET ADV A N^+ A^+
DET ADV A N^+ PA^+
DET ADV PA N^+ A^+
DET ADV PA N^+ PA^+
POS ADV A N^+ A^+
POS ADV A N^+ PA^+
POS ADV PA N^+ A^+
POS ADV PA N^+ PA^+
DEM ADV A N^+ A^+
DEM ADV A N^+ PA^+
DEM ADV PA N^+ A^+
DEM ADV PA N^+ PA^+
CUANT ADV A N^+ A^+
CUANT ADV A N^+ PA^+
CUANT ADV PA N^+ A^+
CUANT ADV PA N^+ PA^+

$$D_{DET}(ER_0) =$$
$$\left[D_{DET}(DET)\ A\ N^+\ A^+ + \alpha(DET)\ D_{DET}\left(A\ N^+\ A^+\right)\right] =$$
$$\left[\lambda\ A\ N^+\ A^+ + \varnothing\right] = A\ N^+\ A^+ = ER_1$$

$$D_A(ER_0) =$$
$$\left[D_A(DET)\ A\ N^+\ A^+ + \alpha(DET)\ D_A\left(A\ N^+\ A^+\right)\right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_N(ER_0) =$$
$$\left[D_N(DET)\ A\ N^+\ A^+ + \alpha(DET)\ D_A\left(A\ N^+\ A^+\right)\right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_{DET}(ER_1) =$$
$$D_{DET}\left(A\ N^+\ A^+\right) =$$
$$D_{DET}\left(A\ N\ N^*\ A\ A^*\right) =$$
$$\left[D_{DET}(A)\ N\ N^*\ A\ A^* + \alpha(A)\ D_{DET}\left(N\ N^*\ A\ A^*\right)\ \right] =$$
$$\left[\varnothing\ \ N\ N^*\ A\ A^* + \varnothing\right] = \varnothing$$

$$D_A(ER_1) =$$
$$D_A\left(A\ N^+\ A^+\right) =$$
$$D_A\left(A\ N\ N^*\ A\ A^*\right) =$$
$$\left[D_A(A)\ N\ N^*\ A\ A^* + \alpha(A)\ D_A\left(N\ N^*\ A\ A^*\right)\ \right] =$$
$$\left[\lambda\ N\ N^*\ A\ A^* + \varnothing\right] =$$
$$N^+\ A^+ = ER_2$$

$$D_N(ER_1) =$$
$$D_N\left(A\ N^+\ A^+\right) =$$
$$D_N\left(A\ N\ N^*\ A\ A^*\right) =$$
$$\left[D_N(A)\ N\ N^*\ A\ A^* + \alpha(A)\ D_N\left(N\ N^*\ A\ A^*\right)\ \right] =$$
$$\left[\varnothing\ N\ N^*\ A\ A^* + \varnothing\right] = \varnothing$$

$$D_{DET}(ER_2) =$$
$$D_{DET}\left(N^+ \; A^+\right) =$$
$$D_{DET}\left(N \; N^* \; A \; A^*\right) =$$
$$\left[D_{DET}(N) \; N^* \; A \; A^* + \alpha(N) \; D_{DET}\left(N \; A \; A^*\right)\right] =$$
$$\left[\varnothing \; N^* \; A \; A^* \; + \varnothing\right] = \varnothing$$

$$D_A(ER_2) =$$
$$D_A\left(N^+ \; A^+\right) =$$
$$D_A\left(N \; N^* \; A \; A^*\right) =$$
$$\left[D_A(N) \; N^* \; A \; A^* + \alpha(N) \; D_A\left(N \; A \; A^*\right)\right] =$$
$$\left[\varnothing \; N^* \; A \; A^* \; + \varnothing\right] = \varnothing$$

$$D_N(ER_2) =$$
$$D_N\left(N^+ \; A^+\right) =$$
$$D_N\left(N \; N^* \; A \; A^*\right) =$$
$$\left[D_N(N) \; N^* \; A \; A^* + \alpha(N) \; D_N\left(A \; A^*\right)\right] =$$
$$\left[\lambda \; N^* \; A \; A^* + \varnothing\right] =$$
$$N^* \; A \; A^* =$$
$$N^* \; A^+ = ER_3$$

$$D_{DET}(ER_3) =$$
$$D_{DET}\left(N^* A^+\right) =$$
$$D_{DET}\left(N^* A A^*\right) =$$
$$\left[D_{DET}\left(N^*\right) A A^* + \propto\left(N^*\right) D_{DET}\left(A A^*\right)\right] =$$
$$\left[\varnothing A A^* + \varnothing\right] = \varnothing$$

$$D_A(ER_3) =$$
$$D_A\left(N^* A^+\right) =$$
$$D_A\left(N^* A A^*\right) =$$
$$\left[D_A\left(N^*\right) A A^* + \propto\left(N^*\right) D_A\left(A A^*\right)\right] =$$
$$\left[D_A(N) N^* A A^* + \lambda D_A(A) A^*\right] =$$
$$\left[\varnothing N^* A A^* + \lambda \lambda A^*\right] =$$
$$A^* = ER_4$$

$$D_N(ER_3) =$$
$$D_N\left(N^* A^+\right) =$$
$$D_N\left(N^* A A^*\right) =$$
$$\left[D_N\left(N^*\right) A A^* + \propto\left(N^*\right) D_N\left(A A^*\right)\right] =$$
$$\left[D_N(N) N^* A A^* + \lambda D_N\left(A A^*\right)\right] =$$
$$\left[\lambda N^* A A^* + \lambda \varnothing\right] =$$
$$N^* A^+ = ER_3$$

$D_{DET}(ER_4) =$

$D_{DET}(A^*) =$

$D_{DET}(A)\ A^* =$

$\varnothing\ A^* = \varnothing$

$D_A(ER_4) =$

$D_A(A^*) =$

$D_A(A)\ A^* =$

$\lambda\ A^* = ER_4$

$D_N(ER_4) =$

$D_N(A^*) =$

$D_N(A)\ A^* =$

$\varnothing\ A^* = \varnothing$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_{68} es la siguiente:

$G = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3, ER_4\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= DET\ ER_1$
$ER_1 ::= A\ ER_2$
$ER_2 ::= N\ ER_3$
$ER_3 ::= N\ ER_3 \mid A\ ER_4$
$ER_4 ::= A\ ER_4 \mid A$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$AF = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3, ER_4, F\}, f, ER_0, F)$

donde la función de transición, f, se define como:

$f\ (\mathrm{ER}_0, \mathrm{DET}) = \mathrm{ER}_1$
$f\ (\mathrm{ER}_0, \mathrm{A}) = \varnothing$
$f\ (\mathrm{ER}_0, \mathrm{N}) = \varnothing$
$f\ (\mathrm{ER}_1, \mathrm{DET}) = \varnothing$
$f\ (\mathrm{ER}_1, \mathrm{A}) = \mathrm{ER}_2$
$f\ (\mathrm{ER}_1, \mathrm{N}) = \varnothing$
$f\ (\mathrm{ER}_2, \mathrm{DET}) = \varnothing$
$f\ (\mathrm{ER}_2, \mathrm{A}) = \varnothing$
$f\ (\mathrm{ER}_2, \mathrm{N}) = \mathrm{ER}_3$
$f\ (\mathrm{ER}_3, \mathrm{DET}) = \varnothing$
$f\ (\mathrm{ER}_3, \mathrm{A}) = \mathrm{ER}_4$
$f\ (\mathrm{ER}_3, \mathrm{N}) = \mathrm{ER}_3$
$f\ (\mathrm{ER}_4, \mathrm{DET}) = \varnothing$
$f\ (\mathrm{ER}_4, \mathrm{A}) = \{\mathrm{ER}_4, F\}$
$f\ (\mathrm{ER}_4, \mathrm{N}) = \varnothing$
$f\ (F, \mathrm{DET}) = \varnothing$
$f\ (F, \mathrm{A}) = \varnothing$
$f\ (F, \mathrm{N}) = \varnothing$

se renombran los estados:

$f\ (q_0, \mathrm{DET}) = q_1$
$f\ (q_0, \mathrm{A}) = \varnothing$
$f\ (q_0, \mathrm{N}) = \varnothing$
$f\ (q_1, \mathrm{DET}) = \varnothing$
$f\ (q_1, \mathrm{A}) = q_2$
$f\ (q_1, \mathrm{N}) = \varnothing$
$f\ (q_2, \mathrm{DET}) = \varnothing$
$f\ (q_2, \mathrm{A}) = \varnothing$
$f\ (q_2, \mathrm{N}) = q_3$
$f\ (q_3, \mathrm{DET}) = \varnothing$
$f\ (q_3, \mathrm{A}) = q_4$
$f\ (q_3, \mathrm{N}) = q_3$
$f\ (q_4, \mathrm{DET}) = \varnothing$
$f\ (q_4, \mathrm{A}) = q_4$
$f\ (q_4, \mathrm{N}) = \varnothing$

y se eliminan las transiciones vacías:

$f\ (q_0, \mathrm{DET}) = q_1$
$f\ (q_1, \mathrm{A}) = q_2$
$f\ (q_2, \mathrm{N}) = q_3$
$f\ (q_3, \mathrm{A}) = q_4$
$f\ (q_3, \mathrm{N}) = q_3$
$f\ (q_4, \mathrm{A}) = q_4$

A continuación, el autómata obtenido se representa en un transductor gráfico que se encarga de asignar marcas a las variantes de los SSNN identificados (Fig. 6.75).

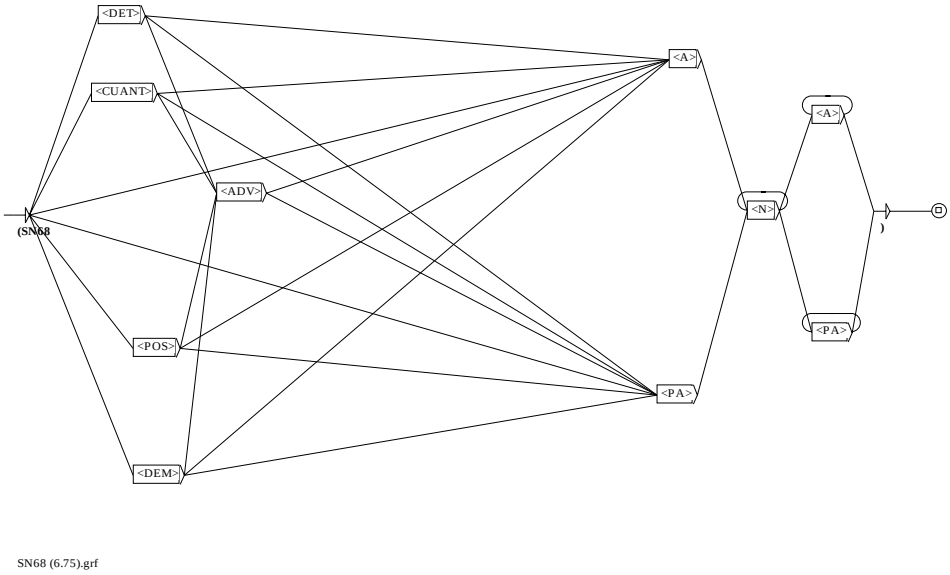


Fig. 6.75: FST gráfico que agrupa las variantes de la estructura $\mathtt{SN}_{68}$

13. **$\mathbf{SN_{69}}$ → `DET A`$^+$ `N A`$^+$** (**`Determinante iter. Adjetivo Nombre iter. Adjetivo`**)

ER_0 = DET A^+ N A^+
A^+ N A^+
A^+ N PA^+
PA^+ N A^+
PA^+ N PA^+
DET A^+ N PA^+
DET PA^+ N A^+
DET PA^+ N PA^+
POS A^+ N A^+
POS A^+ N PA^+
POS PA^+ N A^+
POS PA^+ N PA^+
DEM A^+ N A^+
DEM A^+ N PA^+
DEM PA^+ N A^+
DEM PA^+ N PA^+
CUANT A^+ N A^+
CUANT A^+ N PA^+
CUANT PA^+ N A^+
CUANT PA^+ N PA^+

ER_0 = DET ADV A^+ N A^+
DET ADV A^+ N PA^+
DET ADV PA^+ N A^+
DET ADV PA^+ N PA^+
POS ADV A^+ N A^+
POS ADV A^+ N PA^+
POS ADV PA^+ N A^+
POS ADV PA^+ N PA^+
DEM ADV A^+ N A^+
DEM ADV A^+ N PA^+
DEM ADV PA^+ N A^+
DEM ADV PA^+ N PA^+
CUANT ADV A^+ N A^+
CUANT ADV A^+ N PA^+
CUANT ADV PA^+ N A^+
CUANT ADV PA^+ N PA^+

$$D_{DET}(ER_0) =$$
$$\left[D_{DET}(DET)\ A^+\ N\ A^+ + \alpha(DET)\ D_{DET}\left(A^+\ N\ A^+\right)\right] =$$
$$\left[\lambda\ A^+\ N\ A^+ + \varnothing\right] = A^+\ N\ A^+ = ER_1$$

$$D_A(ER_0) =$$
$$\left[D_A(DET)\ A^+\ N\ A^+ + \alpha(DET)\ D_A\left(A^+\ N\ A^+\right)\right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_N(ER_0) =$$
$$\left[D_N(DET)\ A^+\ N\ A^+ + \alpha(DET)\ D_A\left(A^+\ N\ A^+\right)\right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_{DET}(ER_1) =$$
$$D_{DET}\left(A^+\ N\ A^+\right) =$$
$$D_{DET}\left(A\ A^*\ N\ A\ A^*\right) =$$
$$\left[D_{DET}(A)\ A^*\ N\ A\ A^* + \alpha(A)\ D_{DET}\left(A^*\ N\ A\ A^*\right)\ \right] =$$
$$\left[\varnothing\ A^*\ N\ A\ A^* + \varnothing\right] = \varnothing$$

$$D_A(ER_1) =$$
$$D_A\left(A^+\ N\ A^+\right) =$$
$$D_A\left(A\ A^*\ N\ A\ A^*\right) =$$
$$\left[D_A(A)\ A^*\ N\ A\ A^* + \alpha(A)\ D_A\left(A^*\ N\ A\ A^*\right)\ \right] =$$
$$\left[\lambda\ A^*\ N\ A\ A^* + \varnothing\right] =$$
$$A^*\ N\ A\ A^* =$$
$$A^*\ N\ A^+ = ER_2$$

$$D_N(ER_1) =$$
$$D_N\left(A^+\ N\ A^+\right) =$$
$$D_N\left(A\ A^*\ N\ A\ A^*\right) =$$
$$\left[D_N(A)\ A^*\ N\ \ A\ A^* + \alpha(A)\ D_N\left(A^*\ N\ A\ A^*\right)\ \right] =$$
$$\left[\varnothing\ A^*\ N\ A\ A^* + \varnothing\right] = \varnothing$$

$$D_{DET}(ER_2) =$$
$$D_{DET}\left(A^* \; N \; A^+\right) =$$
$$D_{DET}\left(A^* \; N \; A \; A^*\right) =$$
$$\left[D_{DET}\left(A^*\right) \; N \; A \; A^* + \alpha\left(A^*\right) \; D_{DET}\left(N \; A \; A^*\right)\right] =$$
$$\left[D_{DET}(A) \; A^* \; N \; A \; A^* + \lambda \; \varnothing\right] =$$
$$\left[\varnothing \; A^* \; N \; A \; A^* + \varnothing\right] = \varnothing$$

$$D_A(ER_2) =$$
$$D_A\left(A^* \; N \; A^+\right) =$$
$$D_A\left(A^* \; N \; A \; A^*\right) =$$
$$\left[D_A\left(A^*\right) \; N \; A \; A^* + \alpha\left(A^*\right) \; D_A\left(N \; A \; A^*\right)\right] =$$
$$\left[D_A(A) \; A^* \; N \; A \; A^* + \lambda \; \varnothing\right] =$$
$$\lambda \; A^* \; N \; A \; A^* =$$
$$A^* \; N \; A^+ = ER_2$$

$$D_N(ER_2) =$$
$$D_N\left(A^* \; N \; A^+\right) =$$
$$D_N\left(A^* \; N \; A \; A^*\right) =$$
$$\left[D_N\left(A^*\right) \; N \; A \; A^* + \alpha\left(A^*\right) \; D_N\left(N \; A \; A^*\right)\right] =$$
$$\left[D_N(A) \; A^* \; N \; A \; A^* + \lambda \left[D_N(N) \; A \; A^* + \alpha(N) \; D_N\left(A \; A^*\right)\right]\right] =$$
$$\left[\varnothing \; A^* \; N + \lambda \left[\lambda \;\; A \; A^* + \varnothing\right]\right] =$$
$$\lambda \; A \; A^* = A^+ = ER_3$$

$D_{DET}(ER_3) =$
$D_{DET}(A^+) =$
$D_{DET}(A\ A^*) =$
$\left[D_{DET}(A)\ A^* + \alpha(A)\ D_{DET}(A^*)\right] =$
$\varnothing\ A^* + \varnothing = \varnothing$

$D_A(ER_3) =$
$D_A(A^+) =$
$D_A(A\ A^*) =$
$\left[D_A(A)\ A^* + \alpha(A)\ D_A(A^*)\right] =$
$\lambda\ A^* + \varnothing =$
$A^* = ER_4$

$D_N(ER_3) =$
$D_N(A^+) =$
$D_N(A\ A^*) =$
$\left[D_N(A)\ A^* + \alpha(A)\ D_N(A^*)\right] =$
$\varnothing\ A^* + \varnothing = \varnothing$

$D_{DET}(ER_4) =$
$D_{DET}(A^*) =$
$D_{DET}(A)\ A^* =$
$\varnothing\ A^* = \varnothing$

$D_A(ER_3) =$
$D_A(A^*) =$
$D_A(A)\ A^* =$
$\lambda\ A^* = ER_4$

$D_N(ER_3) =$
$D_N(A^*) =$
$D_N(A)\ A^* =$
$\varnothing\ A^* = \varnothing$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_{69} es la siguiente:

$G = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3, ER_4\}, ER_0, P)$

donde las *reglas de producción*, P, se definen como:

$ER_0 ::= \texttt{DET}\ ER_1$
$ER_1 ::= \texttt{A}\ ER_2$
$ER_2 ::= \texttt{A}\ ER_2 \mid \texttt{N}\ ER_3$
$ER_3 ::= \texttt{A}\ ER_4$
$ER_4 ::= \texttt{A}\ ER_4 \mid \texttt{A}$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$$AF = (\{\texttt{DET}, \texttt{A}, \texttt{N}\}, \{ER_0, ER_1, ER_2, ER_3, ER_4, F\}, f, ER_0, F)$$

donde la función de transición, f, se define como:

$f(ER_0, \texttt{DET}) = ER_1$
$f(ER_0, \texttt{A}) = \varnothing$
$f(ER_0, \texttt{N}) = \varnothing$
$f(ER_1, \texttt{DET}) = \varnothing$
$f(ER_1, \texttt{A}) = ER_2$
$f(ER_1, \texttt{N}) = \varnothing$
$f(ER_2, \texttt{DET}) = \varnothing$
$f(ER_2, \texttt{A}) = ER_2$
$f(ER_2, \texttt{N}) = ER_3$
$f(ER_3, \texttt{DET}) = \varnothing$
$f(ER_3, \texttt{A}) = ER_4$
$f(ER_3, \texttt{N}) = \varnothing$
$f(ER_4, \texttt{DET}) = \varnothing$
$f(ER_4, \texttt{A}) = \{ER_4, F\}$
$f(ER_4, \texttt{N}) = \varnothing$
$f(F, \texttt{DET}) = \varnothing$
$f(F, \texttt{A}) = \varnothing$
$f(F, \texttt{N}) = \varnothing$

se renombran los estados:

$f(q_0, \texttt{DET}) = q_1$
$f(q_0, \texttt{A}) = \varnothing$
$f(q_0, \texttt{N}) = \varnothing$
$f(q_1, \texttt{DET}) = \varnothing$
$f(q_1, \texttt{A}) = q_2$
$f(q_1, \texttt{N}) = \varnothing$
$f(q_2, \texttt{DET}) = \varnothing$
$f(q_2, \texttt{A}) = q_2$
$f(q_2, \texttt{N}) = q_3$
$f(q_3, \texttt{DET}) = \varnothing$
$f(q_3, \texttt{A}) = q_4$
$f(q_3, \texttt{N}) = \varnothing$
$f(q_4, \texttt{DET}) = \varnothing$
$f(q_4, \texttt{A}) = q_4$
$f(q_4, \texttt{N}) = \varnothing$

y se eliminan las transiciones vacías:

$f\ (q_0, \texttt{DET}) = q_1$
$f\ (q_1, \texttt{A}) = q_2$
$f\ (q_2, \texttt{A}) = q_2$
$f\ (q_2, \texttt{N}) = q_3$
$f\ (q_3, \texttt{A}) = q_4$
$f\ (q_4, \texttt{A}) = q_4$

A continuación, el autómata obtenido se representa en un transductor gráfico que se encarga de asignar marcas a las variantes de los SSNN identificados (Fig. 6.76).

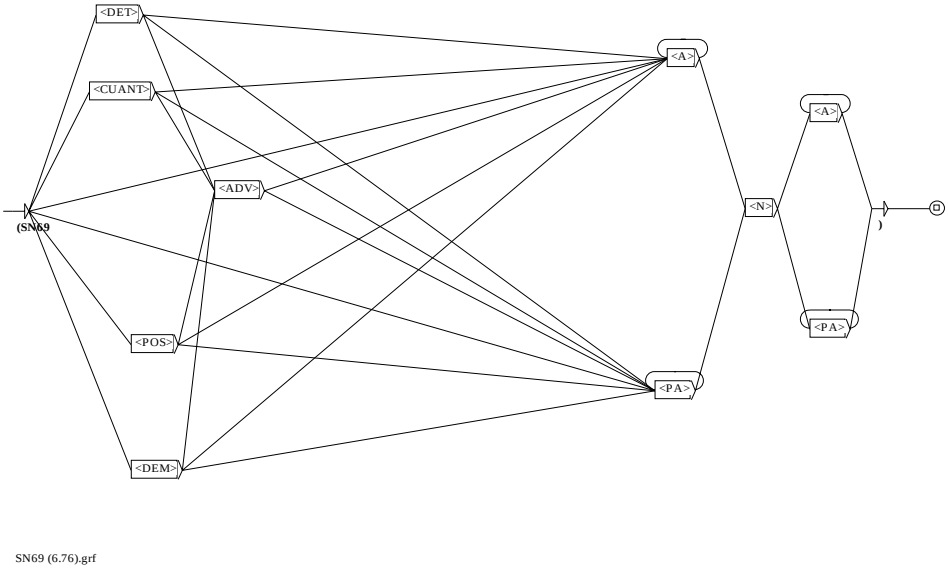


Fig. 6.76: FST gráfico que agrupa las variantes de la estructura $\texttt{SN}_{69}$

14. $\mathbf{SN_{70}} \rightarrow \mathbf{DET\ A^{+}\ N^{+}\ A^{+}}$ (**Determinante iter. Adjetivo iter. Nombre iter. Adjetivo**)

ER_0 = DET A^+ N^+ A^+
| A+ N+ A+
| A+ N+ PA+
| PA+ N+ A+
| PA+ N+ PA+
| DET A+ N+ PA+
| DET PA+ N+ A+
| DET PA+ N+ PA+
| POS A+ N+ A+
| POS A+ N+ PA+
| POS PA+ N+ A+
| POS PA+ N+ PA+
| DEM A+ N+ A+
| DEM A+ N+ PA+
| DEM PA+ N+ A+
| DEM PA+ N+ PA+
| CUANT A+ N+ A+
| CUANT A+ N+ PA+
| CUANT PA+ N+ A+
| CUANT PA+ N+ PA+

ER_0 = DET ADV A^+ N^+ A^+
| DET ADV A+ N+ PA+
| DET ADV PA+ N+ A+
| DET ADV PA+ N+ PA+
| POS ADV A+ N+ A+
| POS ADV A+ N+ PA+
| POS ADV PA+ N+ A+
| POS ADV PA+ N+ PA+
| DEM ADV A+ N+ A+
| DEM ADV A+ N+ PA+
| DEM ADV PA+ N+ A+
| DEM ADV PA+ N+ PA+
| CUANT ADV A+ N+ A+
| CUANT ADV A+ N+ PA+
| CUANT ADV PA+ N+ A+
| CUANT ADV PA+ N+ PA+

$$D_{DET}(ER_0) =$$
$$\left[D_{DET}(DET)\ A^+\ N^+\ A^+ + \alpha(DET)\ D_{DET}\left(A^+\ N^+\ A^+\right) \right] =$$
$$\left[\lambda\ A^+\ N^+\ A^+ + \varnothing \right] =$$
$$A^+\ N^+\ A^+ = ER_1$$

$$D_A(ER_0) =$$
$$\left[D_A(DET)\ A^+\ N^+\ A^+ + \alpha(DET)\ D_A\left(A^+\ N^+\ A^+\right) \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_N(ER_0) =$$
$$\left[D_N(DET)\ A^+\ N^+\ A^+ + \alpha(DET)\ D_A\left(A^+\ N^+\ A^+\right) \right] =$$
$$\varnothing + \varnothing = \varnothing$$

$$D_{DET}(ER_1) =$$
$$D_{DET}\left(A^+\ N^+\ A^+\right) =$$
$$D_{DET}\left(A\ A^*\ N\ N^*\ A\ A^*\right) =$$
$$\left[D_{DET}(A)\ A^*\ N\ N^*\ A\ A^* + \alpha(A)\ D_{DET}\left(A^*\ N\ N^*\ A\ A^*\right) \right] =$$
$$\left[\varnothing\ A^*\ N\ N^*\ A\ A^* + \varnothing \right] = \varnothing$$

$$D_A(ER_1) =$$
$$D_A\left(A^+\ N^+\ A^+\right) =$$
$$D_A\left(A\ A^*\ N\ N^*\ A\ A^*\right) =$$
$$\left[D_A(A)\ A^*\ N\ N^*\ A\ A^* + \alpha(A)\ D_A\left(A^*\ N\ N^*\ A\ A^*\right) \right] =$$
$$\left[\lambda\ A^*\ N\ N^*\ A\ A^* + \varnothing \right] =$$
$$A^*\ N\ N^*\ A\ A^* =$$
$$A^*\ N^+\ A^+ = ER_2$$

$$D_N(ER_1) =$$
$$D_N\left(A^+\ N^+\ A^+\right) =$$
$$D_N\left(A\ A^*\ N\ N^*\ A\ A^*\right) =$$
$$\left[D_N(A)\ A^*\ N\ N^*\ A\ A^* + \alpha(A)\ D_N\left(A^*\ N\ N^*\ A\ A^*\right) \right] =$$
$$\left[\varnothing\ A^*\ N\ N^*\ A\ A^* + \varnothing \right] = \varnothing$$

$$D_{DET}(ER_2) =$$
$$D_{DET}\left(A^* N^+ A^+\right) =$$
$$D_{DET}\left(A^* N N^* A A^*\right) =$$
$$\left[D_{DET}\left(A^*\right) N N^* A A^* + \alpha\left(A^*\right) D_{DET}\left(N N^* A A^*\right)\right] =$$
$$\left[D_{DET}(A) A^* N N^* A A^* + \lambda \varnothing\right] =$$
$$\left[\varnothing A^* N N^* A A^* + \varnothing\right] = \varnothing$$

$$D_A(ER_2) =$$
$$D_A\left(A^* N^+ A^+\right) =$$
$$D_A\left(A^* N N^* A A^*\right) =$$
$$\left[D_A\left(A^*\right) N N^* A A^* + \alpha\left(A^*\right) D_A\left(N N^* A A^*\right)\right] =$$
$$\left[D_A(A) A^* N N^* A A^* + \lambda \varnothing\right] =$$
$$\left[\lambda A^* N N^* A A^*\right] =$$
$$A^* N^+ A^+ = ER_2$$

$$D_N(ER_2) =$$
$$D_N\left(A^* N^+ A^+\right) =$$
$$D_N\left(A^* N N^* A A^*\right) =$$
$$\left[D_N\left(A^*\right) N N^* A A^* + \alpha\left(A^*\right) D_N\left(N N^* A A^*\right)\right] =$$
$$\left[D_N(A) A^* N N^* A A^* + \lambda \left[D_N(N) N^* A A^* + \alpha(N) D_N\left(N^* A A^*\right)\right]\right] =$$
$$\left[\varnothing A^* N N^* A A^* + \lambda \left[\lambda N^* A A^* + \varnothing\right]\right] =$$
$$\varnothing + \lambda \left[N^* A A^*\right] =$$
$$N^* A^+ = ER_3$$

$$D_{DET}(ER_3) =$$
$$D_{DET}\left(N^* A^+\right) =$$
$$D_{DET}\left(N^* A A^*\right) =$$
$$\left[D_{DET}\left(N^*\right) A A^* + \alpha\left(N^*\right) D_{DET}\left(A A^*\right)\right] =$$
$$\varnothing A A^* + \lambda \varnothing = \varnothing$$

$$D_A(ER_3) =$$
$$D_A\left(N^* A^+\right) =$$
$$D_A\left(N^* A A^*\right) =$$
$$\left[D_A\left(N^*\right) A A^* + \alpha\left(N^*\right) D_A\left(A A^*\right)\right] =$$
$$\varnothing A A^* + \lambda \left[D_A(A) A^* + \alpha(A) D_A\left(A^*\right)\right] =$$
$$\left[\lambda A^* + \varnothing\right] = A^* = ER_4$$

$$D_N(ER_3) =$$
$$D_N\left(N^* A^+\right) =$$
$$D_N\left(N^* A A^*\right) =$$
$$\left[D_N\left(N^*\right) A A^* + \alpha\left(N^*\right) D_N\left(A A^*\right)\right] =$$
$$\left[D_N(N) N^* A A^* + \lambda D_N\left(A A^*\right)\right] =$$
$$\left[\lambda N^* A A^* + \lambda \varnothing\right] =$$
$$N^* A A^* =$$
$$N^* A^+ = ER_3$$

$$D_{DET}(ER_4) =$$
$$D_{DET}\left(A^*\right) =$$
$$D_{DET}(A) A^* =$$
$$\varnothing A^* = \varnothing$$

$$D_A(ER_3) =$$
$$D_A\left(A^*\right) =$$
$$D_A(A) A^* =$$
$$\lambda A^* = ER_4$$

$$D_N(ER_3) =$$
$$D_N\left(A^*\right) =$$
$$D_N(A) A^* =$$
$$\varnothing A^* = \varnothing$$

La *Gramática Regular Lineal por la Derecha* que reconoce el sintagma SN_{70} es la siguiente:

$$G = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3, ER_4\}, ER_0, P)$$

donde las *reglas de producción*, P, se definen como:

$$ER_0 ::= DET\ ER_1$$
$$ER_1 ::= A\ ER_2$$
$$ER_2 ::= A\ ER_2 \mid N\ ER_3$$
$$ER_3 ::= A\ ER_4 \mid N\ ER_3$$
$$ER_4 ::= A\ ER_4 \mid A$$

A partir de la *Gramática Regular* se obtiene el *Autómata Finito* definido como:

$$AF = (\{DET, A, N\}, \{ER_0, ER_1, ER_2, ER_3, ER_4, F\}, f, ER_0, F)$$

donde la función de transición, f, se define como:

$$f(ER_0, DET) = ER_1$$
$$f(ER_0, A) = \varnothing$$
$$f(ER_0, N) = \varnothing$$
$$f(ER_1, DET) = \varnothing$$
$$f(ER_1, A) = ER_2$$
$$f(ER_1, N) = \varnothing$$
$$f(ER_2, DET) = \varnothing$$
$$f(ER_2, A) = ER_2$$
$$f(ER_2, N) = ER_3$$
$$f(ER_3, DET) = \varnothing$$
$$f(ER_3, A) = ER_4$$
$$f(ER_3, N) = ER_3$$
$$f(ER_4, DET) = \varnothing$$
$$f(ER_4, A) = \{ER_4, F\}$$
$$f(ER_4, N) = \varnothing$$
$$f(F, DET) = \varnothing$$
$$f(F, A) = \varnothing$$
$$f(F, N) = \varnothing$$

se renombran los estados:

$f\ (q_0, \texttt{DET}) = q_1$
$f\ (q_0, \texttt{A}) = \varnothing$
$f\ (q_0, \texttt{N}) = \varnothing$
$f\ (q_1, \texttt{DET}) = \varnothing$
$f\ (q_1, \texttt{A}) = q_2$
$f\ (q_1, \texttt{N}) = \varnothing$
$f\ (q_2, \texttt{DET}) = \varnothing$
$f\ (q_2, \texttt{A}) = q_2$
$f\ (q_2, \texttt{N}) = q_3$
$f\ (q_3, \texttt{DET}) = \varnothing$
$f\ (q_3, \texttt{A}) = q_4$
$f\ (q_3, \texttt{N}) = q_3$
$f\ (q_4, \texttt{DET}) = \varnothing$
$f\ (q_4, \texttt{A}) = q_4$
$f\ (q_4, \texttt{N}) = \varnothing$

y se eliminan las transiciones vacías:

$f\ (q_0, \texttt{DET}) = q_1$
$f\ (q_1, \texttt{A}) = q_2$
$f\ (q_2, \texttt{A}) = q_2$
$f\ (q_2, \texttt{N}) = q_3$
$f\ (q_3, \texttt{N}) = q_3$
$f\ (q_3, \texttt{A}) = q_4$
$f\ (q_4, \texttt{A}) = q_4$

A continuación, el autómata obtenido se representa en un transductor gráfico que se encarga de asignar marcas a las variantes de los SSNN identificados (Fig. 6.77).

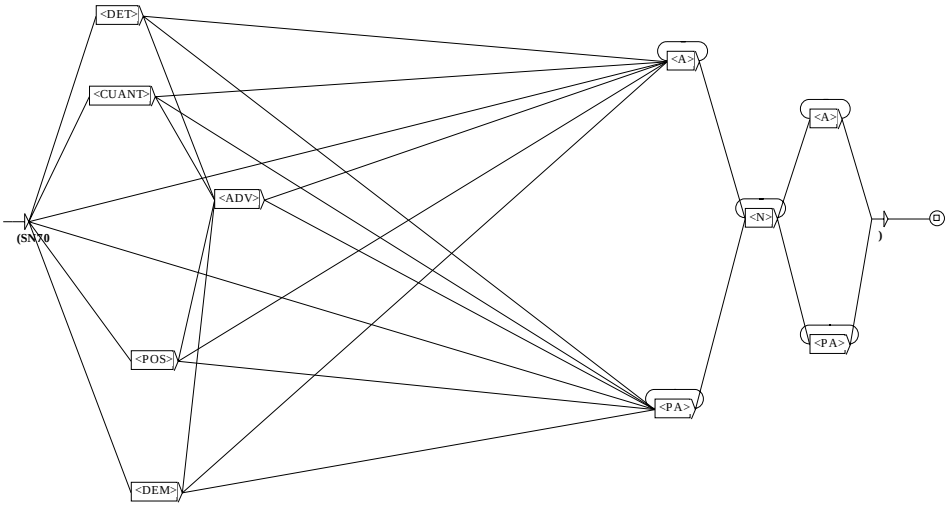


Fig. 6.77: FST gráfico que agrupa las variantes de la estructura SN_{70}

6.2. Construcción de Transductores Sintácticos

La representación de los SSNN que están modificados por *Sintagmas Preposicionales* (SSPP) u *Oraciones* (O) forman los denominados SSNN de estructura compleja. En los sintagmas con estructura compleja, se considera que un SN *domina* a un Sintagma Preposicional, o a una Oración, de tal forma que el sintagma dominante se puede definir como *Sintagma Principal*, frente a los sintagmas dominados que se definen como *Sintagmas u Oraciones Constituyentes*. La función de los SSPP y de las *Oraciones* es modificar el *núcleo* del SN. Partiendo de la consideración de que los modificadores son siempre ***complementos adjuntos***, o no reclamados por el núcleo nominal, los SSNN con estructura compleja, en posición *posnominal*, se pueden clasificar según dos *hipótesis explicativas*:

- Sintagmas compuestos por un **`Nombre`**, *núcleo*, acompañado de un, **`Sintagma Preposicional`**, *modificador*:

  `SN →  N SP  (`**Nombre** `Sintagma Preposicional)`

- Sintagmas compuestos por un **`Nombre`**, *núcleo*, acompañado de un **`Oración`**, *modificador*:

  `SN →  N O  (`**Nombre** `Oración)`

Si utilizáramos Gramáticas Regulares para efectuar el análisis sintáctico de los SSNN que incluyen otros sintagmas sólo obtendríamos una representación lineal de la estructura de constituyentes. Pero en las secuencias lineales no se puede distinguir cuál es el componente sintagmático que modifica al SN considerado el *núcleo nominal*. Para poder reconocer la estructura de constituyentes sería preciso proyectarla bien de forma vertical en *diagramas ramificados*, o bien de forma horizontal en *paréntesis etiquetados*. En los diagramas, o árboles ramificados, las estructuras se organizan en distintos niveles, en los cuales el nodo raíz y los nodos subsidiarios permiten mostrar las relaciones de dominio y dependencia de los distintos constituyentes. Pero el problema está en que las *Gramáticas Regulares* son incapaces de generar tales estructuras.

Por otra parte, en los SSNN de estructura compleja se puede producir una *repetición* de estructuras incrustadas, para representar este fenómeno lingüístico sería preciso utilizar *reglas recursivas*. El uso de estas reglas produce un conjunto *infinito* de estructuras sintagmáticas porque se pueden utilizar indefinidamente en la generación de un mismo SN. La propiedad formal que tienen determinadas reglas de una gramática para producir un número ilimitado de estructuras incrustadas se presenta de tres modos distintos:

a) *Recursividad a la izquierda*: **A** → **A** a

b) *Recursividad a la derecha*: **A** → a **A**

c) *Autoincrustación*: **A** → a **A** b

Las gramáticas que manejan las instrucciones anteriores permiten utilizar indefinidamente la misma regla en la generación de un SN, porque tienen el mismo símbolo no-terminal en la *parte-izquierda* y en la *parte-derecha*, esto es, el mismo elemento aparece a ambos lados de las producciones. Mediante el uso de estas reglas se podrían generar los SSNN con un número ilimitado de *SSPP incrustados*, o con un número ilimitado de *Oraciones autoincrustadas*. La estructura de constituyentes de estos SSNN se obtendría mediante la aplicación sucesiva de las reglas de producción, que se encargarían de expandir, como en otros casos, un símbolo en cadenas de símbolos subordinados. El problema reside en que este tipo de reglas de producción, como ya se ha mencionado, tampoco pertenecen a las *Gramáticas Regulares* sino a las *Gramáticas Libres de Contexto*, o las *Gramáticas Sintagmáticas*.

Aunque para la representación y el reconocimiento de la mayoría de los SSNN de estructura compleja se podrían utilizar las técnicas de estado-finito, hemos puesto de manifiesto que estos formalismos tienen dos limitaciones:

1. Incapacidad para generar *estructuras jerarquizadas* de los SSNN con estructuras complejas.
2. Incapacidad para representar la *recursividad* de determinados SSNN con estructuras complejas.

En consecuencia, el tratamiento que vamos a adoptar para representar las estructuras de los SSNN que incluyen otras estructuras sintagmáticas va a consistir básicamente en trasladar tales estructuras directamente a FST gráficos, que posteriormente serán compilados y minimizados de forma automática con el editor *FSGraph.* El desarrollo de esta metodología se sintetiza en los siguientes pasos:

1. Especificar las Estructuras Complejas de los SSNN por medio de Expresiones Regulares, utilizando en muchos casos las *Gramáticas Parciales* desarrolladas previamente.
2. *Trasladar las Estructuras Complejas* directamente a transductores gráficos.
3. *Compilar* los transductores gráficos en Transductores de Estado-Finito Deterministas (FST).
4. *Minimizar los FST.*
5. Obtener los *FST* que se encarguen de insertar marcas a las variantes estructurales de los SSNN especificados.

De este modo, en los casos en los que las estructuras de los SSNN incluyan constituyentes preposicionales, u oracionales, no vamos a construir las Gramáticas Regulares Parciales que sean capaces de generarlas, no sólo por las razones ya expuestas, y que están en relación con su incapacidad para generar estructuras jerarquizadas, sino porque desde el punto de vista práctico serían demasiado extensas. Por otra parte, para la representación y reconocimiento de los SSNN que incluyen *recursividad de constituyentes* se va a utilizar un procedimiento análogo al expuesto arriba, pero en el que se va a poner un límite a los fenómenos recursivos, restringiendo el número de *transductores imbricados*, tal y como se va a describir y desarrollar en los siguientes apartados.

6.2.1. SSNN de Estructura Compleja

Los SSNN de estructura compleja pueden estar acompañados por SSPP que funcionan como modificadores restrictivos, o pueden estar seguidos por una *Oración* subordinada, introducida por un pronombre relativo, que restringe el contenido del sintagma dominante. Esta distinción nos lleva a considerar dos *hipótesis explicativas* generales:

1. La especificación de las estructuras de los *SSNN modificados por SSPP* se podría formalizar en la siguiente regla:

   SN → N PREP N (**Nombre** Preposición Nombre)

   Si aplicáramos técnicas de estado-finito para generar los SSNN de estructura compleja modificados por SSPP tendríamos que especificar las construcciones sintagmáticas en términos de Expresiones Regulares, obtener las derivadas de dichas expresiones, construir las Gramáticas Regulares Parciales y trasladarlas a los reconocedores de estado-finito. El proceso sería semejante al adoptado en los SSNN de estructura simple.

   En el caso de que quisiéramos representar un sencillo grupo nominal como **SN → N PREP N**, que nos permitiera reconocer y etiquetar construcciones lingüísticas del tipo «*base de conocimiento*», o «*redes de información*», procederíamos de la forma siguiente:

   SN → N PREP N (**Nombre Preposición Nombre**)

   ER_0 = N PREP N

   $D_N(ER_0) =$
   $D_N(N)$ PREP N + $\alpha(N)$ $D_N(PREP\ N) =$
   λ PREP N + $\alpha(N)$ [$D_N(PREP)$ N + $\alpha(PREP)$ $D_N(N)$] =
   PREP N + $\varnothing$ [$\varnothing$ + $\varnothing$] =
   PREP N = ER_1

   $D_{PREP}(ER_0) =$
   $D_{PREP}(N)$ PREP N + $\alpha(N)$ $D_{PREP}(PREP\ N) =$
   $\varnothing$ PREP N + $\alpha(N)$ [$D_{PREP}(PREP)$ N + $\alpha(PREP)$ $D_{PREP}(N)$] =
   $\varnothing$ + $\varnothing$ [λ N + $\varnothing$ $\varnothing$] = $\varnothing$

   $D_N(ER_0) =$
   $D_N(N)$ PREP N + $\alpha(N)$ + $D_N(PREP\ N) =$
   $\varnothing$ PREP N + $\alpha(N)$ [$D_N(PREP)$ N + $\alpha(PREP)$ $D_N(N)$] =
   $\varnothing$ + $\varnothing$ [$\varnothing$ N + $\varnothing$ λ] = $\varnothing$

$D_N(ER_1) =$
$D_N(PREP)\ N + \alpha(PREP)\ D_N(N) =$
$\varnothing\ N + \varnothing\ \lambda = \varnothing$

$D_{PREP}(ER_1) =$
$D_{PREP}(PREP)\ N + \alpha(PREP)\ D_{PREP}(N) =$
$\lambda\ N + \varnothing\ \varnothing =$
$N = ER_2$

$D_N(ER_1) =$
$D_N(PREP)\ N + \alpha(PREP)\ D_N(N) =$
$\varnothing\ N + \varnothing\ \lambda = \varnothing$

$D_N(ER_2) =$
$D_N(N) = \lambda$

$D_{PREP}(ER_2) =$
$D_{PREP}(N) = \varnothing$

$D_N(ER_2) =$
$D_N(N) = \lambda$

La *Gramática Regular* que reconoce el SN anterior sería la siguiente:

$G = (\{N, PREP, N\}, \{ER_0, ER_1, ER_2\}, ER_0, P)$

donde las *reglas de producción*, P, se definirían como:

$ER_0 ::= N\ ER_1$
$ER_1 ::= PREP\ ER_2$
$ER_2 ::= N$

El *Autómata Finito*, que reconocería el lenguaje generado por la gramática, sería:

$AF = (\{N, PREP, N\}, \{ER_0, ER_1, ER_2, F\}, f, ER_0, F)$

donde la función de transición, f, se definiría como:

$f\ (ER_0, N) = ER_1$
$f\ (ER_0, PREP) = \varnothing$
$f\ (ER_0, N) = \varnothing$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_1, PREP) = ER_2$
$f\ (ER_1, N) = \varnothing$
$f\ (ER_2, N) = F$
$f\ (ER_2, PREP) = \varnothing$
$f\ (ER_2, N) = F$
$f\ (F, N) = \varnothing$
$f\ (F, PREP) = \varnothing$
$f\ (F, N) = \varnothing$

Se eliminarían las transiciones vacías, se renombrarían los estados y se redefiniría la función de transición del AF Mínimo:

$f\ (q_0, N) = q_1$
$f\ (q_1, PREP) = q_2$
$f\ (q_2, N) = q_3$

El AFD se puede representar además en un *diagrama de transiciones* (Fig. 6.78), o en una *tabla de transiciones* (Fig. 6.79).

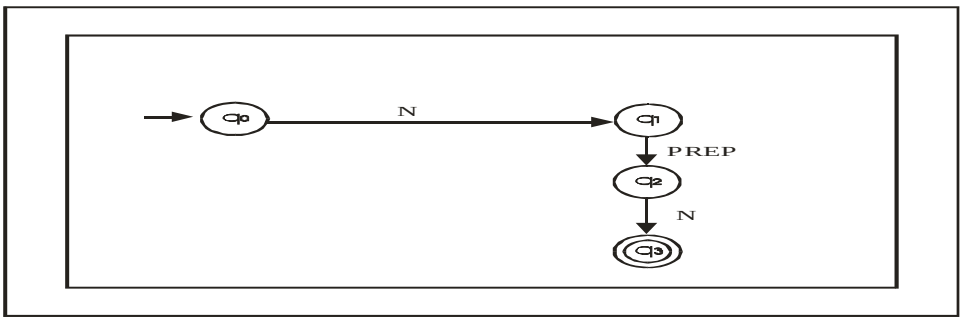


Fig. 6.78: Diagrama de transiciones del AFD que reconoce la estructura compleja de un SN

f	N	PREP	N
→ q_0	q_1	∅	q_1
q_1	∅	q_2	∅
q_2	q_3	∅	q_3
*q_3	∅	∅	∅

Fig. 6.79: Tabla de transiciones del AFD que reconoce la estructura compleja de un SN

Por último, se obtendría el transductor gráfico (Fig. 6.80) que se encargaría de reconocer la estructura compleja del SN:

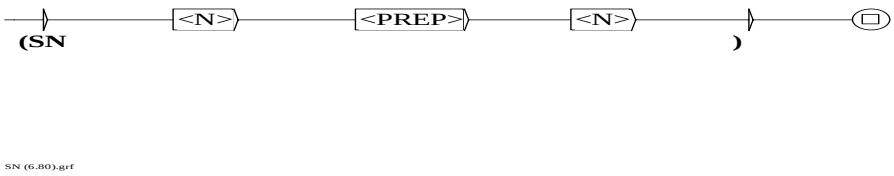


Fig. 6.80: FST gráfico que reconoce la estructura compleja de un SN con un constituyente preposicional

2. La especificación de las estructuras complejas de los *SSNN modificados por una Oración de relativo* se podría formalizar en la siguiente regla:

SN → DET N PRORE V DET N (Det **Nombre Pronombre Relativo Verbo Det Nombre**)

La estructura interna del SN anterior presenta la característica de que la *Oración* constituyentes va siempre introducida por un *nexo subordinante*, que tiene como función vincular la Oración al núcleo nominal. Las cláusulas susceptibles de formar las estructuras complejas de los SSNN se denominan *Oraciones de relativo*, y estarían introducidas por distintos *nexos subordinantes*: *a) Pronombres relativos*

(**PRORE**): *que*, *quien*, precedidos eventualmente de preposición; *b) Adjetivos relativos* (ARE): *cual*, precedido de determinante, *cuanto* y *cuyo*; y *c) Adverbios relativo* (**ADVRE**): *cuando*, *como* y *donde*.

A su vez, las cláusulas de relativo pueden contribuir al desarrollo del contenido del sintagma dominante, o simplemente añadir información adicional, dando lugar a la siguiente distinción:

- *Oraciones de relativo especificativas*, en las que los constituyentes oracionales limitan o restringen el núcleo nominal. Funcionan como ***modificadores restrictivos***, o modificadores necesarios, que se incrustan al SN considerado matriz.

- *Oraciones de relativo explicativas*, en las que los constituyentes oracionales desarrollan más exactamente la magnitud del núcleo nominal. Funcionan como ***modificadores no restrictivos,*** o modificadores no necesarios, y se colocan normalmente entre pausas, o *comas.*

Nuestro objetivo se va a centrar exclusivamente en las cláusulas de relativo especificativas que actúan como ***modificadores restrictivos***. A partir de aquí, si aplicáramos técnicas de estado-finito para generar SSNN con tales constituyentes adoptaríamos la misma metodología utilizada hasta ahora: *especificar los SSNN en términos de Expresiones Regulares, obtener las derivadas de dichas expresiones, construir las Gramáticas Regulares Parciales y trasladarlas a los distintos reconocedores de estado-finito*.

En el caso concreto de que quisiéramos representar el grupo nominal **SN → DET N PRORE V DET N** , que nos permitiera reconocer y etiquetar construcciones sintagmáticas del tipo «*el sistema de recuperación que utiliza el catálogo*»,

emplearíamos el proceso que hemos seguido hasta ahora y que nos llevaría a la obtención del transductor gráfico, capaz de reconocer tal estructura (Fig. 6.81).

SN → DET N PRORE V DET N

ER_0 = DET N PRORE V DET N

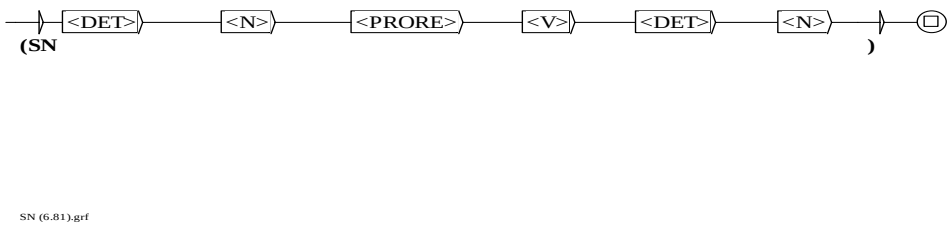


Fig. 6.81: FST gráfico que reconoce la estructura de un SN con un constituyente oracional

Sin embargo, las técnicas de estado-finito tiene dos inconvenientes, a los que ya hemos hecho alusión: las Gramáticas Regulares obtenidas serían demasiado extensas y los transductores gráficos resultantes sólo proporcionarían una representación *lineal* de los constituyentes de los SSNN, en la que no sería posible diferenciar el sintagma constituyente que modifica al sintagma considerado el *núcleo* de tales construcciones lingüísticas. Por esta razón, y con el objetivo de obtener una representación en la que sí se pueda distinguir los modificadores del núcleo nominal, vamos a trasladar directamente los constituyentes a FST gráficos. A su vez, para representar los SSNN de estructura compleja, vamos a utilizar las *Gramáticas Parciales* construidas anteriormente y las vamos a insertar a modo de *transductores imbricados*.

Siguiendo el mismo procedimiento, para formalizar las construcciones sintagmáticas complejas con *recursividad de constituyentes* vamos a utilizar las *Gramáticas Parciales* y los FST gráficos construidos anteriormente como parte de otra *Gramática Global*. De esta forma, las Gramáticas Parciales desarrolladas previamente se van a utilizar a modo de *transductores imbricados*. Para resolver el problema de la limitación de los mecanismos de estado-finito para representar fenómenos recurusivos, se va a *limitar el número de estructuras incrustadas* en los SSNN con tales constituyentes. La descripción y representación de las estructuras de estos SSNN se va a establecer en los siguientes apartados.

6.2.1.1. SSNN modificados por constituyentes preposicionales

Aplicando la metodología expuesta en el apartado anterior la construcción de los analizadores de SSNN de estructura compleja modificados por SSPP se establece del modo siguiente:

1. Transductor gráfico que representa, agrupa y reconoce la construcción sintagmática $_{SN_{71}}$ (Fig. 6.82):

$$\mathbf{SN_{71} \rightarrow SN_{55}\ PREP\ SN_{55}}$$

$$ER_0 = \underline{N}\ PREP\ \underline{N}$$
$$\underline{N\ N}\ PREP\ \underline{N\ N}$$
$$\underline{DET\ N}\ PREP\ \underline{DET\ N}$$
$$\ldots$$

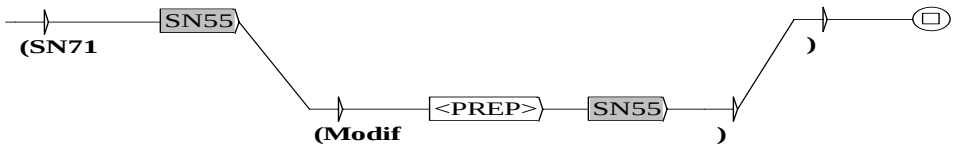


Fig. 6.82: FST gráfico que agrupa las variantes de la estructura compleja SN_{71}

2. Transductor gráfico que representa, reconoce y agrupa la construcción sintagmática SN_{72} (Fig. 6.83):

$SN_{72} \rightarrow SN_{55}$ PREP SN_{56}

ER_0 = N PREP N A
N N PREP N A
DET N PREP N A
. . .

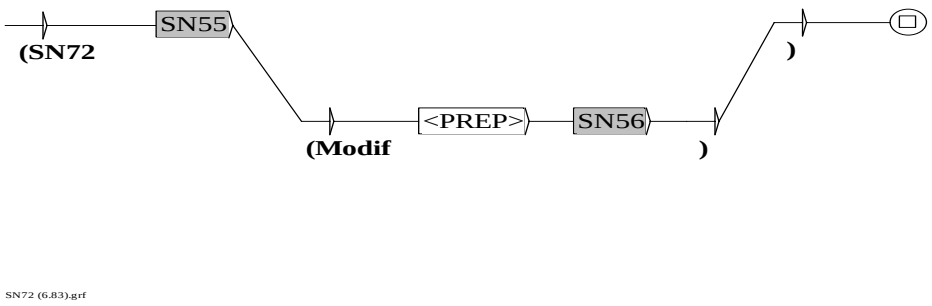


Fig. 6.83: FST gráfico que agrupa las variantes de la estructura compleja SN_{72}

3. Transductor gráfico que representa, reconoce y agrupa la construcción sintagmática SN_{73} (Fig. 6.84):

$SN_{73} \rightarrow SN_{56}$ PREP SN_{56}

$ER_0 =$ N A PREP N A
N A PREP A N
N A PREP N PA
...

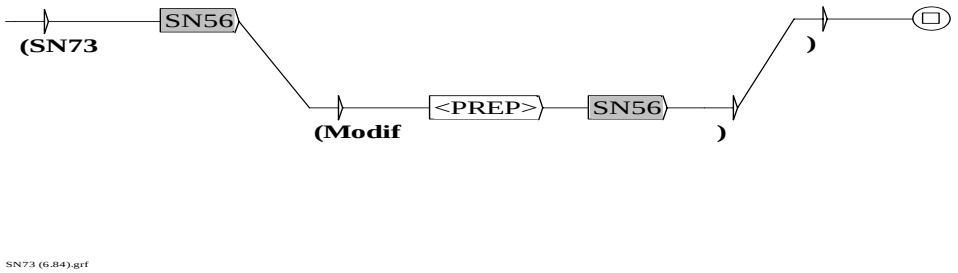


Fig. 6.84: FST gráfico que agrupa las variantes de la estructura compleja SN_{73}

4. Transductor gráfico que representa, reconoce y agrupa la construcción sintagmática SN_{74} (Fig. 6.85):

$SN_{74} \rightarrow SN_{56}$ PREP SN_{55}

$ER_0 =$ N A PREP N
A N PREP N N
N PA PREP DET N
...

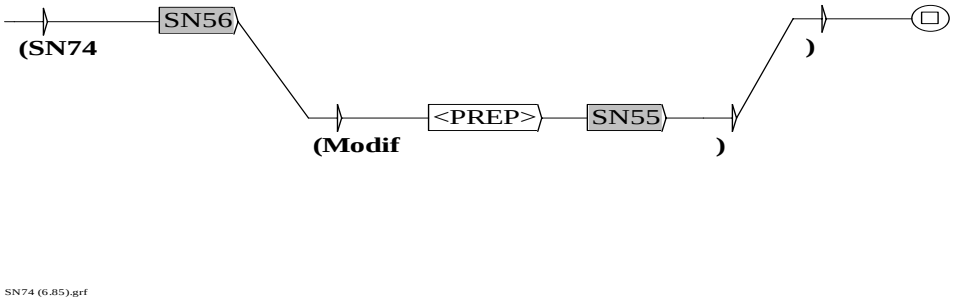


Fig. 6.85: FST gráfico que agrupa las variantes de la estructura compleja SN_{74}

5. Transductor gráfico que representa, reconoce y agrupa la construcción sintagmática SN_{75} con iteración de constituyentes (Fig. 6.86):

$SN_{75} \rightarrow SN_{57}$ PREP SN_{57}

$ER_0 =$ DET N^+ PREP DET N^+
N$^+$ PREP N$^+$
CUANT N^+ PREP CUANT N^+
...

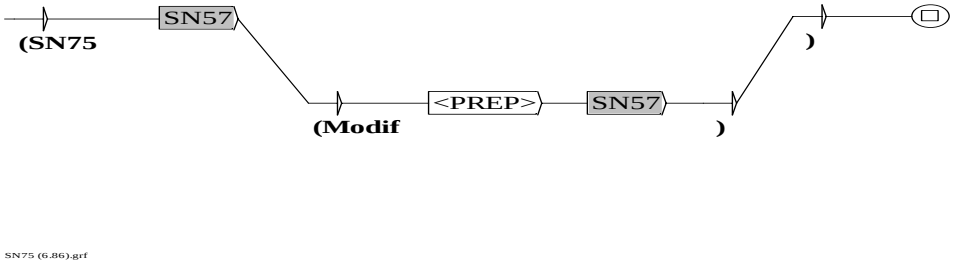


Fig. 6.86: FST gráfico que agrupa las variantes de la estructura compleja SN_{75}

6. Transductor gráfico que representa, reconoce y agrupa la construcción sintagmática SN_{76} con iteración de constituyentes (Fig. 6.87):

$$\mathbf{SN_{76}} \rightarrow \mathbf{SN_{57}\ PREP\ SN_{58}}$$
$$\mathbf{SN_{57}\ PREP\ SN_{59}}$$
$$\mathbf{SN_{57}\ PREP\ SN_{60}}$$
$$\ldots$$

$$ER_0 = \underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+\ \text{A}}$$
$$\underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N A}^+}$$
$$\underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+\ \text{A}^+}$$
$$\ldots$$

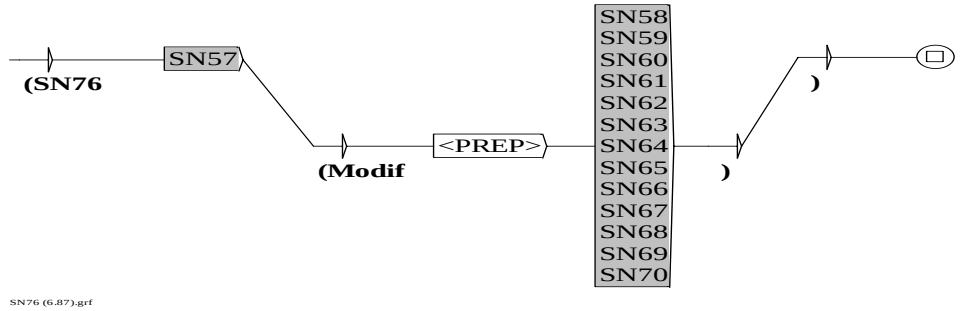


Fig. 6.87: FST gráfico que agrupa las variantes de la estructura compleja SN_{76}

7. Transductor gráfico que representa, reconoce y agrupa la construcción sintagmática SN_{77} con iteración de constituyentes (Fig. 6.88):

$$\mathbf{SN_{77}} \rightarrow \mathbf{SN_{58}\ PREP\ SN_{58}}$$
$$\mathbf{SN_{58}\ PREP\ SN_{59}}$$
$$\mathbf{SN_{58}\ PREP\ SN_{60}}$$
$$\ldots$$

$ER_0 =$ DET N^+ A PREP DET N^+ A
DET N^+ A PREP DET N A^+
DET N^+ A PREP DET N^+ A^+
...

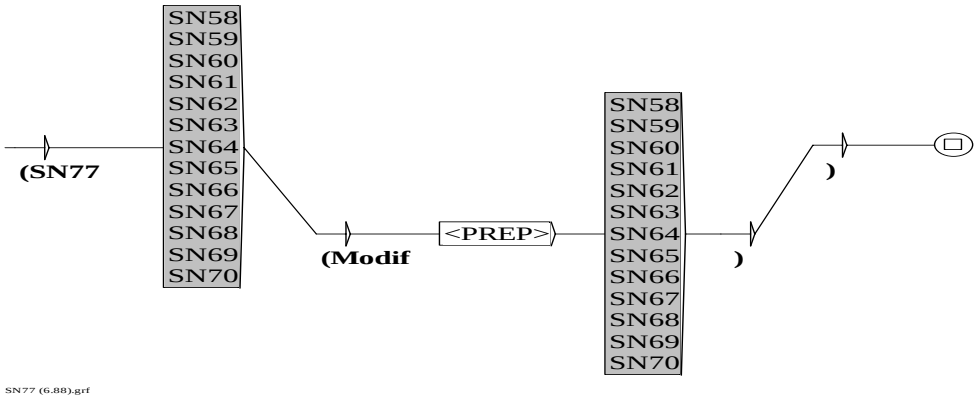


Fig. 6.88: FST gráfico que agrupa las variantes de la estructura compleja SN_{77}

8. Transductor gráfico que representa, reconoce y agrupa la construcción sintagmática SN_{78} con iteración de constituyentes (Fig. 6.89):

$SN_{78} \rightarrow$ SN_{58} PREP SN_{57}
SN_{59} PREP SN_{57}
SN_{60} PREP SN_{57}
...

$ER_0 =$ DET N^+ A PREP DET N^+
DET N A^+ PREP DET N^+
DET N^+ A^+ PREP DET N^+
...

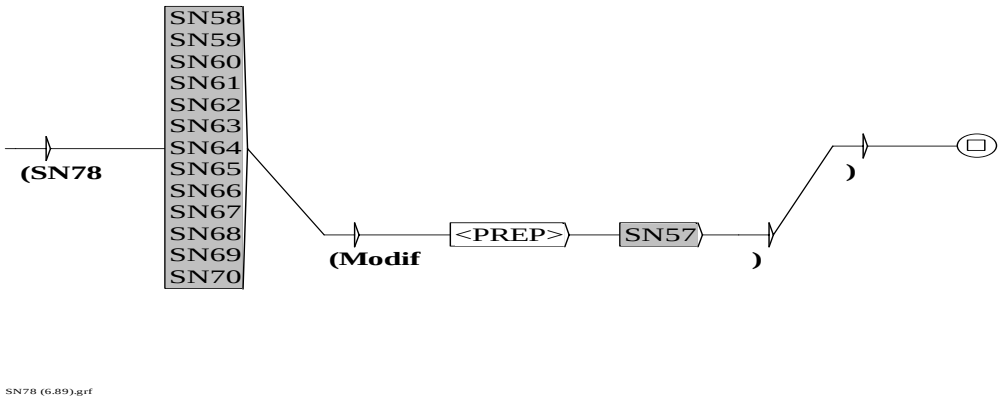


Fig. 6.89: FST gráfico que agrupa las variantes de la estructura compleja SN_{78}

6.2.1.2. SSNN modificados por constituyentes oracionales

La construcción de los analizadores de SSNN modificados por *Oraciones de relativo* se establece, según la metodología que estamos siguiendo, del modo siguiente:

1. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{79} (Fig. 6.90):

$$\mathbf{SN_{79} \rightarrow SN_{55}\ PRORE\ V\ SN_{55}}$$
$$\mathbf{SN_{55}\ PRORE\ V\ SN_{56}}$$
$$\mathbf{SN_{55}\ ARE\ V\ SN_{55}}$$
$$\mathbf{\ldots}$$

$$ER_0 = \underline{N}\ PRORE\ V\ \underline{N}$$
$$\underline{N\ N}\ PRORE\ V\ \underline{N\ A}$$
$$\underline{DET\ N}\ PRORE\ \underline{PA\ N}$$
$$\ldots$$

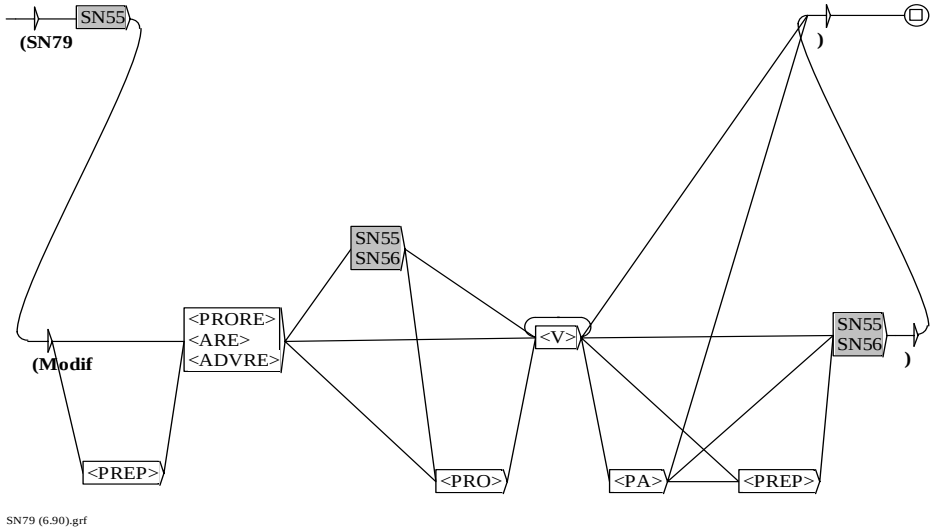


Fig.6.90: FST gráfico que agrupa las variantes de la estructura compleja SN_{79}

2. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{80} (Fig. 6.91):

$\mathbf{SN_{80}} \rightarrow \mathbf{SN_{56}\ PRORE\ V\ SN_{55}}$
$\mathbf{SN_{56}\ PRORE\ V\ SN_{56}}$
$\mathbf{SN_{56}\ ARE\ V\ SN_{55}}$
...

$ER_0 =$ N A PRORE V N
N A PRORE V N A
N A ARE V N
...

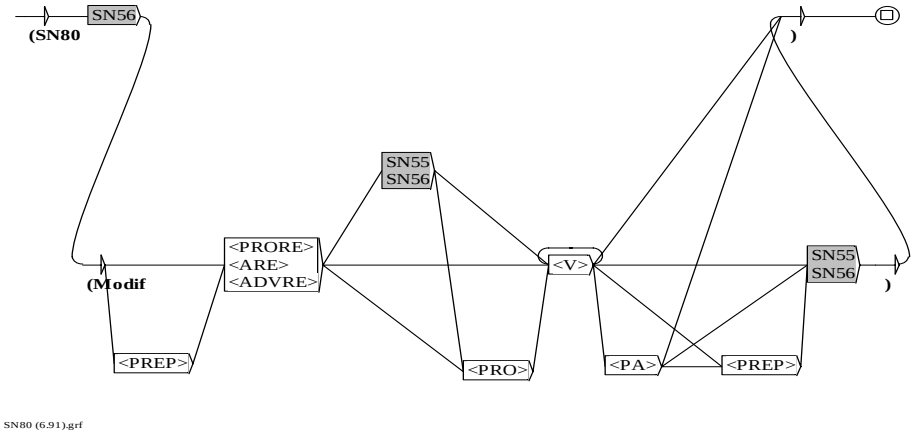


Fig. 6.91: FST gráfico que agrupa las variantes de la estructura compleja SN_{80}

3. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{81} con iteración de constituyentes (Fig. 6.92):

$$\mathbf{SN_{81} \rightarrow SN_{57}\ PRORE\ V\ SN_{57}}$$
$$\mathbf{SN_{57}\ ARE\ V\ SN_{57}}$$
$$\mathbf{SN_{57}\ ADVE\ SN_{57}}$$
$$\mathbf{\ldots}$$

$$ER_0 = \underline{DET\ N^+}\ PRORE\ V\ \underline{DET\ N^+}$$
$$\underline{DET\ N^+}\ ARE\ V\ \underline{DET\ N^+}$$
$$\underline{DET\ N^+}\ ADVE\ V\ \underline{DET\ N^+}$$
$$\ldots$$

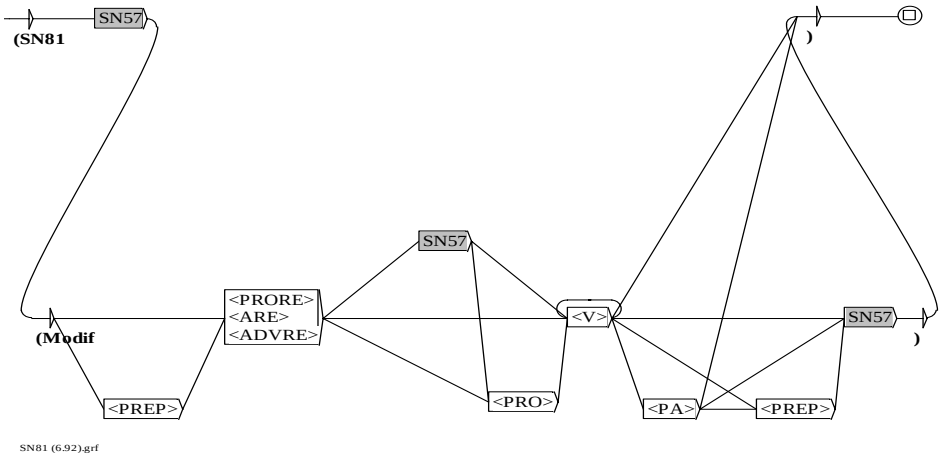


Fig. 6.92: FST gráfico que agrupa las variantes de la estructura compleja SN_{81}

4. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{82} con iteración de consituyentes (Fig. 6.93):

$$\mathbf{SN_{82} \rightarrow SN_{57}\ PRORE\ V\ SN_{58}}$$
$$\mathbf{SN_{57}\ PRORE\ V\ SN_{59}}$$
$$\mathbf{SN_{57}\ PRORE\ V\ SN_{60}}$$
$$\mathbf{...}$$

$$ER_0 = \underline{DET\ N^+}\ PRORE\ V\ \underline{DET\ N^+\ A}$$
$$\underline{DET\ N^+}\ PRORE\ V\ \underline{DET\ N\ A^+}$$
$$\underline{DET\ N^+}\ PRORE\ V\ \underline{DET\ N^+\ A^+}$$
$$...$$

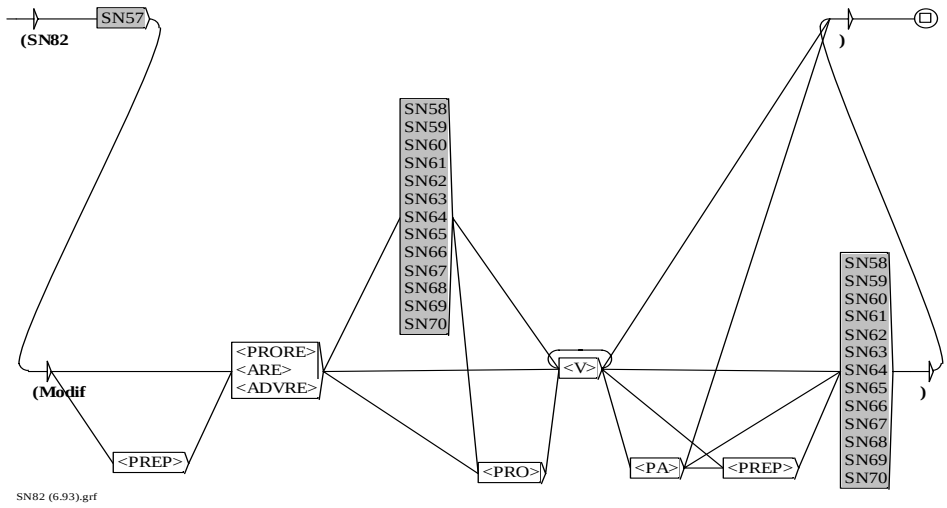


Fig. 6.93: FST gráfico que agrupa las variantes de la estructura compleja SN_{82}

5. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{83} con iteración de constituyentes (Fig. 6.94):

$$\mathbf{SN_{83}} \rightarrow \mathbf{SN_{58}\ PRORE\ V\ SN_{58}}$$
$$\mathbf{SN_{58}\ PRORE\ V\ SN_{59}}$$
$$\mathbf{SN_{58}\ PRORE\ V\ SN_{60}}$$
$$\mathbf{\ldots}$$

$$ER_0 = \underline{DET\ N^+\ A}\ PRORE\ V\ \underline{DET\ N^+\ A}$$
$$\underline{DET\ N^+\ A}\ PRORE\ V\ \underline{DET\ N\ A^+}$$
$$\underline{DET\ N^+\ A}\ PRORE\ V\ \underline{DET\ N^+\ A^+}$$
$$\ldots$$

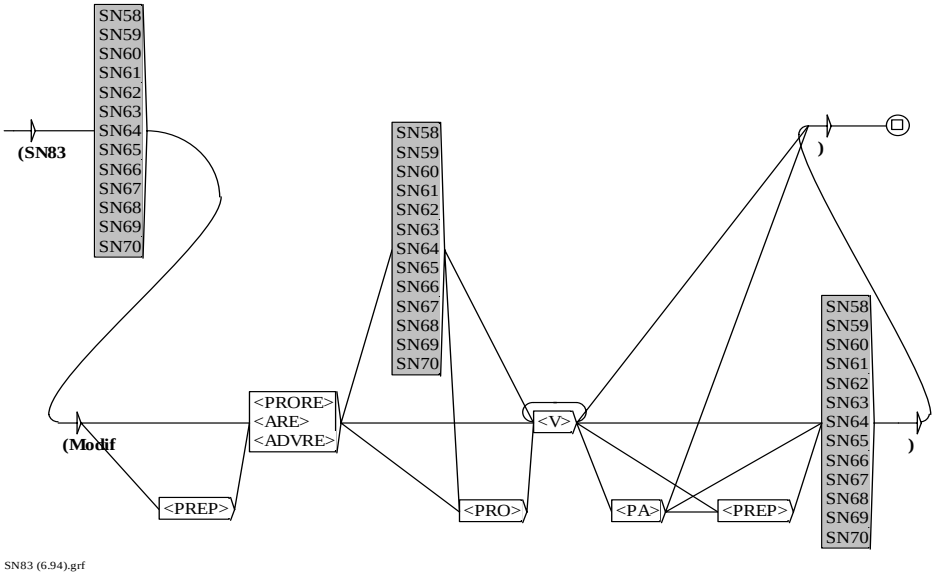


Fig. 6.94: FST gráfico que agrupa las variantes de la estructura compleja SN_{83}

6. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{84} (Fig. 6.95):

$$\mathbf{SN_{84}} \rightarrow \mathbf{SN_{55}\ PREP\ SN_{55}\ \ PRORE\ V\ SN_{55}\ \ PREP\ SN_{55}}$$
$$\mathbf{SN_{55}\ PREP\ SN_{56}\ \ PRORE\ V\ SN_{55}\ \ PREP\ SN_{55}}$$
$$\mathbf{SN_{56}\ PREP\ SN_{56}\ \ PRORE\ V\ SN_{56}\ \ PREP\ SN_{56}}$$
$$\ldots$$

$ER_0 =$ N PREP N PRORE V N PREP N
N N PREP N A PRORE V N N PREP N N
N A PREP N A PRORE V N A PREP N A
. . .

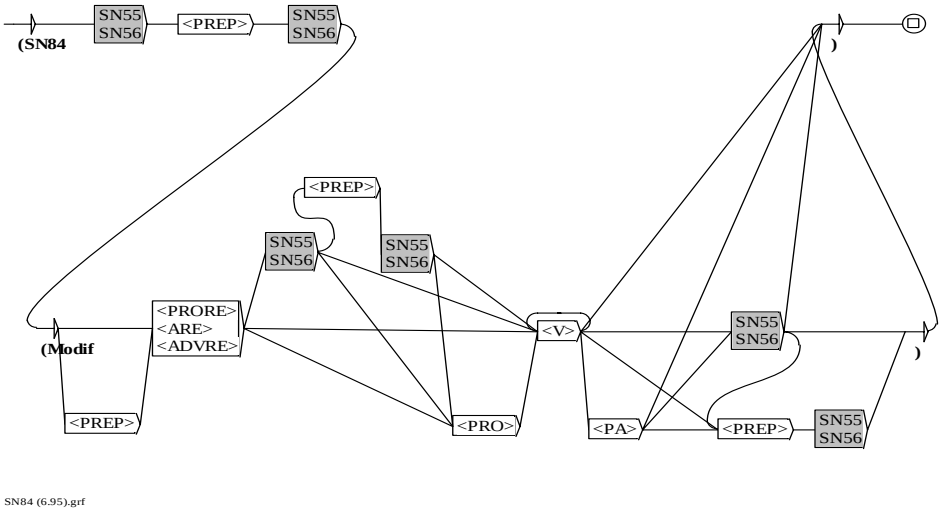


Fig. 6.95: FST gráfico que agrupa las variantes de la estructura compleja SN_{84}

7. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{85} con iteración de constituyentes (Fig. 6.96):

$$SN_{85} \rightarrow SN_{57}\ \text{PREP}\ SN_{57}\ \text{PRORE V}\ SN_{57}\ \text{PREP}\ SN_{57}$$
$$SN_{57}\ \text{PREP}\ SN_{57}\ \text{ARE V}\ SN_{57}\ \text{PREP}\ SN_{57}$$
$$SN_{57}\ \text{PREP}\ SN_{57}\ \text{ADVRE V}\ SN_{57}\ \text{PREP}\ SN_{57}$$
$$\ldots$$

$$ER_0 = \underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+}\ \text{PRORE V}\ \underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+}$$
$$\underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+}\ \text{ARE V}\ \underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+}$$
$$\underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+}\ \text{ADVRE V}\ \underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+}$$
$$\ldots$$

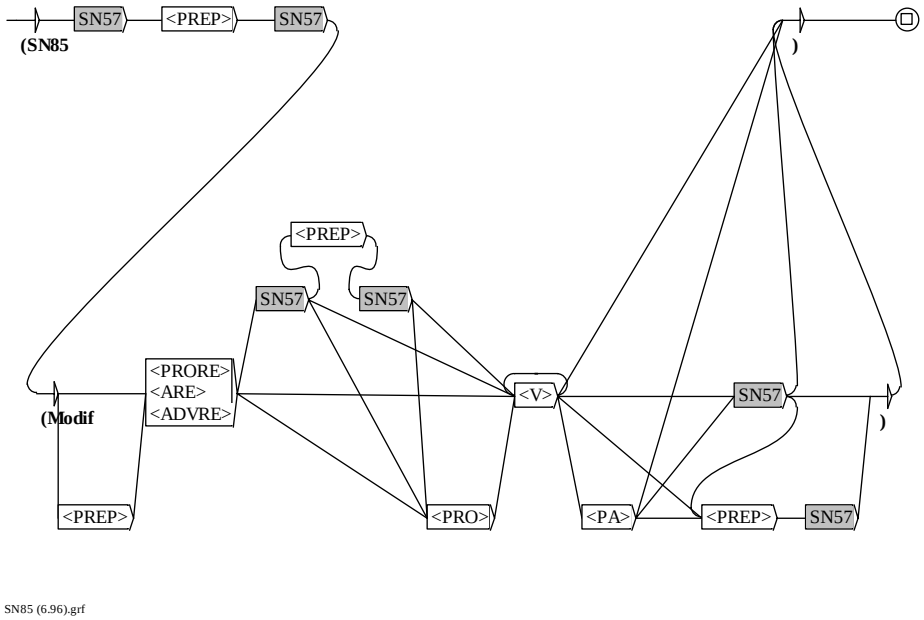


Fig. 6.96: FST gráfico que agrupa las variantes de la estructura compleja SN_{85}

8. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{86} con iteración de constituyentes (Fig. 6.97):

$$\mathbf{SN_{86} \rightarrow SN_{57}\ \ PREP\ SN_{57}\ \ PRORE\ V\ SN_{58}\ \ PREP\ SN_{58}}$$
$$\mathbf{SN_{57}\ \ PREP\ SN_{57}\ \ PRORE\ V\ SN_{58}\ \ PREP\ SN_{59}}$$
$$\mathbf{SN_{57}\ \ PREP\ SN_{57}\ \ ARE\ V\ SN_{58}\ \ PREP\ SN_{59}}$$
$$\mathbf{\ldots}$$

$$ER_0 = \underline{DET\ N^+}\ PREP\ \underline{DET\ N^+}\ PRORE\ V\ \underline{DET\ N^+\ A}\ PREP\ \underline{DET\ N^+\ A}$$
$$\underline{DET\ N^+}\ PREP\ \underline{DET\ N^+}\ PRORE\ V\ \underline{DET\ N^+\ A}\ PREP\ \underline{DET\ N\ A^+}$$
$$\underline{DET\ N^+}\ PREP\ \underline{DET\ N^+}\ ARE\ V\ \underline{DET\ N^+\ A}\ PREP\ \underline{DET\ N\ A^+}$$
$$\ldots$$

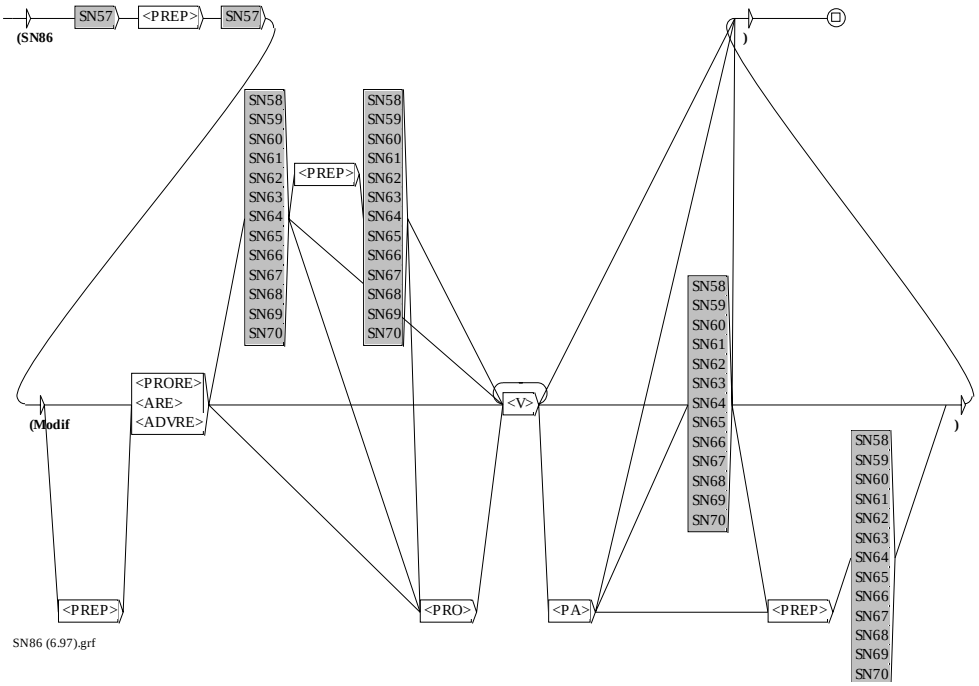


Fig. 6.97: FST gráfico que agrupa las variantes de la estructura compleja SN_{86}

9. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{87} con iteración de constituyentes (Fig. 6.98):

$$SN_{87} \rightarrow SN_{57}\ \text{PREP}\ SN_{58}\ \text{PRORE V}\ SN_{57}\ \text{PREP}\ SN_{57}$$
$$SN_{57}\ \text{PREP}\ SN_{59}\ \text{PRORE V}\ SN_{57}\ \text{PREP}\ SN_{57}$$
$$SN_{57}\ \text{PREP}\ SN_{60}\ \text{PRORE V}\ SN_{57}\ \text{PREP}\ SN_{57}$$
$$\ldots$$

$$ER_0 = \underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+\ \text{A}}\ \text{PRORE V}\ \underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+}$$
$$\underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N A}^+}\ \text{PRORE V}\ \underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+}$$
$$\underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+\ \text{A}^+}\ \text{PRORE V}\ \underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+}$$
$$\ldots$$

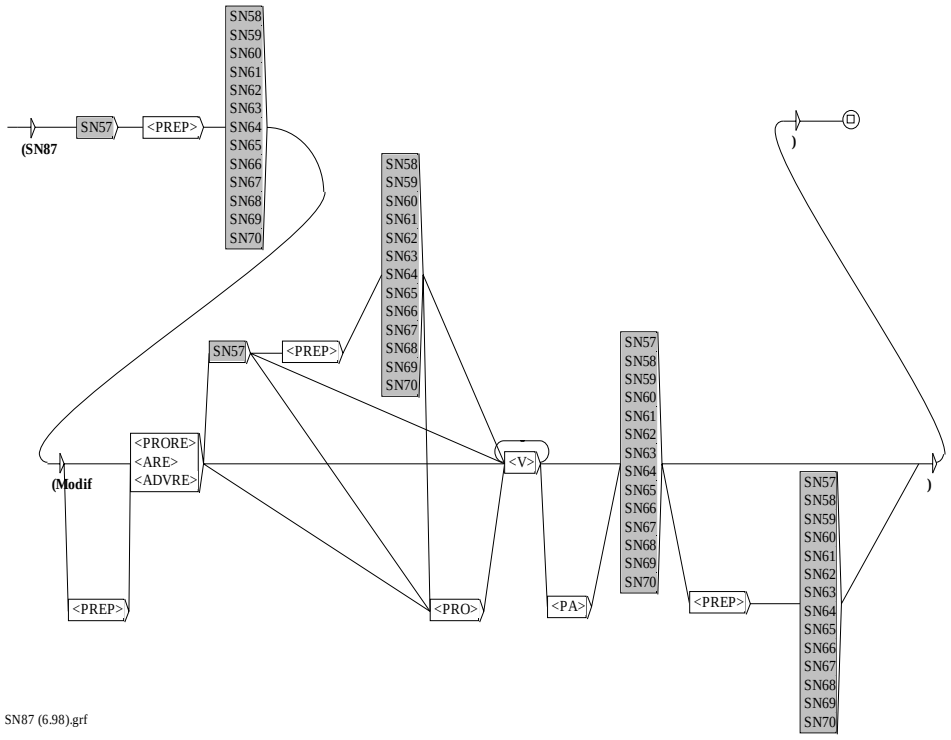


Fig. 6.98: FST gráfico que agrupa las variantes de la estructura compleja SN_{87}

10. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{88} con iteración de constituyentes (Fig. 6.99):

$$\mathbf{SN_{88} \rightarrow SN_{58}\ \ PREP\ SN_{58}\ \ PRORE\ V\ SN_{58}\ \ PREP\ SN_{58}}$$
$$\mathbf{SN_{59}\ \ PREP\ SN_{58}\ PRORE\ V\ SN_{59}\ \ PREP\ SN_{58}}$$
$$\mathbf{SN_{60}\ \ PREP\ SN_{60}\ PRORE\ V\ SN_{59}\ \ PREP\ SN_{58}}$$
$$\mathbf{\ldots}$$

$$ER_0 = \underline{DET\ N^+\ A}\ PREP\ \underline{DET\ N^+\ A}\ PRORE\ V\ \underline{DET\ N^+\ A}\ PREP\ \underline{DET\ N^+\ A}$$
$$\underline{DET\ N\ A^+}\ PREP\ \underline{DET\ N^+\ A}\ PRORE\ V\ \underline{DET\ N\ A^+}\ PREP\ \underline{DET\ N^+\ A}$$
$$\underline{DET\ N^+\ A^+}\ PREP\ \underline{DET\ N^+\ A^+}\ PRORE\ V\ \underline{DET\ N\ A^+}\ PREP\ \underline{DET\ N^+\ A}$$
$$\ldots$$

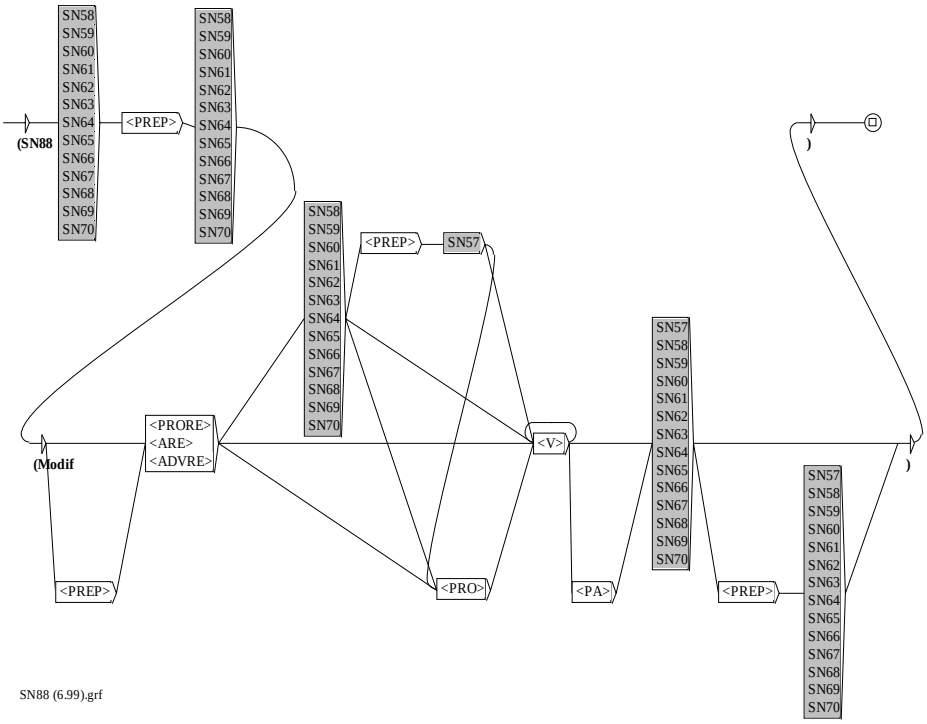


Fig. 6.99: FST gráfico que agrupa las variantes de la estructura compleja SN_{88}

11. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{89} con iteración de constituyentes (Fig. 6.100):

$$\begin{aligned} \mathbf{SN_{89}} \rightarrow\ & \mathbf{SN_{58}}\ \ \mathbf{PREP\ SN_{57}}\ \ \mathbf{PRORE\ V\ SN_{57}}\ \ \mathbf{PREP\ SN_{57}} \\ & \mathbf{SN_{59}}\ \ \mathbf{PREP\ SN_{57}}\ \ \mathbf{PRORE\ V\ SN_{57}}\ \ \mathbf{PREP\ SN_{57}} \\ & \mathbf{SN_{60}}\ \ \mathbf{PREP\ SN_{57}}\ \ \mathbf{PRORE\ V\ SN_{57}}\ \ \mathbf{PREP\ SN_{57}} \\ & \mathbf{...} \end{aligned}$$

$$\begin{aligned} ER_0 = & \underline{\text{DET N}^+}\ \text{A PREP}\ \underline{\text{DET N}^+}\ \text{PRORE V}\ \underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+} \\ & \underline{\text{DET N A}^+}\ \text{PREP}\ \underline{\text{DET N}^+}\ \text{PRORE V}\ \underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+} \\ & \underline{\text{DET N}^+\ \text{A}^+}\ \text{PREP}\ \underline{\text{DET N}^+}\ \text{PRORE V}\ \underline{\text{DET N}^+}\ \text{PREP}\ \underline{\text{DET N}^+} \\ & \ldots \end{aligned}$$

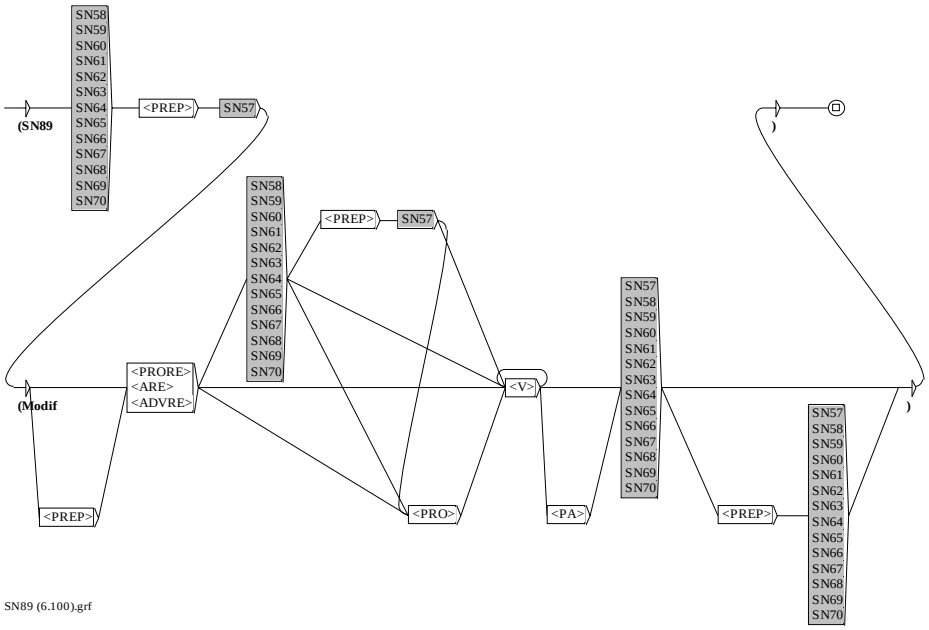


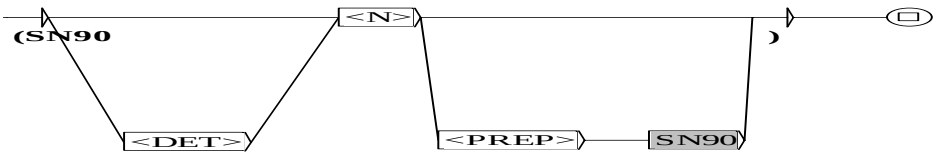
Fig. 6.100: FST gráfico que agrupa las variantes de la estructura compleja SN_{89}

6.2.2. SSNN de Estructura Compleja con *recursividad* de constituyentes

La representación de SSNN de estructura compleja, en los cuales aparece bien un número ilimitado de *SSPP incrustados* o bien un número ilimitado de *Oraciones de relativo autoincrustadas*, se realiza por medio de la posibilidad de utilización de *reglas recursivas*. Las reglas recursivas tienen la capacidad de generar un número infinito de estructuras porque las *producciones* presentan el mismo símbolo no-terminal en la *parte-izquierda* y en la *parte-derecha*. A su vez, la propiedad que tienen las gramáticas de emplear más de un vez la misma regla en la generación de una construcción lingüística se denomina *recursividad*, y esta propiedad es la que les permite generar un número ilimitado de estructuras lingüísticas con medios limitados.

El problema está en que las reglas recursivas no pertenecen a las Gramáticas Regulares y, en consecuencia, los mecanismos de estado-finito no pueden generar los SSNN con tales constituyentes. Con la finalidad de representar estas estructuras sintagmáticas partimos de dos planteamientos:

1. En los casos en los que se precise representar *SSNN con recursividad de constituyentes preposicionales* sería necesario utilizar reglas que permitieran generar estructuras lingüísticas con *recursividad a la derecha*, como $\mathbf{A} \rightarrow$ a $\mathbf{A}$. Este tipo de reglas pertenecen a las Gramáticas Libres de Contexto, o a las Gramáticas Sintagmáticas, y, por tanto, no se puedan representar con mecanismos de estado-finito. No obstante, se puede crear una aproximación a estas gramáticas, con técnicas de estado-finito, que consiste en trasladar las estructuras recursivas a un *transductor gráfico directamente recursivo* que contiene infinitas referencias a sí mismo SN90 (Fig. 6.101):



SN90 (6.101).grf

Fig. 6.101: FST gráfico que reconoce SSNN con un número limitado de SSPP incrustados

Con este procedimiento, la recurrencia de las estructuras se traslada a un *transductor autorrefencial* y, aunque desde un punto de vista teórico, las estructuras recursivas no se puedan representar con mecanismos de estado-finito, debido a que la recursividad permite *la posibilidad de la no-terminación* en la

generación de estructuras, el editor *FSGraph* aporta una solución intermedia consistente en limitar el número de estructuras incrustada a *tres niveles*. De esta forma, aunque el FST gráfico SN_{90} contenga *infinitas* referencias así mismo, se puede compilar en mecanismos de estado-finito porque el editor reduce el número de estructuras incrustadas, esto es, limita el número de FST imbricados.

Si aplicáramos el FST SN_{90} a la secuencia lingüística: «*el sistema de recuperación de información del catálogo de la biblioteca de la Universidad de Granada...*» se obtendría el siguiente resultado:

2 matches

1:

(SN90 el sistema de(SN90 recuperación de(SN90 información de(SN90 el catálogo de(SN90 la biblioteca)))))

2:

de (SN90 la Universidad de(SN90 Granada))

El método anterior ofrecería una solución parcial porque simplemente limita el número de estructuras *anidadas*. Sin embargo, existe la posibilidad de utilizar otro procedimiento consistente en crear Gramáticas Regulares equivalentes a las Gramáticas Libres de Contexto, que son las que generan este tipo de estructuras. Si adoptáramos esta forma de enfocar el problema, procederíamos de forma análoga a como se ha realizado hasta ahora: *especificar* las estructuras complejas en términos de Expresiones Regulares, obtener las derivadas de estas expresiones, *construir las Gramáticas Regulares* a partir de dichas derivaciones y, por último, trasladar *las gramáticas a FST gráficos* (Fig. 6.102):

SN → (DET) N SP
SP → PREP SN
SN → (DET) N PREP SN

$ER_0 = (DET)\ N\ (PREP\ (DET)\ N)^*$

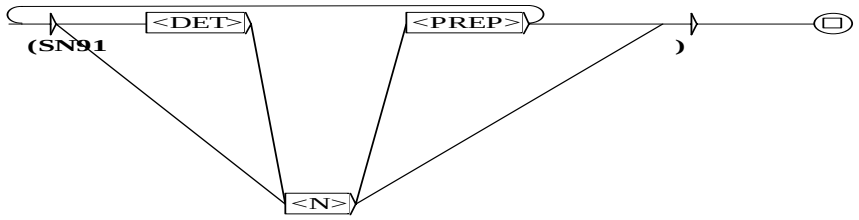


Fig. 6.102: FST gráfico que reconoce SSNN con un número ilimitado de SSPP incrustados

Si aplicáramos el FST SN_{91} a la secuencia lingüística: «*el sistema de recuperación de información del catálogo de la biblioteca de la Universidad de Granada...*» obtendríamos una sola equiparación:

1 match

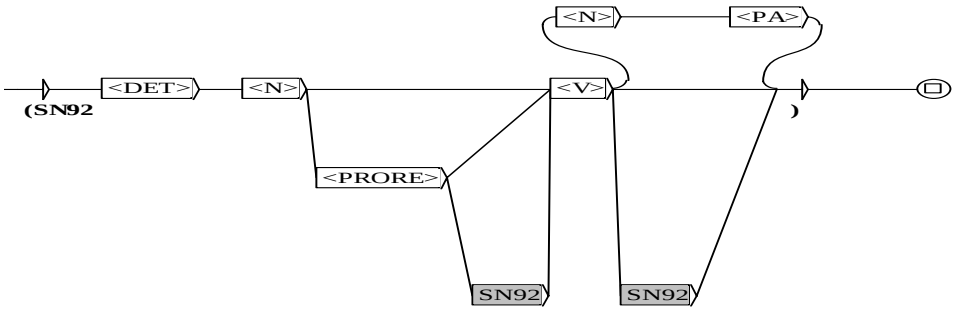
1:

(SN91 el sistema de(SN91 recuperación de(SN91 información de(SN91 el catálogo de(SN91 la biblioteca de(SN91 la Universidad de(SN91 Granada)

Como se puede comprobar, con este método es posible reconocer y agrupar los SSNN con un número ilimitado de SSPP incrustados sin ningún problema, y este es definitivamente el procedimiento que vamos a utilizar para el reconocimiento de este tipo de estructuras.

2. En los casos en los que se precise representar *SSNN con recursividad de constituyentes oracionales* sería necesario utilizar reglas que permitieran generar estructuras lingüísticas con *autoincrustación*, como $\mathbf{A} \rightarrow$ a **A** b. Este tipo de reglas pertenecen a las Gramáticas Libres de Contexto, o a las Gramáticas Sintagmáticas, y tampoco se pueden representar con mecanismos de estado-finito. Para representar y reconocer este tipo construcciones sintácticas las estructuras de los SSNN se trasladan a FST gráficos directamente recursivos que, aunque contengan *infinitas* referencias así mismo SN92, se pueden compilar con mecanismos de estado-finito porque, como hemos dicho, la interfaz gráfica permite limitar el número de estructuras *autoincrustadas a tres niveles* (Fig. 6.103). Las estructuras de algunos de estos SSNN, con un número limitado de *Oraciones de Relativo* (OR) autoincrustadas, se especifica de la forma siguiente:

$\mathbf{SN_{92}} \rightarrow$ **DET N** (**PRORE**) (**SN92**) **V** (**SN92**) (**N**) (**PA**)



SN92 (6.103).grf

Fig. 6.103: FST gráfico que reconoce SSNN con un número limitado de OR autoincrustadas

Si aplicáramos el FST SN_{92} para que intente reconocer la secuencia lingüística: «*el sistema que automatiza el catálogo que utiliza la biblioteca que la Universidad adquirió permite búsquedas avanzadas*» obtendríamos una sola equiparación:

1 match

1:

(SN92 el sistema que automatiza(SN92 el catálogo que utiliza(SN92 la biblioteca que(SN92 la Universidad adquirió) permite búsquedas avanzadas)))

Pero si aplicamos el mismo transductor a la secuencia: «*el sistema que automatiza el catálogo que incorpora imágenes que utiliza la biblioteca que la Universidad adquirió permite búsquedas avanzas...*» obtendríamos la siguiente equiparación :

2 matches

1:

(SN92 el sistema que automatiza(SN92 el catálogo que incorpora))

2:

imágenes que utiliza (SN92 la biblioteca que(SN92 la Universidad adquirió) permite búsquedas avanzadas)

Según el resultado anterior, el analizador sintáctico no es capaz de reconocer todas las *estructuras autoincrustadas*. Sin embargo, para la representación y reconocimiento de este tipo de construcciones sintagmáticas, ésta es la única solución que se puede adoptar porque en este caso no es posible crear Gramáticas Regulares equivalentes. No obstante, la representación que se obtiene con este procedimiento nos parece más que aceptable porque, en la práctica, los fenómenos

recursivos de las lenguas naturales nunca son de longitud infinita, aunque aparentemente la formulación de las reglas recursivas sí los puedan generar.

La consecuencia directa de esta primera aproximación es la siguiente: *1)* cuando los SSNN estén modificados por la *recursividad de SSPP* es posible crear Gramáticas Regulares, equivalentes a las Gramáticas Libres de Contexto; y *2)* cuando los SSNN estén modificados por la *recursividad de Oraciones de relativo* se van crear *transductores gráficos directamente recursivos*, aunque con un número limitado de transductores imbricados.

Aún así, aunque en el primer caso se puedan crear Gramáticas Regulares equivalentes, volvería a surgir el problema de que las gramáticas resultantes serían demasiado extensas, además de que con este procedimiento sólo obtendríamos representaciones *lineales* en las que no sería posible distinguir los sintagmas que modifican al núcleo nominal. Por esta razón, para la representación de este tipo de construcciones vamos a adoptar dos tratamientos:

A. Procedimiento para la representación de *SSNN con un número ilimitado de SSPP incrustados*:
    1. *Especificar* las estructuras complejas de los SSNN por medio de Expresiones Regulares, utilizando las Gramáticas Parciales desarrolladas previamente.
    2. *Trasladar* las estructuras directamente a *transductores Gráficos*.
    3. *Compilar* los transductores gráficos en *Transductores de Estado-Finito Deterministas* (FST).
    4. *Minimizar* los *Transductores de Estado-Finito Deterministas.*
    5. *Obtener los FST* que se encarguen de insertar marcas a las variantes estructurales de los SSNN especificados.

B. Procedimiento para la representación de *SSNN con un número ilimitado de Oraciones de relativo autoincrustadas*:
   1. *Especificar* las estructuras complejas de los SSNN utilizando las *Gramáticas Parciales* desarrolladas previamente.
   2. *Trasladar las estructuras anidadas* directamente a *transductores gráficos autorreferenciales*.
   3. *Compilar* los transductores gráficos en *Transductores de Estado-Finito Deterministas* (FST), en los que se *limita el número de estructuras anidadas* restringiendo el número de FST imbricados.
   4. *Minimizar los Transductores de Estado-Finito Deterministas*.
   5. Obtener los *FST directamente recursivos* que se encarguen de insertar marcas a las variantes estructurales de los SSNN especificados.

En los siguientes apartados se va a desarrollar esta metodología, volviéndose a poner de manifiesto que las Gramáticas Regulares son capaces de generar la mayoría de las estructuras sintagmáticas, excepto las que tienen un número ilimitado de constituyentes oracionales.

6.2.2.1. SSNN con recursividad de constituyentes preposicionales

La construcción de los analizadores de SSNN modificados por un número ilimitado de SSPP incrustados, se establece, según la metodología planteada arriba, del modo siguiente:

1. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{93} (Fig. 6.104):

$$SN_{93} \rightarrow SN_{55}\ \text{PREP}\ SN_{93}$$

ER_0 = N (PREP N)*
N N (PREP N N)*
DET N (PREP DET N)*
. . .

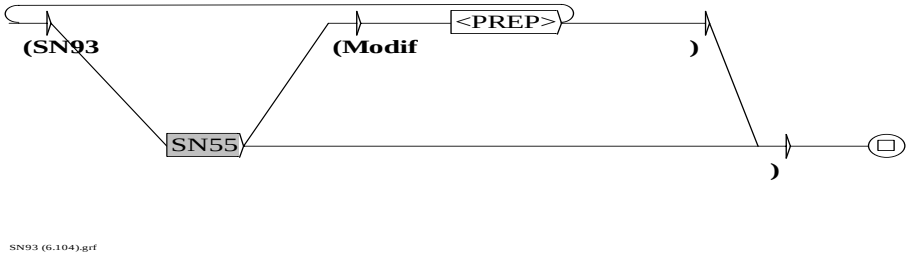


Fig. 6.104: FST gráfico que agrupa las variantes de la estructura recursiva SN_{93}

2. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{94} (Fig. 6.105):

$SN_{94} \rightarrow SN_{55}$ PREP (SN_{56}) SN_{94}

ER_0 = N (PREP (N A) N)*
N N (PREP (N A) N N)*
DET N (PREP (A N) DET N)*
. . .

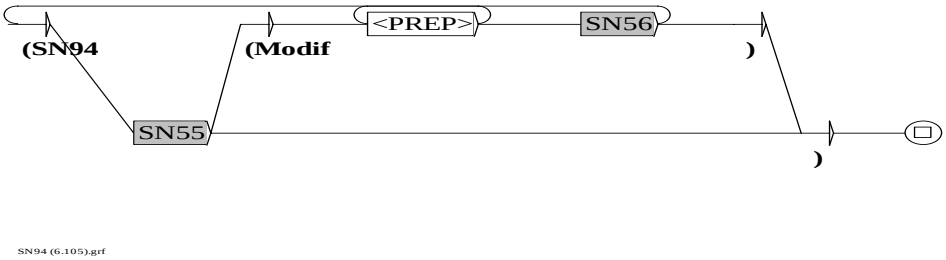


Fig. 6.105: FST gráfico que agrupa las variantes de la estructura recursiva SN_{94}

3. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{95} (Fig. 6.106):

$$SN_{95} \rightarrow SN_{56}\ PREP\ (SN_{56})\ SN_{95}$$

$$ER_0 = \underline{N\ A}\ (PREP\ (\underline{N\ A})\ \underline{N\ A})^*$$
$$\underline{A\ N}\ (PREP\ (\underline{A\ N})\ \underline{A\ N}\ )^*$$
$$\underline{N\ PA}\ (PREP\ (\underline{N\ PA})\ \underline{N\ PA}\ )^*$$
$$\ldots$$

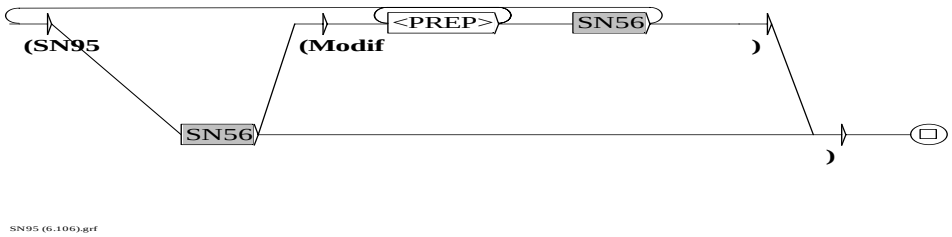


Fig. 6.106: FST gráfico que agrupa las variantes de la estructura recursiva SN_{95}

4. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{96} (Fig. 6.107):

$$SN_{96} \rightarrow SN_{56}\ \mathbf{PREP}\ (SN_{55})\ SN_{96}$$

$$ER_0 = \underline{N\ A}\ (PREP\ (\underline{N})\ \underline{N\ A})^*$$
$$\underline{A\ N}\ (PREP\ (\underline{N})\ \underline{A\ N}\ )^*$$
$$\underline{N\ PA}\ (PREP\ (\underline{N})\ \underline{N\ PA}\ )^*$$
$$\ldots$$

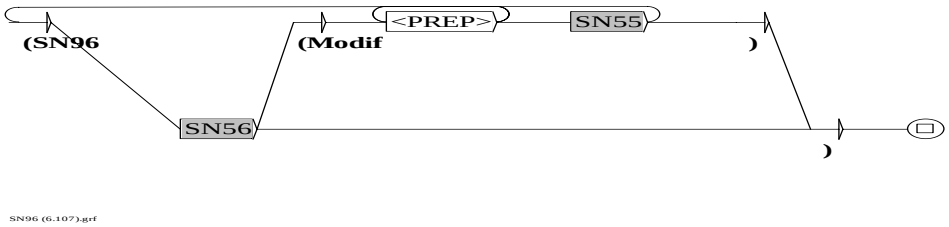


Fig. 6.107: FST gráfico que agrupa las variantes de la estructura recursiva SN_{96}

5. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{97} con iteración de constituyentes (Fig. 6.108):

$$SN_{97} \rightarrow SN_{57}\ \mathbf{PREP}\ (SN_{57})\ SN_{97}$$

$$ER_0 = \underline{DET\ N^+}\ \left(PREP\ \left(\underline{DET\ N^+}\right)\ \underline{DET\ N^+}\right)^*$$
$$\underline{N^+}\ \left(PREP\ \left(\underline{N^+}\right)\ \underline{N^+}\ \right)^*$$
$$\underline{CUANT\ N^+}\ \left(PREP\ \left(\underline{CUANT\ N^+}\right)\ \underline{CUANT\ N^+}\ \right)^*$$
$$\ldots$$

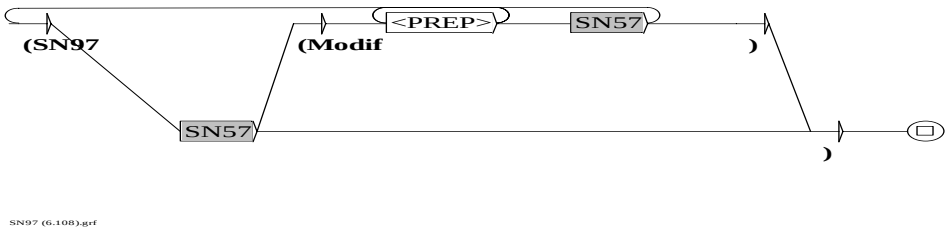


Fig. 6.108: FST gráfico que agrupa las variantes de la estructura recursiva SN_{97}

6. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{98} con iteración de constituyentes (Fig. 6.109):

$$\begin{aligned}
\mathbf{SN_{98}} \rightarrow\ & \mathbf{SN_{57}}\ \mathbf{PREP}\ (\mathbf{SN_{58}})\ \mathbf{SN_{98}} \\
& \mathbf{SN_{57}}\ \mathbf{PREP}\ (\mathbf{SN_{59}})\ \mathbf{SN_{98}} \\
& \mathbf{SN_{57}}\ \mathbf{PREP}\ (\mathbf{SN_{60}})\ \mathbf{SN_{98}} \\
& \ldots
\end{aligned}$$

$$\begin{aligned}
ER_0 = & \underline{\text{DET N}^+} \left(\text{PREP} \left(\underline{\text{DET N}^+\ \text{A}} \right) \underline{\text{DET N}^+} \right)^* \\
& \underline{\text{N}^+} \left(\text{PREP} \left(\underline{\text{DET N A}^+} \right) \underline{\text{N}^+} \right)^* \\
& \underline{\text{CUANT N}^+} \left(\text{PREP} \left(\underline{\text{DET N}^+\ \text{A}^+} \right) \underline{\text{CUANT N}^+} \right)^* \\
& \ldots
\end{aligned}$$

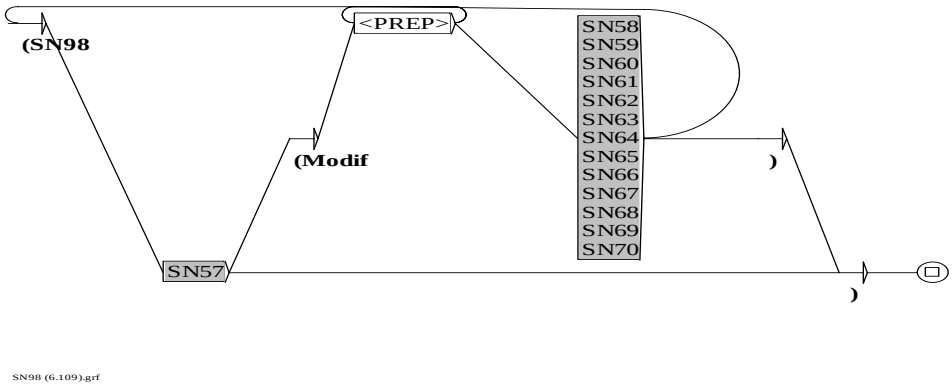


Fig. 6.109: FST gráfico que agrupa las variantes de la estructura recursiva SN_{98}

7. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{99} con iteración de constituyentes (Fig. 6.110):

$$SN_{99} \rightarrow SN_{58}\ PREP\ (SN_{58})\ SN_{99}$$
$$SN_{58}\ PREP\ (SN_{59})\ SN_{99}$$
$$SN_{58}\ PREP\ (SN_{60})\ SN_{99}$$
...

$$ER_0 = \underline{DET\ N^+\ A}\ \left(PREP\ \left(\underline{DET\ N^+\ A}\right)\ \underline{DET\ N^+}\right)^*$$
$$\underline{DET\ N^+\ A}\ \left(PREP\ \left(\underline{DET\ N\ A^+}\right)\ \underline{DET\ N^+}\ \right)^*$$
$$\underline{DET\ N^+\ A}\ \left(PREP\ \left(\underline{DET\ N^+\ A^+}\right)\ \underline{DET\ N^+}\ \right)^*$$
...

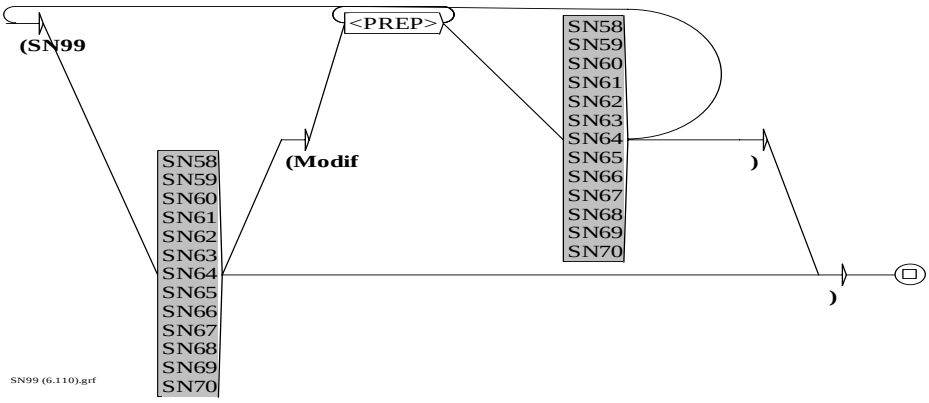


Fig. 6.110: FST gráfico que agrupa las variantes de la estructura recursiva SN_{99}

8. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{100} con iteración de constituyentes (Fig. 6.111):

$$SN_{100} \rightarrow SN_{58}\ PREP\ (SN_{57})\ SN_{100}$$
$$SN_{59}\ PREP\ (SN_{57})\ SN_{100}$$
$$SN_{60}\ PREP\ (SN_{57})\ SN_{100}$$
...

$$ER_0 = \underline{\text{DET N}^+ \text{ A}} \left(\text{PREP} \left(\underline{\text{DET N}^+} \right) \underline{\text{DET N}^+ \text{ A}} \right)^*$$
$$\underline{\text{DET N A}^+} \left(\text{PREP} \left(\underline{\text{DET N}^+} \right) \underline{\text{DET N A}^+} \right)^*$$
$$\underline{\text{DET N}^+ \text{ A}^+} \left(\text{PREP} \left(\underline{\text{DET N}^+} \right) \underline{\text{DET N}^+ \text{ A}^+} \right)^*$$
$$\ldots$$

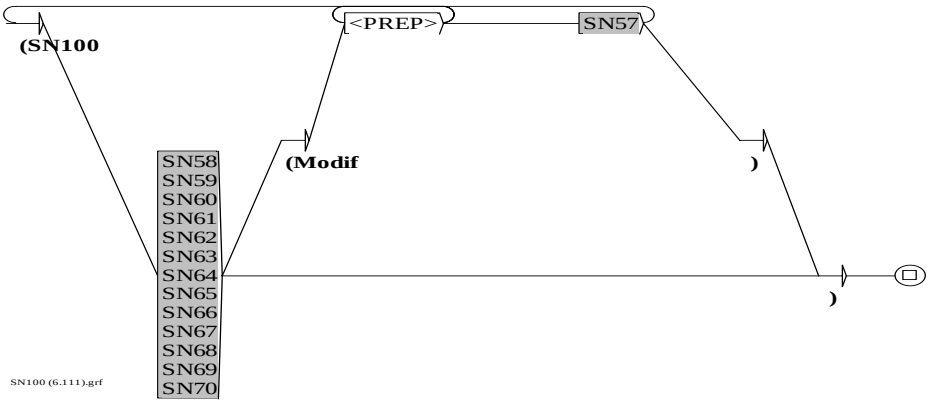


Fig. 6.111: FST gráfico que agrupa las variantes de la estructura recursiva SN_{100}

9. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{101}, modificada por una *Oración de relativo*, con recursividad de constituyentes preposicionales (Fig. 6.112):

$$\mathbf{SN_{101} \rightarrow SN_{93}\ \ PRORE\ V\ SN_{93}}$$
$$\mathbf{SN_{93}\ \ PRORE\ SN_{93}\ V}$$
$$\mathbf{SN_{93}\ \ ARE\ SN_{93}\ V}$$
$$\ldots$$

$$ER_0 = \underline{\text{N } (\text{PREP N})^*} \text{ PRORE V } \underline{\text{N } (\text{PREP N})^*}$$
$$\underline{\text{N N } (\text{PREP N N})^*} \text{ PRORE V } \underline{\text{N N } (\text{PREP N N})^*}$$
$$\underline{\text{DET N } (\text{PREP DET N})^*} \text{ ARE V } \underline{\text{DET N } (\text{PREP DET N})^*}$$
$$\ldots$$

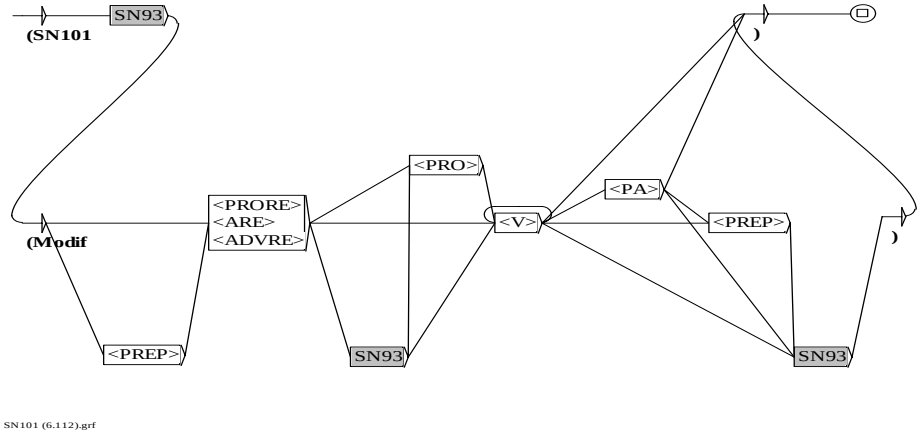


Fig. 6.112: FST gráfico que agrupa las variantes de la estructura recursiva SN_{101}

10. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{102}, modificada por una *Oración de relativo*, con recursividad de constituyentes preposicionales (Fig. 6.113):

$$\begin{aligned} \mathbf{SN_{102}} \rightarrow\ & \mathbf{SN_{94}}\ \ \mathbf{PRORE\ V\ SN_{94}} \\ & \mathbf{SN_{94}}\ \ \mathbf{PRORE\ SN_{94}\ V} \\ & \mathbf{SN_{94}}\ \ \mathbf{ARE\ SN_{94}\ V} \\ & \mathbf{\ldots} \end{aligned}$$

$$\begin{aligned} ER_0 = {} & \underline{N\ (PREP\ (N\ A)\ N)^*}\ PRORE\ V\ \underline{N\ (PREP\ (N\ A)\ N)^*} \\ & \underline{N\ N\ (PREP\ (N\ A)\ N\ N)^*}\ PRORE\ V\ \underline{N\ N\ (PREP\ (N\ A)\ N\ N)^*} \\ & .\underline{DET\ N\ (PREP\ (N\ A)\ DET\ N)^*}\ PRORE\ V\ \underline{DET\ N\ (PREP\ (N\ A)\ DET\ N)^*} \\ & \ldots \end{aligned}$$

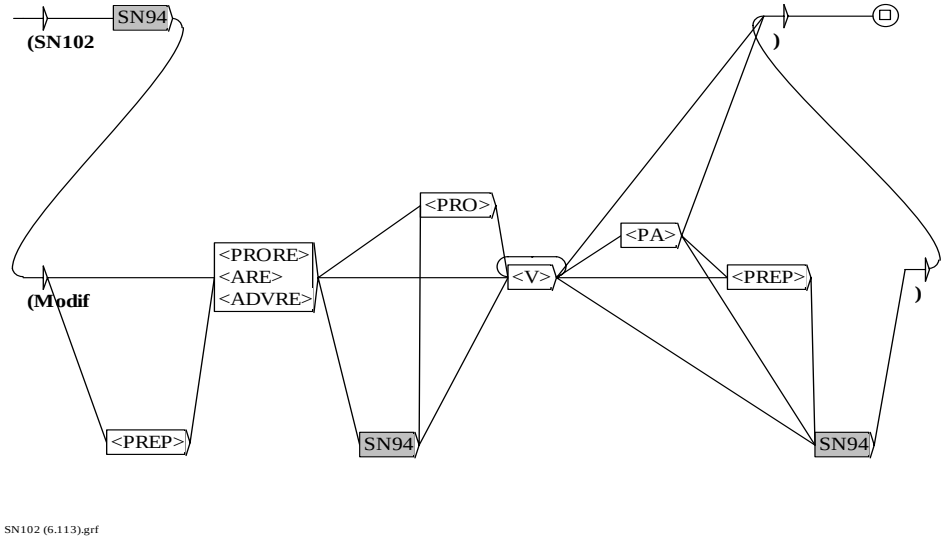


Fig. 6.113: FST gráfico que agrupa las variantes de la estructura recursiva SN_{102}

11. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{103}, modificada por una *Oración de relativo*, con recursividad de constituyentes preposicionales (Fig. 6.114):

$$\begin{aligned} \mathbf{SN_{103}} \rightarrow\ & \mathbf{SN_{95}}\ \ \mathbf{PRORE\ V\ SN_{95}} \\ & \mathbf{SN_{95}}\ \ \mathbf{PRORE\ SN_{95}\ V} \\ & \mathbf{SN_{95}}\ \ \mathbf{ARE\ SN_{95}\ V} \\ & \ldots \end{aligned}$$

$$\begin{aligned} ER_0 = {} & \underline{\mathrm{N\ A\ (PREP\ (N\ A)\ N\ A)^*}}\ \mathrm{PRORE\ V}\ \underline{\mathrm{N\ A\ (PREP\ (N\ A)\ N\ A)^*}} \\ & \underline{\mathrm{A\ N\ (PREP\ (A\ N)\ A\ N)^*}}\ \mathrm{PRORE\ V}\ \underline{\mathrm{A\ N\ (PREP\ (A\ N)\ A\ N)^*}} \\ & \underline{\mathrm{N\ PA\ (PREP\ (N\ PA)\ N\ PA)^*}}\ \mathrm{PRORE\ V}\ \underline{\mathrm{N\ PA\ (PREP\ (N\ PA)\ N\ PA)^*}} \\ & \ldots \end{aligned}$$

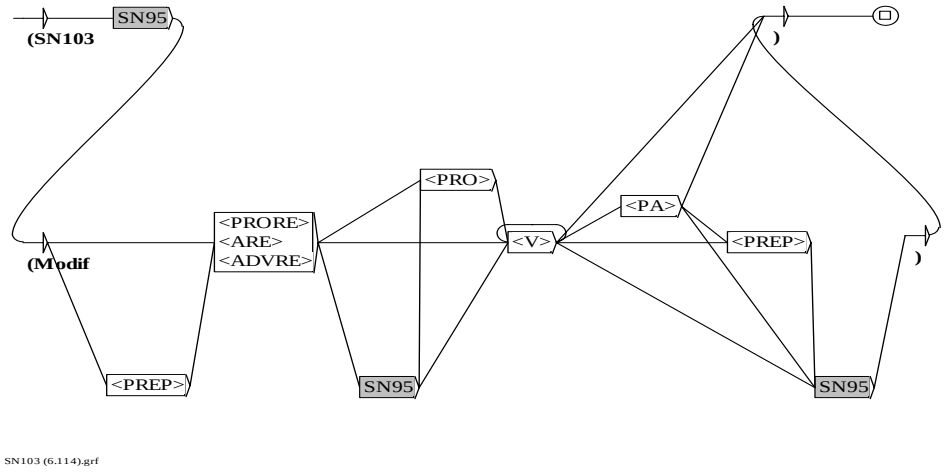


Fig. 6.114: FST gráfico que agrupa las variantes de la estructura recursiva SN_{103}

12. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{104}, modificada por una *Oración de relativo*, con recursividad de constituyentes preposicionales (Fig. 6.115):

$$\begin{aligned} \mathbf{SN_{104}} \rightarrow\ & \mathbf{SN_{96}}\ \ \mathbf{PRORE\ V\ SN_{96}} \\ & \mathbf{SN_{96}}\ \ \mathbf{PRORE\ SN_{96}\ V} \\ & \mathbf{SN_{96}}\ \ \mathbf{ARE\ SN_{96}\ V} \\ & \ldots \end{aligned}$$

$$\begin{aligned} ER_0 = {} & \underline{\mathrm{N\ A\ (PREP\ (N)\ N\ A)^*}}\ \mathrm{PRORE\ V}\ \underline{\mathrm{N\ A\ (PREP\ (N)\ N\ A)^*}} \\ & \underline{\mathrm{A\ N\ (PREP\ (N)\ A\ N)^*}}\ \mathrm{PRORE\ V}\ \underline{\mathrm{A\ N\ (PREP\ (N)\ A\ N)^*}} \\ & \underline{\mathrm{N\ PA\ (PREP\ (N)\ N\ PA)^*}}\ \mathrm{PRORE\ V}\ \underline{\mathrm{N\ PA\ (PREP\ (N)\ N\ PA)^*}} \\ & \ldots \end{aligned}$$

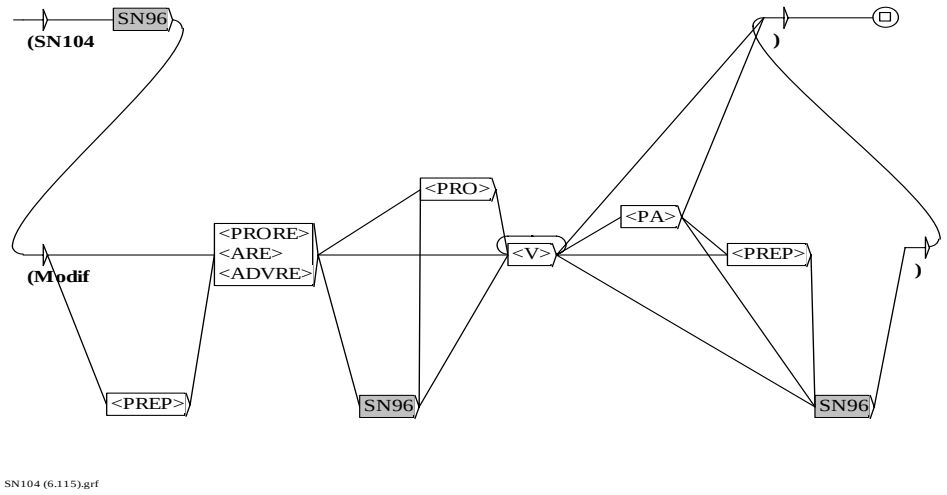


Fig. 6.115: FST gráfico que agrupa las variantes de la estructura recursiva SN_{104}

13. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{105}, modificada por una *Oración de relativo*, con iteración de constituyentes y recursividad de constituyentes preposicionales (Fig. 6.116):

$$SN_{105} \rightarrow SN_{97}\ \text{PRORE V}\ SN_{97}$$
$$SN_{97}\ \text{PRORE}\ SN_{97}\ \text{V}$$
$$SN_{97}\ \text{ARE}\ SN_{97}\ \text{V}$$
$$\ldots$$

$$ER_0 = \underline{\text{DET N}^+ \left(\text{PREP}\left(\text{DET N}^+\right)\text{DET N}^+\right)^*}\ \text{PRORE V}\ \underline{\text{DET N}^+ \left(\text{PREP}\left(\text{DET N}^+\right)\text{DET N}^+\right)^*}$$
$$\underline{\text{N}^+ \left(\text{PREP}\left(\text{N}^+\right)\text{N}^+\right)^*}\ \text{PRORE V}\ \underline{\text{N}^+ \left(\text{PREP}\left(\text{N}^+\right)\text{N}^+\right)^*}$$
$$\underline{\text{CUANT N}^+ \left(\text{PREP}\left(\text{CUANT N}^+\right)\text{CUANT N}^+\right)^*}\ \text{PRORE V}\ \underline{\text{CUANT N}^+ \left(\text{PREP}\left(\text{CUANT N}^+\right)\text{CUANT N}^+\right)^*}$$
$$\ldots$$

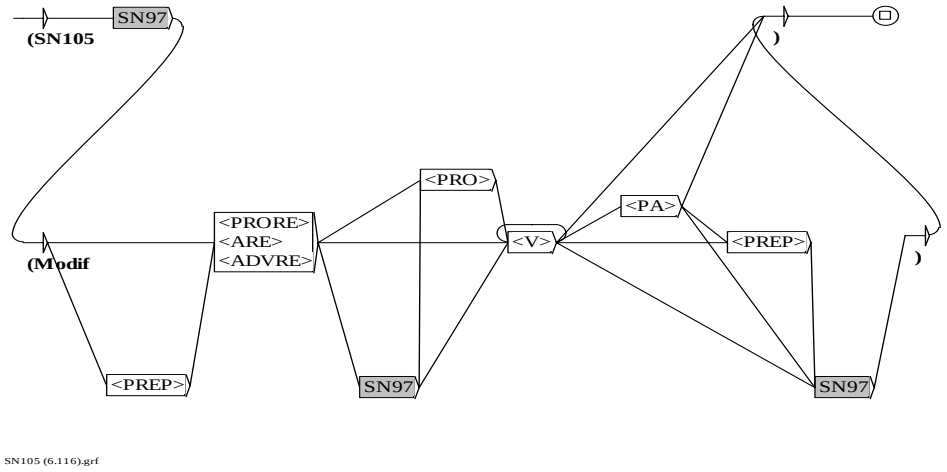


Fig. 6.116: FST gráfico que agrupa las variantes de la estructura recursiva SN_{105}

14. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{106}, modificada por una *Oración de relativo*, con iteración de constituyentes y recursividad de constituyentes preposicionales (Fig. 6.117):

$$\mathbf{SN_{106} \rightarrow SN_{98}\ \ PRORE\ V\ SN_{98}}$$
$$\mathbf{SN_{98}\ \ PRORE\ SN_{98}\ V}$$
$$\mathbf{SN_{98}\ \ ARE\ SN_{98}\ V}$$
$$\ldots$$

$$ER_0 = \underline{\text{DET N}^+ \left(\text{PREP}\left(\text{DET N}^+\ \text{A}\right)\text{DET N}^+\right)^*}\ \text{PRORE V}\ \underline{\text{DET N}^+ \left(\text{PREP}\left(\text{DET N}^+\ \text{A}\right)\text{DET N}^+\right)^*}$$
$$\underline{\text{N}^+ \left(\text{PREP}\left(\text{N}^+\ \text{A}\right)\text{N}^+\right)^*}\ \text{PRORE V}\ \underline{\text{N}^+ \left(\text{PREP}\left(\text{N}^+\ \text{A}\right)\text{N}^+\right)^*}$$
$$\underline{\text{CUANT N}^+ \left(\text{PREP}\left(\text{N}^+\ \text{A}\right)\text{CUANT N}^+\right)^*}\ \text{PRORE V}\ \underline{\text{CUANT N}^+ \left(\text{PREP}\left(\text{N}^+\ \text{A}\right)\text{CUANT N}^+\right)^*}$$
$$\ldots$$

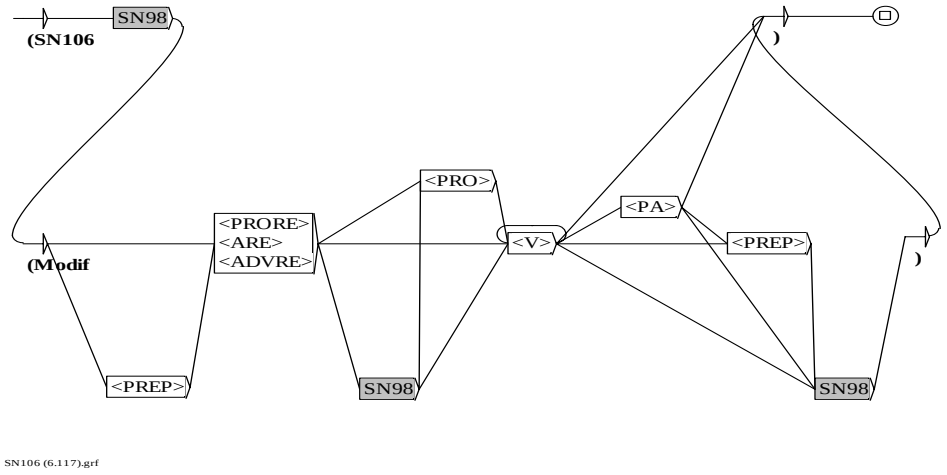


Fig. 6.117: FST gráfico que agrupa las variantes de la estructura recursiva SN_{106}

15. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{107}, modificada por una *Oración de relativo*, con iteración de constituyentes y recursividad de constituyentes preposicionales (Fig. 6.118):

$$\mathbf{SN_{107} \rightarrow SN_{99}\ \ PRORE\ V\ SN_{99}}$$
$$\mathbf{SN_{99}\ \ PRORE\ SN_{99}\ V}$$
$$\mathbf{SN_{99}\ \ ARE\ SN_{99}\ V}$$
$$\ldots$$

$$ER_0 = \underline{\text{DET N}^+ \text{ A } \left(\text{PREP } \left(\text{DET N}^+ \text{ A}\right) \text{ DET N}^+\right)^*} \text{ PRORE V } \underline{\text{DET N}^+ \text{ A } \left(\text{PREP } \left(\text{DET N}^+ \text{ A}\right) \text{ DET N}^+\right)^*}$$
$$\underline{\text{N}^+ \text{ A } \left(\text{PREP } \left(\text{N}^+ \text{ A}\right) \text{ N}^+ \text{ A}\right)^*} \text{ PRORE V } \underline{\text{N}^+ \text{ A } \left(\text{PREP } \left(\text{N}^+ \text{ A}\right) \text{ N}^+ \text{ A}\right)^*}$$
$$\underline{\text{N}^+ \text{ A}^+ \left(\text{PREP } \left(\text{N}^+ \text{ A}^+\right) \text{ N}^+ \text{ A}^+\right)^*} \text{ PRORE V } \underline{\text{N}^+ \text{ A}^+ \left(\text{PREP } \left(\text{N}^+ \text{ A}^+\right) \text{ N}^+ \text{ A}^+\right)^*}$$
$$\ldots$$

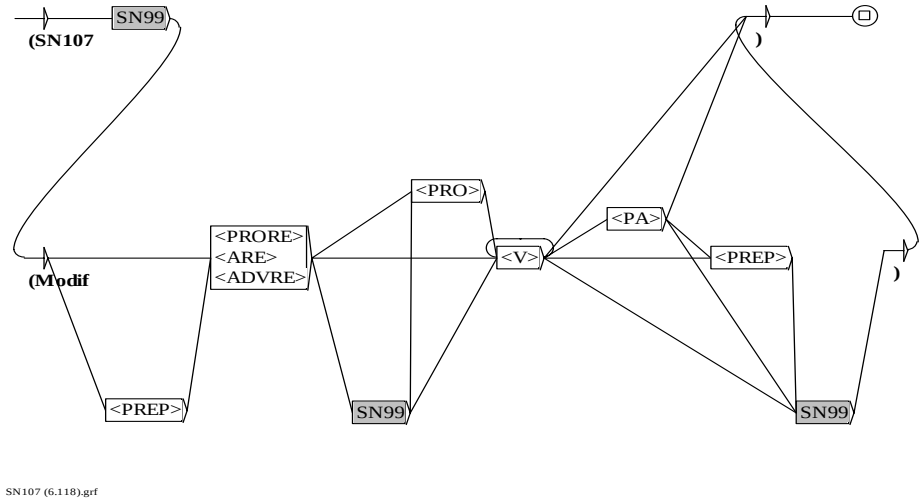


Fig. 6.118: FST gráfico que agrupa las variantes de la estructura recursiva SN_{107}

16. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{108}, modificada por una *Oración de relativo*, con iteración de constituyentes y recursividad de constituyentes preposicionales (Fig. 6.119):

$$\mathbf{SN_{108} \rightarrow SN_{100}\ PRORE\ V\ SN_{100}}$$
$$\mathbf{SN_{100}\ PRORE\ SN_{100}\ V}$$
$$\mathbf{SN_{100}\ ARE\ SN_{100}\ V}$$
$$\ldots$$

$$ER_0 = \underline{\text{DET N}^+ \text{ A } \left(\text{PREP } \left(\text{DET N}^+\right) \text{ DET N}^+ \text{ A}\right)^*}\ \text{PRORE V } \underline{\text{DET N}^+ \text{ A } \left(\text{PREP } \left(\text{DET N}^+\right) \text{ DET N}^+ \text{ A}\right)^*}$$
$$\underline{\text{DET N A}^+ \left(\text{PREP } \left(\text{DET N}^+\right) \text{ DET N A}^+\right)^*}\ \text{PRORE V } \underline{\text{DET N A}^+ \left(\text{PREP } \left(\text{DET N}^+\right) \text{ DET N A}^+\right)^*}$$
$$\underline{\text{DET N}^+ \text{ A}^+ \left(\text{PREP } \left(\text{DET N}^+\right) \text{ DET N}^+ \text{ A}^+\right)^*}\ \text{PRORE V } \underline{\text{DET N}^+ \text{ A}^+ \left(\text{PREP } \left(\text{DET N}^+\right) \text{ DET N}^+ \text{ A}^+\right)^*}$$
$$\ldots$$

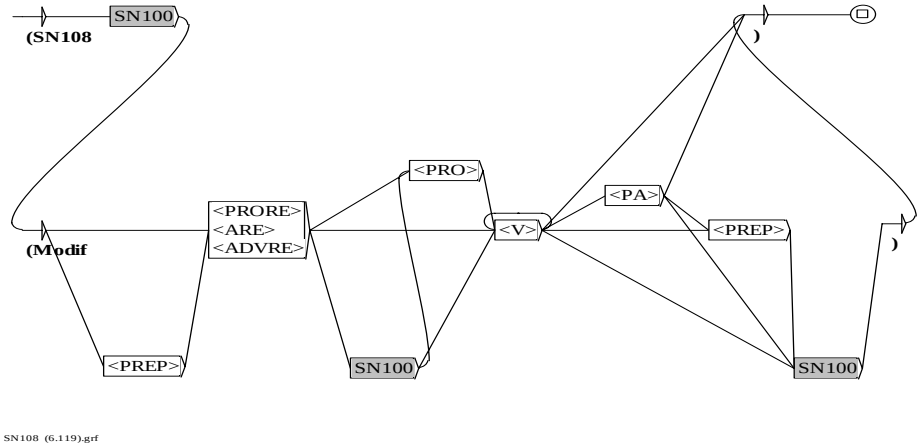


Fig. 6.119: FST gráfico que agrupa las variantes de la estructura recursiva SN_{108}

6.2.2.2. SSNN con recursividad de constituyentes oracionales

La representación de SSNN *modificados por la recursividad de Oraciones de relativo se va a realizar por medio de transductores gráficos autorreferenciales*, aunque sólo permitan reconocer un número limitado de estructuras recursivas. A su vez, para la construcción de este tipo de analizadores de SSNN vamos a utilizar las Gramáticas Parciales ya elaboradas por medio de *transductores gráficos imbricados*. Aplicando la metodología expuesta anteriormente, la construcción de los analizadores de tales estructuras complejas se establece del modo siguiente:

1. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{109} con recursividad de constituyentes oracionales (Fig. 6.120):

SN_{109} → SN_{55} PRORE V (SN_{109}**)**
SN_{55} PRORE (SN_{109}**) V**
SN_{55} PRORE (SN_{109}**) PRO V PA**
...

ER_0 = N PRORE V (SN_{109})
N N PRORE (SN_{109}) V
DET N PRORE (SN_{109}) PRO V PA
...

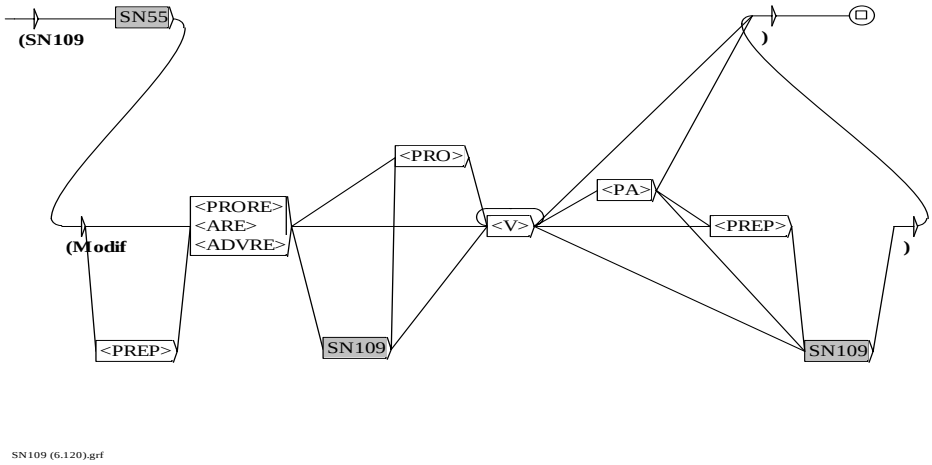


Fig. 6.120: FST gráfico que agrupa las variantes de la estructura recursiva SN_{109}

2. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{110} con recursividad de constituyentes oracionales (Fig. 6.121):

SN_{110} → SN_{56} PRORE V (SN_{110}**)**
SN_{56} PRORE (SN_{110}**) V**
SN_{56} PRORE (SN_{110}**) PRO V PA**
...

ER_0 = N A PRORE V (SN_{110})
A N PRORE (SN_{110}) V
N PA PRORE (SN_{110}) PRO V PA
...

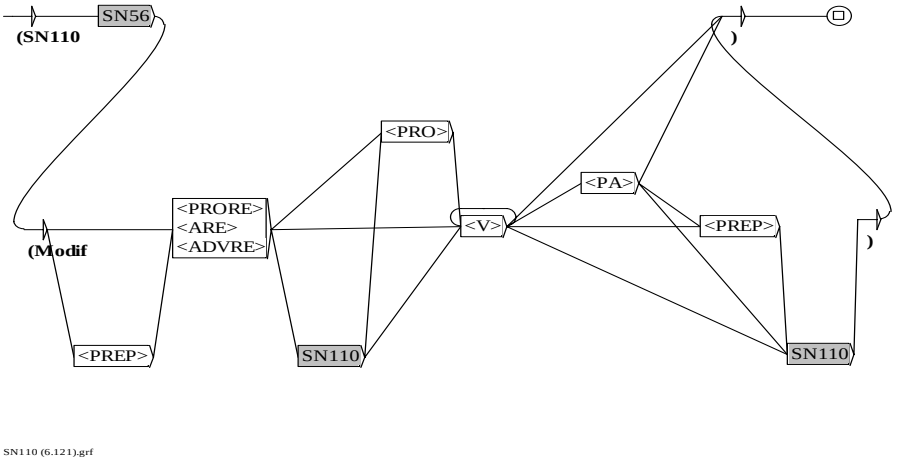


Fig. 6.121: FST gráfico que agrupa las variantes de la estructura recursiva SN_{110}

3. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{111} con iteración de constituyentes y recursividad de constituyentes oracionales (Fig. 6.122):

$$\mathbf{SN_{111} \rightarrow SN_{57}\ PRORE\ V\ (SN_{111})}$$
$$\mathbf{SN_{57}\ PRORE\ (SN_{111})\ V}$$
$$\mathbf{SN_{57}\ PRORE\ (SN_{111})\ PRO\ V\ PA}$$
$$\mathbf{\ldots}$$

$$ER_0 = \underline{DET\ N^+}\ PRORE\ V\ (SN_{111})$$
$$\underline{N^+}\ PRORE\ (SN_{111})\ V$$
$$\underline{CUANT\ N^+}\ PRORE\ (SN_{111})\ PRO\ V\ PA$$
$$\ldots$$

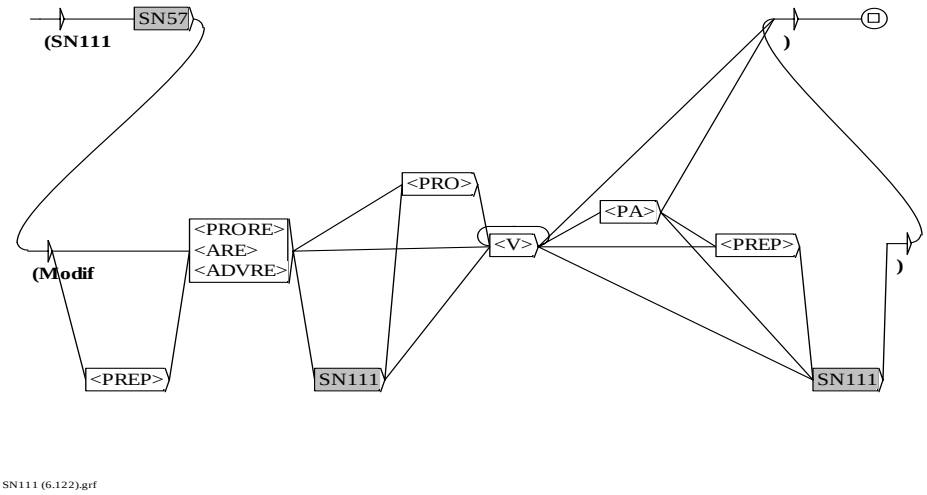


Fig. 6.122: FST gráfico que agrupa las variantes de la estructura recursiva SN_{111}

4. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{112} con iteración de constituyentes y recursividad de constituyentes oracionales (Fig. 6.123):

$$\mathbf{SN_{112}} \rightarrow \mathbf{SN_{58}\ PRORE\ V\ (SN_{112})}$$
$$\mathbf{SN_{59}\ PRORE\ (SN_{112})\ V}$$
$$\mathbf{SN_{60}\ PRORE\ (SN_{112})\ PRO\ V\ PA}$$
$$\ldots$$

$$ER_0 = \underline{DET\ N^+\ A}\ PRORE\ V\ (SN_{112})$$
$$\underline{DET\ N\ A^+}\ PRORE\ (SN_{112})\ V$$
$$\underline{DET\ N^+\ A^+}\ PRERE\ (SN_{112})\ PRO\ V\ PA$$
$$\ldots$$

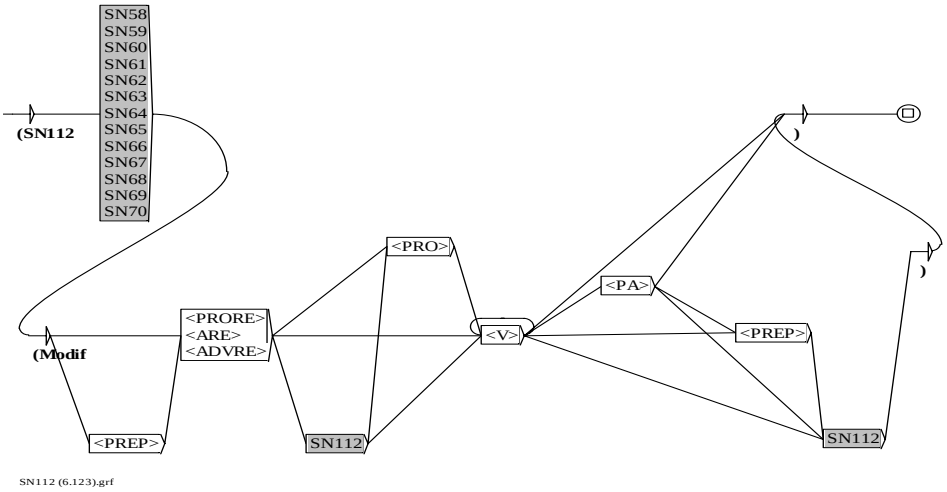


Fig. 6.123: FST gráfico que agrupa las variantes de la estructura recursiva SN_{112}

5. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{113} con iteración de constituyentes y recursividad de constituyentes oracionales (Fig. 6.124):

$$SN_{113} \rightarrow SN_{55}\ \text{PREP}\ SN_{55}\ \text{PRORE}\ \text{V}\ (SN_{113})$$
$$SN_{55}\ \text{PREP}\ SN_{55}\ \text{PRORE}\ (SN_{113})\ \text{V}$$
$$SN_{55}\ \text{PREP}\ SN_{56}\ \text{PRORE}\ \text{V}\ (SN_{113})$$
$$\ldots$$

$$ER_0 = \underline{\text{N}}\ \text{PREP}\ \underline{\text{N}}\ \text{PRORE}\ \text{V}\ (SN_{113})$$
$$\underline{\text{N N}}\ \text{PRERP}\ \underline{\text{N N}}\ \text{PRORE}\ (SN_{113})\ \text{V}$$
$$\underline{\text{DET N}}\ \text{PREP}\ \underline{\text{A N}}\ \text{PRORE}\ \text{V}\ (SN_{113})$$
$$\ldots$$

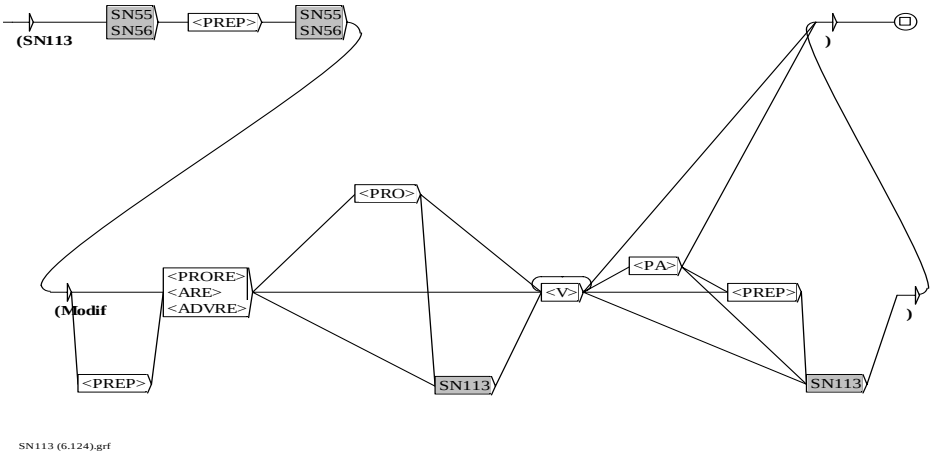


Fig. 6.124: FST gráfico que agrupa las variantes de la estructura recursiva SN_{113}

6. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{114} con iteración de constituyentes y recursividad de constituyentes oracionales (Fig. 6.125):

$$\mathbf{SN_{114} \rightarrow SN_{57}\ PREP\ SN_{57}\ PRORE\ V\ (SN_{114})}$$
$$\mathbf{SN_{57}\ PREP\ SN_{57}\ PRORE\ (SN_{114})\ V}$$
$$\mathbf{SN_{57}\ PREP\ SN_{57}\ PRORE\ PRO\ V\ PA\ (SN_{114})}$$
$$\mathbf{\ldots}$$

$$ER_0 = \underline{DET\ N^+}\ PREP\ \underline{DET\ N^+}\ PRORE\ V\ (SN_{114})$$
$$\underline{N^+}\ PREP\ \underline{N^+}\ PRORE\ (SN_{114})\ V$$
$$\underline{CUANT\ N^+}\ PREP\ \underline{CUANT\ N^+}\ PRORE\ PRO\ V\ PA\ (SN_{114})$$
$$\ldots$$

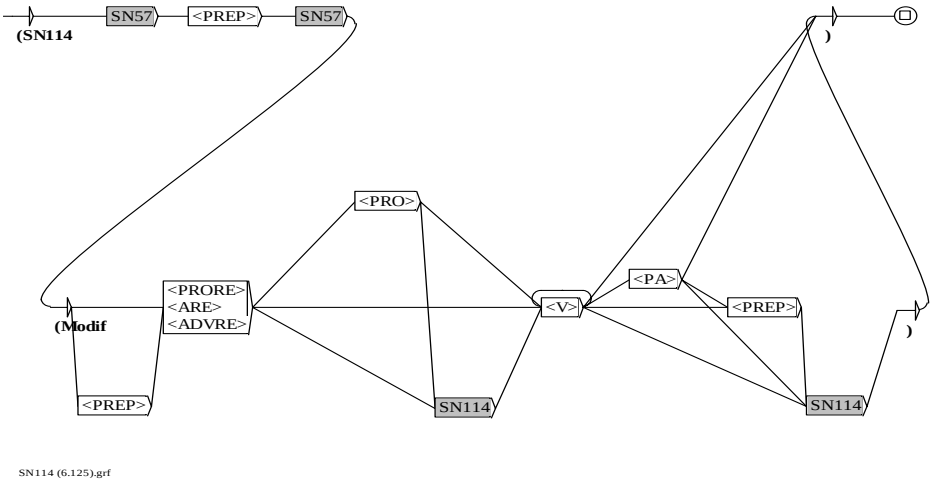


Fig. 6.125: FST gráfico que agrupa las variantes de la estructura recursiva SN_{114}

7. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{115} con iteración de constituyentes y recursividad de constituyentes oracionales (Fig. 6.126):

$$\mathbf{SN_{115}} \rightarrow \mathbf{SN_{57}\ PREP\ SN_{58}\ PRORE\ V\ (SN_{115})}$$
$$\mathbf{SN_{57}\ PREP\ SN_{59}\ PRORE\ V\ (SN_{115})}$$
$$\mathbf{SN_{57}\ PREP\ SN_{60}\ PRORE\ V\ (SN_{115})}$$
$$\ldots$$

$$ER_0 = \underline{DET\ N^+}\ PREP\ \underline{DET\ N^+\ A}\ PRORE\ V\ (SN_{115})$$
$$\underline{N^+}\ PREP\ \underline{DET\ N\ A^+}\ PRORE\ V\ (SN_{115})$$
$$\underline{CUANT\ N^+}\ PREP\ \underline{DET\ N^+\ A^+}\ PRORE\ V\ (SN_{115})$$
$$\ldots$$

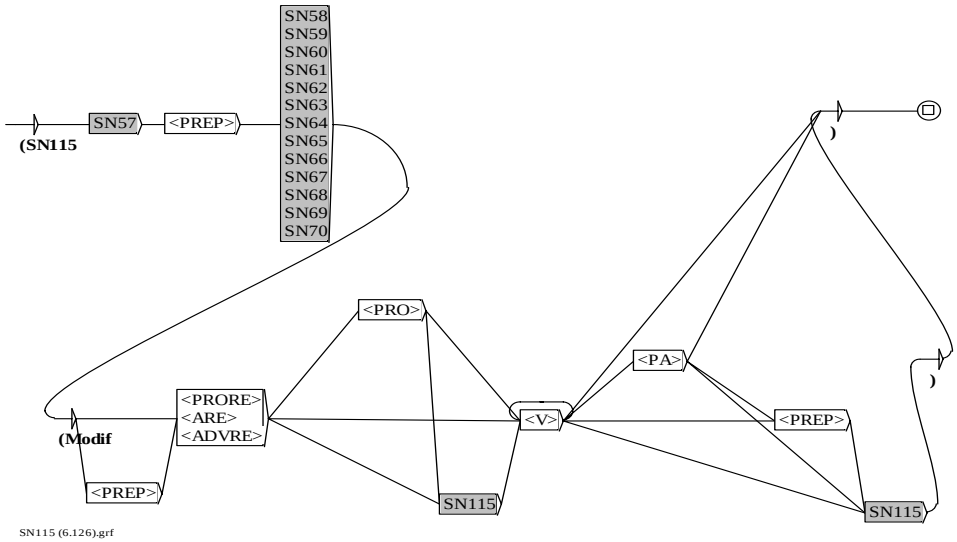


Fig. 6.126: FST gráfico que agrupa las variantes de la estructura recursiva SN_{115}

8. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{116} con iteración de constituyentes y recursividad de constituyentes oracionales (Fig. 6.127):

$$\mathbf{SN_{116}} \rightarrow \mathbf{SN_{58}\ PREP\ SN_{58}\ PRORE\ V\ (SN_{116})}$$
$$\mathbf{SN_{58}\ PREP\ SN_{59}\ PRORE\ V\ (SN_{116})}$$
$$\mathbf{SN_{58}\ PREP\ SN_{60}\ PRORE\ V\ (SN_{116})}$$
$$\ldots$$

$$ER_0 = \underline{DET\ N^+\ A}\ PREP\ \underline{DET\ N^+\ A}\ PRORE\ V\ (SN_{116})$$
$$\underline{DET\ N^+\ A}\ PREP\ \underline{DET\ N\ A^+}\ PRORE\ V\ (SN_{116})$$
$$\underline{DET\ N^+\ A}\ PREP\ \underline{DET\ N^+\ A^+}\ PRORE\ V\ (SN_{116})$$
$$\ldots$$

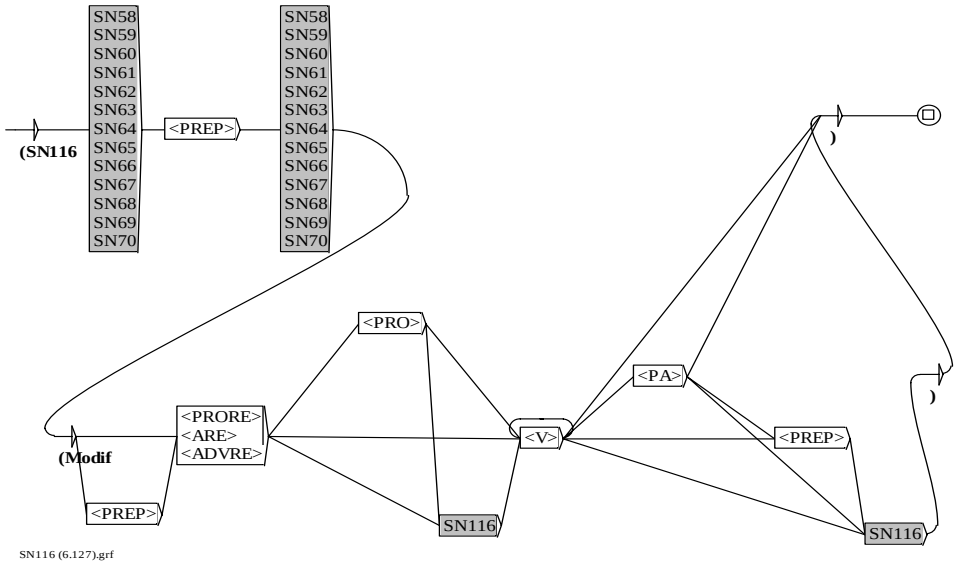


Fig. 6.127: FST gráfico que agrupa las variantes de la estructura recursiva SN_{116}

9. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{117} con iteración de constituyentes y recursividad de constituyentes oracionales (Fig. 6.128):

$$SN_{117} \rightarrow SN_{58}\ \text{PREP}\ SN_{57}\ \text{PRORE V}\ (SN_{117})$$
$$SN_{59}\ \text{PREP}\ SN_{57}\ \text{PRORE V}\ (SN_{117})$$
$$SN_{60}\ \text{PREP}\ SN_{57}\ \text{PRORE V}\ (SN_{117})$$
$$\ldots$$

$$ER_0 = \underline{\text{DET N}^+\ \text{A}}\ \text{PREP}\ \underline{\text{DET N}^+}\ \text{PRORE V}\ (SN_{117})$$
$$\underline{\text{DET N A}^+}\ \text{PREP}\ \underline{\text{DET N}^+}\ \text{PRORE V}\ (SN_{117})$$
$$\underline{\text{DET N}^+\ \text{A}^+}\ \text{PREP}\ \underline{\text{DET N}^+}\ \text{PRORE V}\ (SN_{117})$$
$$\ldots$$

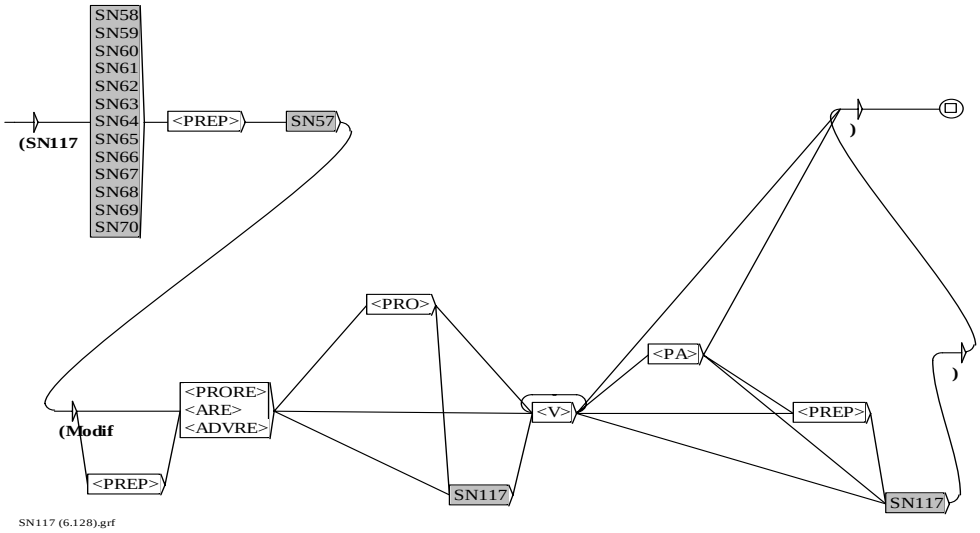


Fig. 6.128: FST gráfico que agrupa las variantes de la estructura recursiva SN_{117}

10. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{118} con recursividad de constituyentes preposicionales y oracionales (Fig. 6.129):

$$\mathbf{SN_{118}} \rightarrow \mathbf{SN_{93}\ PRORE\ V\ (SN_{118})}$$
$$\mathbf{SN_{93}\ PRORE\ (SN_{118})\ V}$$
$$\mathbf{SN_{93}\ PRORE\ PRO\ V\ PA\ (SN_{118})}$$
$$\ldots$$

$$ER_0 = \underline{N\ (PREP\ N)^*}\ PRORE\ V\ (SN_{118})$$
$$\underline{N\ N\ (PREP\ N\ N)^*}\ PRORE\ (SN_{118})\ V$$
$$\underline{DET\ N\ (PREP\ DET\ N)^*}\ PRORE\ PRO\ V\ PA\ (SN_{118})$$
$$\ldots$$

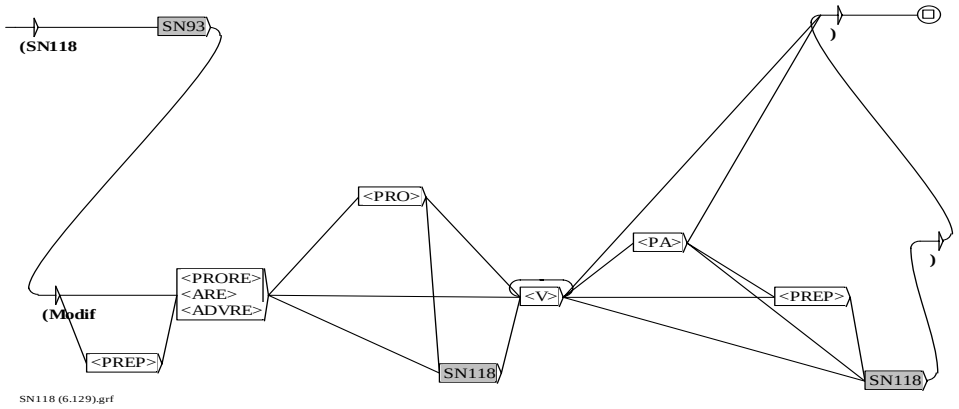


Fig. 6.129: FST gráfico que agrupa las variantes de la estructura recursiva SN_{118}

11. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{119} con recursividad de constituyentes preposicionales y oracionales (Fig. 6.130):

$$\mathbf{SN_{119}} \rightarrow \mathbf{SN_{94}\ PRORE\ V\ (SN_{119})}$$
$$\mathbf{SN_{94}\ PRORE\ (SN_{119})\ V}$$
$$\mathbf{SN_{94}\ PRORE\ PRO\ V\ PA\ (SN_{119})}$$
$$\ldots$$

$$ER_0 = \underline{N\ (PREP\ (N\ A)\ N)^*}\ PRORE\ V\ (SN_{119})$$
$$\underline{N\ N\ (PREP\ (N\ A)\ N\ N)^*}\ PRORE\ (SN_{119})\ V$$
$$\underline{DET\ N\ (PREP\ (N\ A)\ DET\ N)^*}\ PRORE\ V\ PA\ (SN_{119})$$
$$\ldots$$

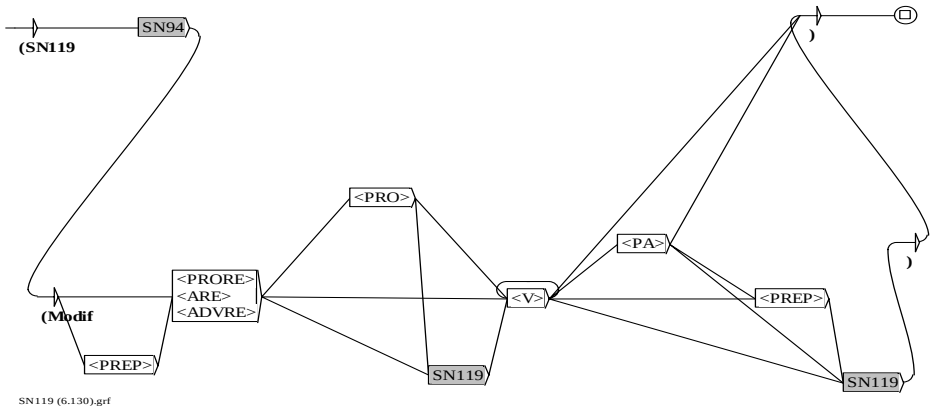


Fig. 6.130: FST gráfico que agrupa las variantes de la estructura recursiva SN_{119}

12. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{120} con recursividad de constituyentes preposicionales y oracionales (Fig. 6.131):

$$\mathbf{SN_{120}} \rightarrow \mathbf{SN_{95}\ PRORE\ V\ (SN_{120})}$$
$$\mathbf{SN_{95}\ PRORE\ (SN_{120})\ V}$$
$$\mathbf{SN_{95}\ PRORE\ PRO\ V\ PA\ (SN_{120})}$$
$$\ldots$$

$$ER_0 = \underline{N\ A\ (PREP\ (N\ A)\ N\ A)^*\ PRORE\ V\ (SN_{120})}$$
$$\underline{A\ N\ (PREP\ (A\ N)\ A\ N)^*\ PRORE\ (SN_{120})\ V}$$
$$\underline{N\ PA\ (PREP\ (N\ PA)\ N\ PA)^*\ PRORE\ PRO\ V\ PA\ (SN_{120})}$$
$$\ldots$$

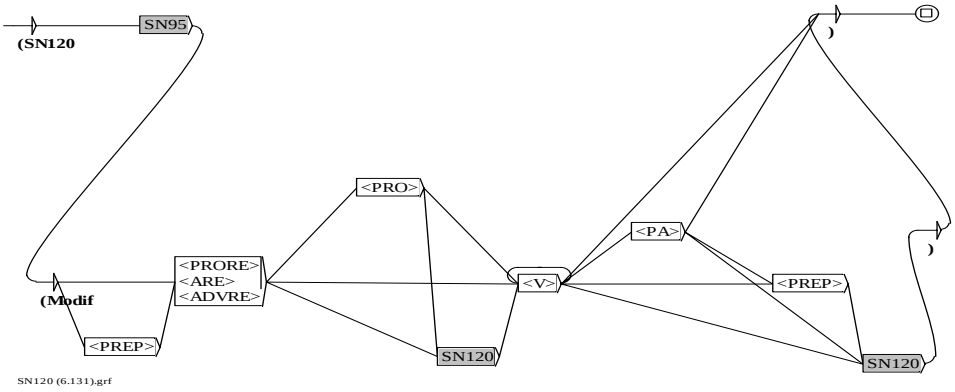


Fig. 6.131: FST gráfico que agrupa las variantes de la estructura recursiva SN_{120}

13. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{121} con recursividad de constituyentes preposicionales y oracionales (Fig. 6.132):

$$\mathbf{SN_{121} \rightarrow SN_{96}\ PRORE\ V\ (SN_{121})}$$
$$\mathbf{SN_{96}\ PRORE\ (SN_{121})\ V}$$
$$\mathbf{SN_{96}\ PRORE\ PRO\ V\ PA\ (SN_{121})}$$
$$\ldots$$

$$ER_0 = \underline{N\ A\ (PREP\ (N)\ N\ A)^*}\ PRORE\ V\ (SN_{121})$$
$$\underline{A\ N\ (PREP\ (N)\ A\ N)^*}\ PRORE\ (SN_{121})\ V$$
$$\underline{N\ PA\ (PREP\ (N)\ N\ PA)^*}\ PRORE\ PRO\ V\ PA\ (SN_{121})$$
$$\ldots$$

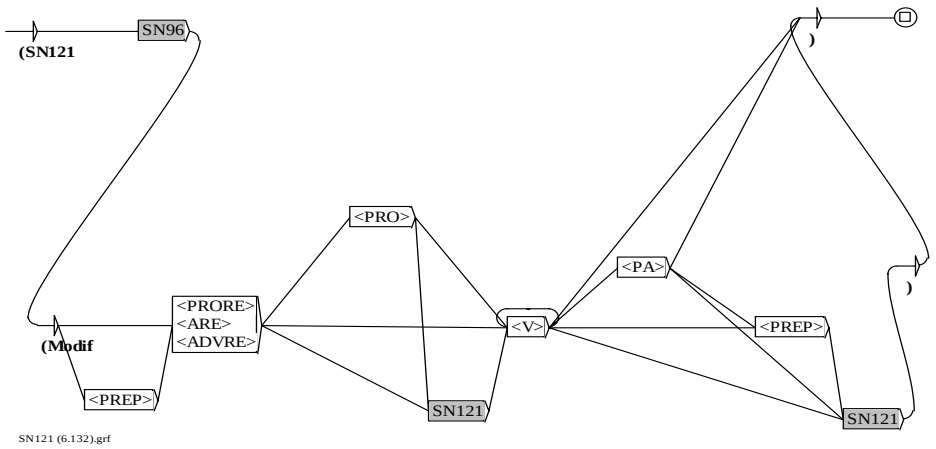


Fig. 6.132: FST gráfico que agrupa las variantes de la estructura recursiva $\mathtt{SN}_{121}$

14. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática $\mathtt{SN}_{122}$, con iteración de constituyentes, y con recursividad de constituyentes preposicionales y oracionales (Fig. 6.133):

$$\mathbf{SN_{122}} \rightarrow \mathbf{SN_{97}\ PRORE\ V\ (SN_{122})}$$
$$\mathbf{SN_{97}\ PRORE\ (SN_{122})\ V}$$
$$\mathbf{SN_{97}\ PRORE\ PRO\ V\ PA\ (SN_{122})}$$
$$\ldots$$

$$ER_0 = \underline{\mathtt{DET\ N^+\ \left(PREP\ \left(DET\ N^+\right)\ DET\ N^+\right)^*\ PRORE\ V\ (SN_{122})}}$$
$$\underline{\mathtt{N^+\ \left(PREP\ \left(N^+\right)\ N^+\right)^*\ PRORE\ (SN_{122})\ V}}$$
$$\underline{\mathtt{CUANT\ N^+\ \left(PREP\ \left(CUANT\ N^+\right)\ CUANT\ N^+\right)^*\ PRORE\ PRO\ V\ PA\ (SN_{122})}}$$
$$\ldots$$

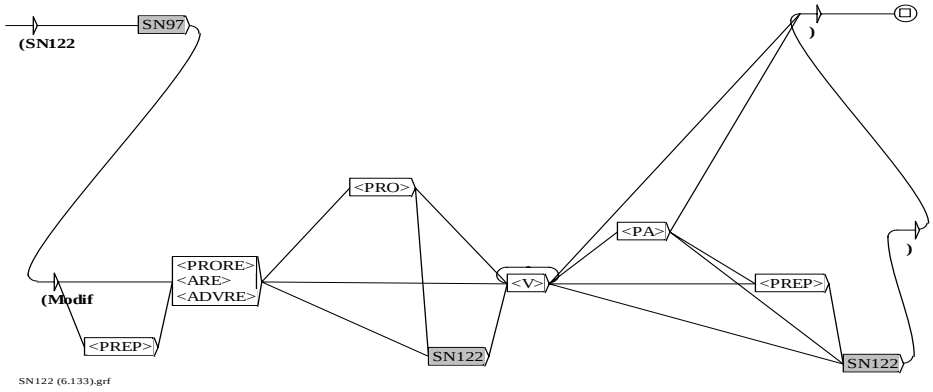


Fig. 6.133: FST gráfico que agrupa las variantes de la estructura recursiva SN_{122}

15. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{123}, con iteración de constituyentes, y con recursividad de constituyentes preposicionales y oracionales (Fig. 6.134):

$$\mathbf{SN_{123} \rightarrow SN_{98}\ PRORE\ V\ (SN_{123})}$$
$$\mathbf{SN_{98}\ PRORE\ (SN_{123})\ V}$$
$$\mathbf{SN_{98}\ PRORE\ PRO\ V\ PA\ (SN_{123})}$$
$$\mathbf{\ldots}$$

$$ER_0 = \underline{\text{DET N}^+ \left(\text{PREP}\left(\text{DET N}^+\ \text{A}\right)\text{DET N}^+\right)^*\ \text{PRORE V}\ (SN_{123})}$$
$$\underline{\text{N}^+ \left(\text{PREP}\left(\text{N}^+\ \text{A}\right)\text{N}^+\right)^*\ \text{PRORE}\ (SN_{123})\ \text{V}}$$
$$\underline{\text{CUANT N}^+ \left(\text{PREP}\left(\text{N}^+\ \text{A}\right)\text{CUANT N}^+\right)^*\ \text{PRORE PRO V PA}\ (SN_{123})}$$
$$\ldots$$

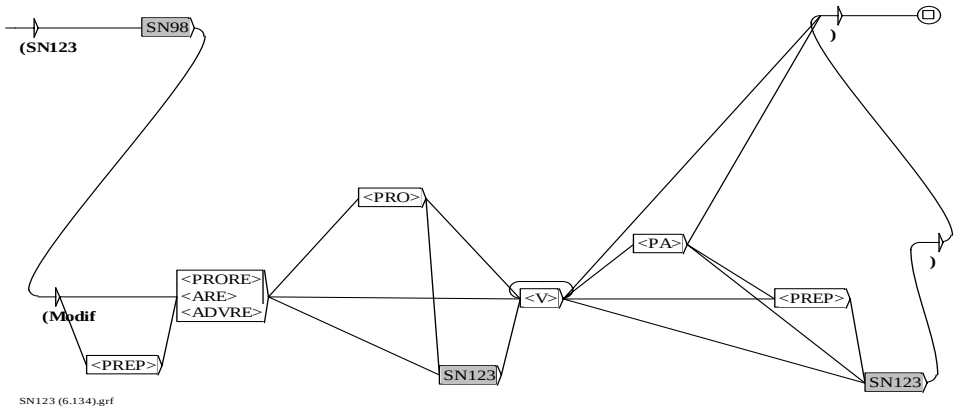


Fig. 6.134: FST gráfico que agrupa las variantes de la estructura recursiva SN_{123}

16. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{124}, con iteración de constituyentes, y con recursividad de constituyentes preposicionales y oracionales (Fig. 6.135):

$$\mathbf{SN_{124}} \rightarrow \mathbf{SN_{99}\ PRORE\ V\ (SN_{124})}$$
$$\mathbf{SN_{99}\ PRORE\ (SN_{124})\ V}$$
$$\mathbf{SN_{99}\ PRORE\ PRO\ V\ PA\ (SN_{124})}$$
$$\ldots$$

$$ER_0 = \underline{\text{DET N}^+ \text{ A } \left(\text{PREP } \left(\text{DET N}^+ \text{ A}\right) \text{ DET N}^+\right)^* \text{ PRORE V } (SN_{124})}$$
$$\underline{\text{N}^+ \text{ A } \left(\text{PREP } \left(\text{N}^+ \text{ A}\right) \text{ N}^+ \text{ A}\right)^* \text{ PRORE } (SN_{124}) \text{ V}}$$
$$\underline{\text{N}^+ \text{ A}^+ \left(\text{PREP } \left(\text{N}^+ \text{ A}^+\right) \text{ N}^+ \text{ A}^+\right)^* \text{ PRORE PRO V PA } (SN_{124})}$$
$$\ldots$$

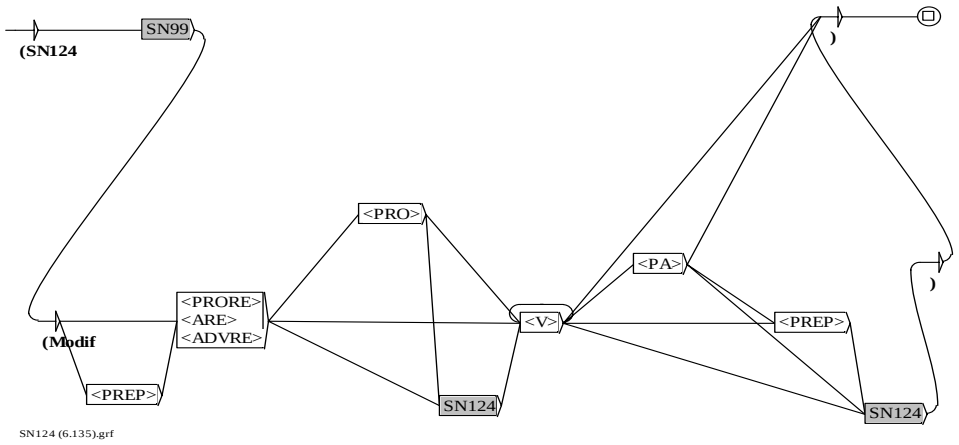


Fig. 6.135: FST gráfico que agrupa las variantes de la estructura recursiva SN_{124}

17. Transductor gráfico que representa, reconoce y agrupa las variantes de la construcción sintagmática SN_{125}, con iteración de constituyentes, y con recursividad de constituyentes preposicionales y oracionales (Fig. 6.136):

$$\begin{aligned} \mathbf{SN_{125}} \rightarrow\ & \mathbf{SN_{100}\ PRORE\ V\ (SN_{125})} \\ & \mathbf{SN_{100}\ PRORE\ (SN_{125})\ V} \\ & \mathbf{SN_{100}\ PRORE\ PRO\ V\ PA\ (SN_{125})} \\ & \ldots \end{aligned}$$

$$\begin{aligned} ER_0 = & \underline{\texttt{DET N}^{+}\ \texttt{A}\left(\texttt{PREP}\left(\texttt{DET N}^{+}\right)\texttt{DET N}^{+}\ \texttt{A}\right)^{*}\ \texttt{PRORE V}\ (SN_{125})} \\ & \underline{\texttt{DET N A}^{+}\left(\texttt{PREP}\left(\texttt{DET N}^{+}\right)\texttt{DET N A}^{+}\right)^{*}\ \texttt{PRORE}\ (SN_{125})\ \texttt{V}} \\ & \underline{\texttt{DET N}^{+}\ \texttt{A}^{+}\left(\texttt{PREP}\left(\texttt{DET N}^{+}\right)\texttt{DET N}^{+}\ \texttt{A}^{+}\right)^{*}\ \texttt{PRORE PRO V PA}\ (SN_{125})} \\ & \ldots \end{aligned}$$

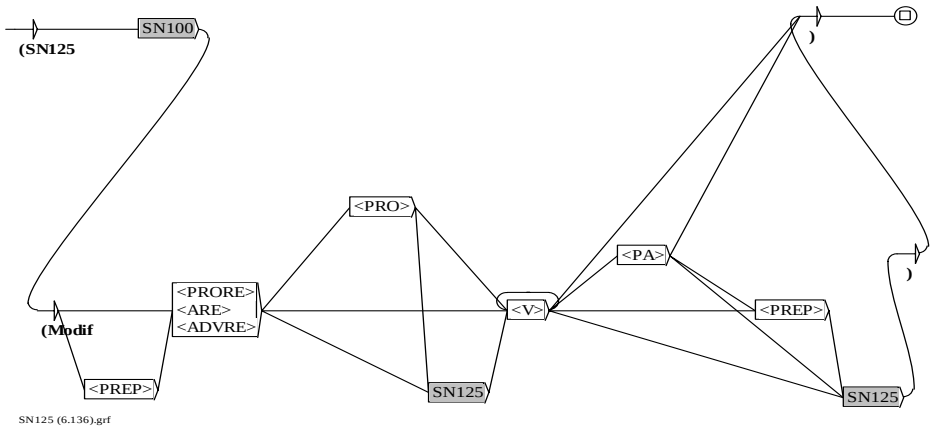


Fig. 6.136: FST gráfico que agrupa las variantes de la estructura recursiva SN_{125}

Capítulo 7

EVALUACIÓN DE LOS ANALIZADORES LÉXICOS Y SINTÁCTICOS

Una vez desarrolladas las herramientas de análisis, el paso siguiente es comprobar si las *hipótesis explicativas* con las que hemos desarrollado las bases de información permiten reconocer las variantes lingüísticas y, a continuación, evaluar si los índices generados adoptando métodos del PLN son eficaces para el reconocimiento y la agrupación de dichas variantes. Para ello vamos a aplicar los analizadores a un *corpus de verificación*. Como parámetro de evaluación vamos emplear una adaptación de las métricas clásicas de *precisión* y *exhaustividad* (Salton y McGill 1983). Este planteamiento no es nuevo, son muchos los trabajos que utilizan esta métrica para medir la adecuación de las gramáticas computacionales a la identificación de estructuras lingüísticas.

La evaluación de los analizadores se va a realizar con la aplicación informática diseñada por Silberztein (Silberztein 1999). El primer paso de este proceso será etiquetar el *corpus de*

verificación con la información aportada por los diccionarios electrónicos, los distintos modos con los que se pueden utilizar los diccionarios nos va a permitir identificar de forma automática todos los elementos léxicos necesarios para la evaluación. A continuación, seleccionaremos una muestra de los transductores sintácticos, que representen y agrupen distintos tipos de estructuras sintácticas canónicas de SSNN, y localizaremos en el *corpus* las distintas configuraciones sintácticas, resolviendo, en cada caso, los problemas de ambigüedad.

Con los datos obtenidos verificaremos la efectividad de los analizadores para generar índices con técnicas lingüísticas, mostrando los problemas y las limitaciones de estas herramientas en el tratamiento informático de los fenómenos lingüísticos, fundamentalmente el *infraanálisis* y el *sobreanálisis*. Por último, vamos a proponer distintas formas de mejorar los índices de *precisión* y *exhaustividad* obtenidos, con el objetivo de mejorar el rendimiento de los analizadores.

7.1. Composición del *corpus de verificación*

La construcción de los analizadores léxicos y sintácticos ha estado orientada por una colección de registros obtenidos de una base de datos, a su vez esa misma colección se ha utilizado como *corpus de verificación* para evaluar los resultados. La *orientación-a-los-datos* nos ha permitido tratar expresiones léxicas y sintácticas –como son ciertas expresiones de dominio– que no se hubieran podido tratar con otras herramientas de propósito general. Los registros se han obtenido de la *Base de Datos ISOC-Biblioteconomía y Documentación*, producida por el *Centro de Información y Documentación Científica* (CINDOC), en la que se recogen básicamente revistas y actas de congreso sobre literatura científica española en el área temática de Documentación Científica. En este contexto, la selección de la colección se ha dirigido según los siguientes criterios de representatividad:

- *Registros monolingües en idioma español.*
- *Registros con una cobertura amplia en el dominio de la documentación.*
- *Registros con un formato normalizado.*

Estos registros constituyen el *corpus de verificación* formado por un fichero de texto en formato *Windows ANSI*, en el que la corriente de caracteres que lo integran se divide en secuencias de unidades entre espacios en blanco, o *tokens*, cuya distribución estadísticas se muestra en la Tabla 7.1. Las formas léxicas agrupan las unidades léxicas, las formas etiquetadas agrupan las formas léxicas vinculadas a información lingüística, los dígitos integran las cifras y los delimitadores integran los distintos separadores –como puntos, comas o guiones–.

TABLA 7.1: Composición del *corpus de verificación*

	tokens	*tokens differentes*
Formas léxicas	*18215*	*3296*
Formas etiquetadas	*0*	*0*
Dígitos	*4082*	*10*
Delimitadores	*5508*	*16*
Líneas	*1676*	
Número total de tokens	***27805***	***3322***

Antes de aplicar los analizadores léxicos y sintácticos es necesario efectuar un pre-procesamiento del *corpus* consistente en someter el texto de entrada a una serie de transformaciones con las funciones de: *a)* identificar los elementos que reflejen la *estructura lógica del texto* –párrafos, oraciones, signos de puntuación, o delimitadores–, *b)* reconocer las *formas compuestas no-autónomas* y *c) controlar ciertas formas* que no se pueden identificar con los analizadores léxicos. La etapa de pre-análisis lingüístico se realiza por medio de las siguientes fases:

1. La primera fase consiste en *insertar marcas de delimitación*, {**S**}, al final de cada elemento lógico, o sentencia reconocida. Delimitar las sentencias no es tan fácil como pudiera parecer en un principio porque las abreviaturas (tales como «*Univ.*») o las siglas (tales como «*I.S.B.N.*») pueden conducir a una segmentación errónea de sentencias, mientras que los títulos que no son el final de un período pueden provocar la concatenación de varias sentencias, cuando en realidad se trataría de sentencias distintas. Para la adecuada fragmentación de estas unidades lógicas de análisis hemos diseñado un FST gráfico (Fig.7.1) a partir de una adaptación previa de la aplicación desarrollada por Silberztein, que se encarga de introducir marcas superficiales al comienzo de cada párrafo, después de *un punto*, después de *un punto y coma*, o después de cualquier *delimitador*. Además, incluye otro transductor imbricado LetraMayus (Fig. 7.2), cuya función es reconocer cualquier letra mayúscula seguida de un *punto*. Una vez aplicado el transductor anterior, a cada sentencia se le asigna un número, lo cual facilita que podamos hacer referencia a una sentencia en particular, además de darnos la posibilidad de conocer el número de sentencias en las que se divide el *corpus*.

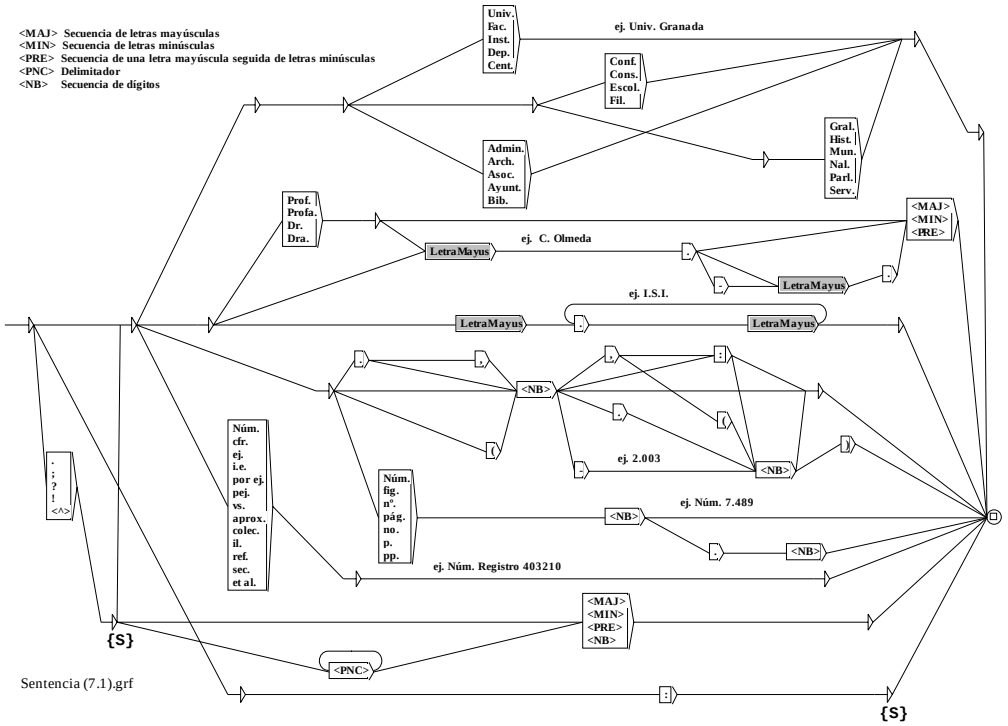


Fig. 7.1: FST gráfico que realiza el etiquetado estructural

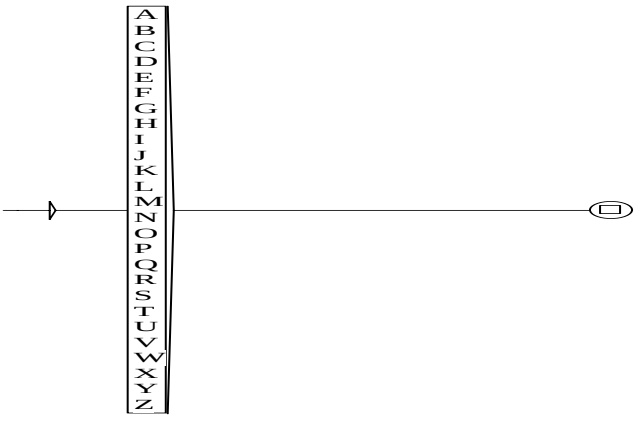
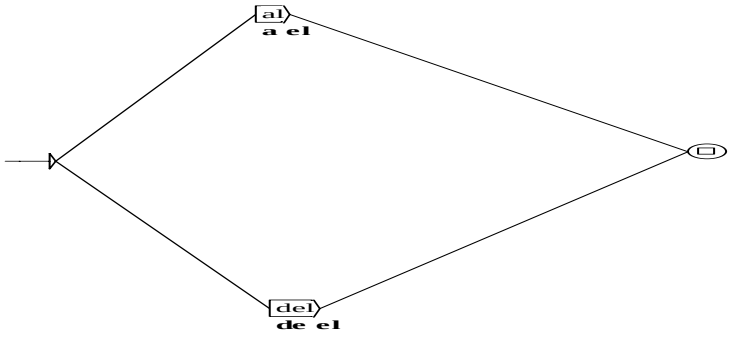


Fig. 7.2: Representación de las letras mayúsculas en un FST

2. La segunda fase consiste en aplicar el *diccionario de palabras compuestas no-ambiguas* que se utiliza para analizar los términos compuestos de forma integrada, y no a partir de sus elementos constituyentes. Se trataría de identificar y etiquetar *términos compuestos no-ambiguos*, así como locuciones o expresiones constituidas por varias palabras, con una forma fija, pero que se utilizan como un solo términos. Los términos compuestos se pueden representar en forma de un diccionario, o en forma de transductores. Las entradas del diccionario serían del tipo:

Consejo Superior de Investigaciones Científicas,*Consejo Superior de Investigaciones Científicas*.**N:ms**
Centro de Documentación Europea,*Centro de Documentación Europea*.**N:ms**
Univ. Politécnica de Valencia,*Universidad Politécnica de Valencia*.**N:fs**
Univ. Internacional de Andalucía,*Universidad Internacional de Andalucía*.**N:fs**
Univ. Nacional de Educación a Distancia,*Universidad Nacional de Educación a Distancia*.**N:fs**
/.../

3. La tercera fase del pre-procesamiento consiste en *eliminar la ambigüedad de las formas contractas* por medio de FST gráficos, dichas formas se definen como aquellos términos que no se pueden adscribir a una sola categoría porque formalmente equivalen a dos categorías sucesivas. Esto es lo que ocurre con las formas «*al*» y «*del*» que equivalen a la combinación del artículo «*el*» con las preposiciones «*a*» y «*de*». El FST (Fig. 7.3) que se encarga de transformar las formas contractas en unidades no ambiguas es el siguiente:



Replace (7.3).grf

Fig. 7.3: FST que separa formas contractas

La ejecución de las tres etapas de pre-procesamiento a la colección de verificación da como resultado la segmentación y normalización del *corpus*, en la que los marcadores superficiales se encargan de reflejar la estructura lógica del texto, en la que se identifican los términos compuestos y en la que se transforman las formas contractas, tal y como se muestra en la Tabla 7.2.

TABLA 7. 2: Resultado de la etapa de *pre-procesamiento* en un registro de la colección

{S}Núm. Registro:{S} 408834

{S}Autores:{S} Moya Anegón, Félix de;{S}Moscoso, Purificación;{S}Olmeda, Carlos;{S}Ortiz Repiso, Virginia;{S}Herrero, Víctor;{S}Guerrero, Vicente

{S}Título:{S} Neurolsoc:{S} Un modelo de red neuronal para la representación de el conocimiento

{S}Lugar de trabajo:{S} Univ. Granada, España;{S}Univ. Alcalá de Henares, Madrid, España;{S}Univ. Carlos III, Fac. Humanidades, Comunicación y Documentación, Dep. Biblioteconomía y Documentación, Getafe, Madrid, España;{S}Univ. Extremadura, España

{S}ISBN:{S} 84-699-0289-X

{S}Congreso:{S} Congreso ISKO-ESPAÑA:{S} La representación y la organización de el conocimiento en sus distintas perspectivas.{S} IV. 1999.{S} Granada

{S}Datos fuente:{S} 1999, : 151-156, 8 ref

{S}Tipo documento:{S} Actas de congresos

{S}Modo documento:{S} Ponencia.{S} Comunicación

{S}Lengua:{S} Español

{S}Localización:{S} ISOC

{S}Editor:{S} Granada:{S} S.N., 1999

{S}Descriptores:{S} Bases de datos;{S}Producción científica;{S}Representación de el conocimiento;{S}Recuperación de la información

{S}Identificadores:{S} ISOC (Base de datos)

{S}Clasificación:{S} 200304 Bases de datos.

{S}Resumen:{S} El propósito de esta ponencia es presentar un modelo de red neuronal que se ha desarrollado con el fin de representar el conocimiento expresado a través de la producción científica en el campo de las Ciencias Sociales y las Humanidades.{S} Dicho modelo se ha aplicado a el caso concreto de la base de datos ISOC, producida y distribuida por el Consejo Superior de Investigaciones Científicas.{S} Esta aplicación forma parte de un proyecto de investigación cuyo objetivo principal es el desarrollo de una interfaz de realidad virtual basada en motores de clasificación que utilizan técnicas multivariantes o redes neuronales para posibilitar el acceso mediante browsing a los registros contenidos en una base de datos.{S} Con el fin de representar las relaciones existentes entre las distintas materias que conforman el área de

las Ciencias Sociales y las Humanidades, se han formado conjuntos de documentos a partir de los códigos de clasificación utilizados en la base de datos ISOC.{S} Dichas relaciones se representan mediante matrices de coocurrencia de números de clasificación.{S} Las matrices se forman siguiendo la estructura jerárquica de la propia clasificación.{S} Estas matrices, una vez normalizadas, constituyen la entrada de un proceso de red neuronal que se basa en los mapas afltoorganizativos de Kohonen (SOM).{S} De las distintas salidas que produce el simulador de la red neuronal se utiliza la matriz de tasas de activación como entrada de una aplicación ad hoc que genera los mapas cuyas topologías representan el conocimiento extraído de la base de datos.{S} El resultado de la aplicación de la metodología descrita es un árbol de mapas que permite a el usuario navegar a través de el conocimiento extraído de la base de datos.{S} De esta forma, se genera una interfaz que expresa la topología, entendida como conjunto de vecindades, de las distintas categorías temáticas codificadas en esta base de datos.

7.2. Aplicación de las herramientas de análisis léxico

Una vez pre-procesadas las sentencias de la colección de verificación, el paso siguiente es identificar las unidades léxicas, para ello se utilizan los analizadores léxicos que se han construido previamente y que están integradas por: *1) Diccionarios electrónicos* de palabras simples y compuestas representados en FST; y *2) Transductores léxicos.* Las bases de información léxicas desarrolladas con estas herramientas constituyen un *Diccionario o Lexicón Electrónico*, cuyas entradas se distribuyen según las especificaciones de la Tabla 7.3.

TABLA 7.3: Composición del diccionario electrónico

Formas flexionadas simples		*60511*
Formas canónicas simples	*4772*	
Formas flexionadas compuestas		*1148*
Formas compuestas	*1148*	
Número total de entradas		***61659***

Con la aplicación de los analizadores léxicos se obtienen dos transformaciones sobre las unidades léxicas del *corpus* que aportan información lingüística por medio de dos procesos: *1) Lematización*, o reducción de formas flexionadas a formas canónicas; y *2) Asignación de etiquetas morfosintácticas*, categorías POS, a todas las unidades léxicas. A su vez, el resultado de los dos procesos anteriores se puede presentar de cuatro modos distintos en el etiquetado lineal:

1. Análisis de las unidades léxicas del *corpus de verificación* en **`Lemas`**, como se muestra en la Tabla 7.4.

TABLA 7.4: Resultado de la transformación de las unidades léxicas de un registro en *lemas*

número. Registro: 408834
autor: Moya Anegón, Félix de;Moscoso, Purificación;Olmeda, Carlos;Ortiz Repiso, Virginia;Herrero, Víctor;Guerrero, Vicente
título: Neurolsoc: un modelo de red neuronal para la representación de el conocimiento
lugar de trabajo: universidad. Granada, España;universidad. Alcalá de Henares, Madrid, España;universidad. Carlos III, facultad. humanidad, comunicación y Documentación, departamento. Biblioteconomía y Documentación, Getafe, Madrid, España;universidad. Extremadura, España
ISBN: 84-699-0289-X
congreso: congreso ISKO-España: La representación y la organización de el conocimiento en suyo distinto perspectiva. IV. 1999. Granada
dato fuente: 1999, : 151-156, 8 referencia
tipo documento: acta de congreso
modo documento: ponencia. comunicación
lengua: español
localización: ISOC
editor: Granada: S.nombre., 1999
descriptor: Bases de dato;producción científico;representación de el conocimiento;recuperación de la información
identificador: ISOC (Base de dato)
clasificación: 200304 Bases de dato.
Resumen: El propósito de esta ponencia ser presentar un modelo de red neuronal que se haber desarrollado con el fin de representar el conocimiento expresado a través de la producción científico en el campo de las ciencia social y las humanidad. Dicho modelo se haber aplicado a el caso concreto de la base de dato ISOC, producido y distribuido por el Consejo Superior de Investigaciones Científicas. Esta aplicación forma parte de un proyecto de investigación

cuyo objetivo principal ser el desarrollar de una interfaz de realidad virtual basado en motor de clasificación que utilizar técnicas multivariante o red neuronal para posibilitar el acceso mediante browsing a los registro contenido en una base de dato. con el fin de representar las relaciones existente entre las distinto materia que conformar el área de las ciencia social y las humanidad, se haber formado conjunto de documento a partir de los código de clasificación utilizado en la base de dato ISOC. dicha relaciones se representar mediante matriz de coocurrencia de número de clasificación. Las matriz se formar seguir la estructura jerárquico de la propio clasificación. Estas matriz, una vez normalizado, constituir la entrada de un proceso de red neuronal que se basar en los mapa afltoorganizativos de Kohonen (SOM). de las distinto salida que producir el simulador de la red neuronal se utilizar la matriz de tasa de activación como entrada de una aplicación ad hoc que generar los mapa cuyo topología representar el conocimiento extraído de la base de dato. El resultado de la aplicación de la metodología descrito ser un árbol de mapa que permitir a el usuario navegar a través de el conocimiento extraído de la base de dato. de esta forma, se generar una interfaz que expresar la topología, entendido como conjunto de vecindad, de las distinto categoría temático codificado en esta base de dato.

2. Etiquetado de las unidades léxicas en **`Lemas`** + **`categorías POS`**, como se muestra en la Tabla 7.5.

TABLA 7.5: Resultado de la *etiquetación* de las unidades léxicas de un registro en *lemas* y *categorías*

{S}{número,.N}. Registro:{S} 408834

{S}{autor,.N}:{S} Moya Anegón, Félix {de,.PREP};{S}Moscoso, Purificación;{S}Olmeda, Carlos;{S}Ortiz Repiso, Virginia;{S}Herrero, Víctor;{S}Guerrero, Vicente

{S}{título,.N}:{S} {Neurolsoc,.N}:{S} {un,.DET} modelo {de,.PREP} {red,.N} {neuronal,.A} para la {representación,.N} {de,.PREP} {el,.DET} {conocimiento,.N}

{S}{lugar,.N} {de,.PREP} trabajo:{S} {Universidad de Granada,.N}, {España,.N};{S}{Universidad de Alcalá de Henares,.N}, {Madrid,.N}, {España,.N};{S}{Universidad Carlos III,.N}, {facultad,.N}. {humanidad,.N}, {comunicación,.N} {y,.CONJC} Documentación, {departamento,.N}. Biblioteconomía {y,.CONJC} Documentación, {Getafe,.N}, {Madrid,.N}, {España,.N};{S}{Universidad de Extremadura,.N}, {España,.N}

{S}{ISBN,.N}:{S} 84-699-0289-X

{S}{congreso,.N}:{S} {congreso,.N} {ISKO,.N}-{España,.N}:{S} La {representación,.N} {y,.CONJC} la {organización,.N} {de,.PREP} {el,.DET} {conocimiento,.N} {en,.PREP} {suyo,.POS} {distinto,.A} {perspectiva,.N}.{S} IV. 1999.{S} {Granada,.N}

{S}{dato,.N} {fuente,.N}:{S} 1999, : 151-156, 8 {referencia,.N}

{S}{tipo,.N} documento:{S} {acta,.N} {de,.PREP} {congreso,.N}

{S}{modo,.N} documento:{S} {ponencia,.N}.{S} {comunicación,.N}

{S}{lengua,.N}:{S} {español,.A}

{S}{localización,.N}:{S} {ISOC,.N}

{S}{editor,.N}:{S} {Granada,.N}:{S} S.N., 1999

{S}{descriptor,.N}:{S} Bases {de,.PREP} {dato,.N};{S}{producción,.N} científica;{S}{representación,.N} {de,.PREP} {el,.DET} {conocimiento,.N};{S}{recuperación,.N} {de,.PREP} la {información,.N}

{S}{identificador,.N}:{S} {ISOC,.N} (Base {de,.PREP} {dato,.N})

{S}{clasificación,.N}:{S} 200304 Bases {de,.PREP} {dato,.N}.

{S}Resumen:{S} El {propósito,.N} {de,.PREP} esta {ponencia,.N} {ser,.V} {presentar,.V} {un,.DET} modelo {de,.PREP} {red,.N} {neuronal,.A} que {se,.PRO} {haber,.V} desarrollado {con el fin de,.PREP} {representar,.V} {el,.DET} {conocimiento,.N} expresado {a través de,.PREP} la {producción,.N} científica {en,.PREP} {el,.DET} {campo,.N} {de,.PREP} las {ciencia,.N} {social,.A} {y,.CONJC} las {humanidad,.N}.{S} Dicho modelo {se,.PRO} {haber,.V} aplicado {a,.PREP} {el,.DET} {caso,.N} concreto {de,.PREP} la base {de,.PREP} {dato,.N} {ISOC,.N}, {producido,.PA} {y,.CONJC} {distribuido,.PA} {por,.PREP} {el,.DET} {Consejo Superior de Investigaciones Científicas,.N}.{S} Esta {aplicación,.N} forma parte {de,.PREP} {un,.DET} proyecto {de,.PREP} {investigación,.N} {cuyo,.ARE} objetivo {principal,.A} {ser,.V} {el,.DET} {desarrollar,.V} {de,.PREP} una {interfaz,.N} {de,.PREP} {realidad,.N} {virtual,.A} {basado,.PA} {en,.PREP} {motor,.N} {de,.PREP} {clasificación,.N} que {utilizar,.V} técnicas {multivariante,.A} {o,.CONJC} {red,.N} {neuronal,.A} para {posibilitar,.V} {el,.DET} {acceso,.N} {mediante,.A} {browsing,.N} {a,.PREP} los {registro,.N} contenidos {en,.PREP} una base {de,.PREP} {dato,.N}.{S} {con el fin de,.PREP} {representar,.V} las relaciones {existente,.A} entre las {distinto,.A} {materia,.N} que {conformar,.V} {el,.DET} {área,.N} {de,.PREP} las {ciencia,.N} {social,.A} {y,.CONJC} las {humanidad,.N}, {se,.PRO} {haber,.V} formado {conjunto,.N} {de,.PREP} {documento,.N} {a partir de,.PREP} los {código,.N} {de,.PREP} {clasificación,.N} {utilizado,.PA} {en,.PREP} la base {de,.PREP} {dato,.N} {ISOC,.N}.{S} {dicha,.N} relaciones {se,.PRO} {representar,.V} {mediante,.A} {matriz,.N} {de,.PREP} {coocurrencia,.N} {de,.PREP} {número,.N} {de,.PREP} {clasificación,.N}.{S} Las {matriz,.N} {se,.PRO} {formar,.V} {seguir,.V} la estructura {jerárquico,.A} {de,.PREP} la {propio,.A} {clasificación,.N}.{S} Estas {matriz,.N}, una {vez,.N} {normalizado,.PA}, {constituir,.V} la entrada {de,.PREP} {un,.DET} proceso {de,.PREP} {red,.N} {neuronal,.A} que {se,.PRO} {basar,.V} {en,.PREP} los {mapa,.N} afltoorganizativos {de,.PREP} Kohonen ({SOM,.N}).{S} {de,.PREP} las {distinto,.A} {salida,.N} que {producir,.V} {el,.DET} {simulador,.N} {de,.PREP} la {red,.N} {neuronal,.A} {se,.PRO} {utilizar,.V} la {matriz,.N} {de,.PREP} {tasa,.N} {de,.PREP} {activación,.N} como entrada {de,.PREP} una {aplicación,.N} {ad hoc,.NL} que {generar,.V} los {mapa,.N} {cuyo,.ARE} {topología,.N} {representar,.V} {el,.DET} {conocimiento,.N} extraído {de,.PREP} la base {de,.PREP} {dato,.N}.{S} El resultado {de,.PREP} la {aplicación,.N} {de,.PREP} la {metodología,.N} {descrito,.PA} {ser,.V} {un,.DET} {árbol,.N} {de,.PREP} {mapa,.N} que {permitir,.V} {a,.PREP} {el,.DET} {usuario,.N} {navegar,.V} {a través de,.PREP} {el,.DET} {conocimiento,.N} extraído {de,.PREP} la base {de,.PREP} {dato,.N}.{S} {de,.PREP} esta forma, {se,.PRO} {generar,.V} una {interfaz,.N} que {expresar,.V} la {topología,.N}, {entendido,.PA} como conjunto {de,.PREP} {vecindad,.N}, {de,.PREP} las {distinto,.A} {categoría,.N} {temático,.A} {codificado,.PA} {en,.PREP} esta base {de,.PREP} {dato,.N}.

3. Etiquetado de las unidades léxicas en **`Formas flexionadas`** + **`categorías POS`**, como se muestra en la Tabla 7.6.

TABLA 7.6: Resultado de la etiquetación de las unidades léxicas de un registro en *formas flexionadas* y *categorías*

{S}{Núm,.N}. Registro:{S} 408834
{S}{autores,.N}:{S} Moya Anegón, Félix {de,.PREP};{S}Moscoso, Purificación;{S}Olmeda, Carlos;{S}Ortiz Repiso, Virginia;{S}Herrero, Víctor;{S}Guerrero, Vicente
{S}{título,.N}:{S} {Neurolsoc,.N}:{S} {un,.DET} modelo {de,.PREP} {red,.N} {neuronal,.A} para la {representación,.N} {de,.PREP} {el,.DET} {conocimiento,.N}
{S}{lugar,.N} {de,.PREP} trabajo:{S} {Univ. Granada,.N}, {España,.N};{S}{Univ. Alcalá de Henares,.N}, {Madrid,.N}, {España,.N};{S}{Univ. Carlos III,.N}, {Fac,.N}. {humanidades,.N}, {comunicación,.N} {y,.CONJC} {Documentación,.N}, {Dep,.N}. {Biblioteconomía,.N} {y,.CONJC} {Documentación,.N}, {Getafe,.N}, {Madrid,.N}, {España,.N};{S}{Univ. Extremadura,.N}, {España,.N}
{S}{ISBN,.N}:{S} 84-699-0289-X
{S}{congreso,.N}:{S} {congreso,.N} {ISKO,.N}-{España,.N}:{S} La {representación,.N} {y,.CONJC} la {organización,.N} {de,.PREP} {el,.DET} {conocimiento,.N} {en,.PREP} {sus,.POS} {distintas,.A} {perspectivas,.N}.{S} IV. 1999.{S} {Granada,.N}
{S}{datos,.N} {fuente,.N}:{S} 1999, : 151-156, 8 {ref,.N}
{S}{tipo,.N} documento:{S} {actas,.N} {de,.PREP} {congresos,.N}
{S}{modo,.N} documento:{S} {ponencia,.N}.{S} {comunicación,.N}
{S}{lengua,.N}:{S} {español,.A}
{S}{localización,.N}:{S} {ISOC,.N}
{S}{editor,.N}:{S} {Granada,.N}:{S} S.N., 1999
{S}{descriptores,.N}:{S} Bases {de,.PREP} {datos,.N};{S}{producción,.N} científica;{S}{representación,.N} {de,.PREP} {el,.DET} {conocimiento,.N};{S}{recuperación,.N} {de,.PREP} la {información,.N}
{S}{identificadores,.N}:{S} {ISOC,.N} (Base {de,.PREP} {datos,.N})
{S}{clasificación,.N}:{S} 200304 Bases {de,.PREP} {datos,.N}.
{S}Resumen:{S} El {propósito,.N} {de,.PREP} esta {ponencia,.N} {es,.V} {presentar,.V} {un,.DET} modelo {de,.PREP} {red,.N} {neuronal,.A} que {se,.PRO} {ha,.V} desarrollado {con el fin de,.PREP} {representar,.V} {el,.DET} {conocimiento,.N} expresado {a través de,.PREP} la {producción,.N} científica {en,.PREP} {el,.DET} {campo,.N} {de,.PREP} las {ciencias,.N} {sociales,.A} {y,.CONJC} las {humanidades,.N}.{S} Dicho modelo {se,.PRO} {ha,.V} aplicado {a,.PREP} {el,.DET} {caso,.N} concreto {de,.PREP} la base {de,.PREP} {datos,.N} {ISOC,.N}, {producida,.PA} {y,.CONJC} {distribuida,.PA}

{por,.PREP} {el,.DET} {Consejo Superior de Investigaciones Científicas,.N}.{S} Esta {aplicación,.N} forma parte {de,.PREP} {un,.DET} proyecto {de,.PREP} {investigación,.N} {cuyo,.ARE} objetivo {principal,.A} {es,.V} {el,.DET} {desarrollo,.V} {de,.PREP} una {interfaz,.N} {de,.PREP} {realidad,.N} {virtual,.A} {basada,.PA} {en,.PREP} {motores,.N} {de,.PREP} {clasificación,.N} que {utilizan,.V} técnicas {multivariantes,.A} {o,.CONJC} {redes,.N} {neuronales,.A} para {posibilitar,.V} {el,.DET} {acceso,.N} {mediante,.A} {browsing,.N} {a,.PREP} los {registros,.N} contenidos {en,.PREP} una base {de,.PREP} {datos,.N}.{S} {con el fin de,.PREP} {representar,.V} las relaciones {existentes,.A} entre las {distintas,.A} {materias,.N} que {conforman,.V} {el,.DET} {área,.N} {de,.PREP} las {ciencias,.N} {sociales,.A} {y,.CONJC} las {humanidades,.N}, {se,.PRO} {han,.V} formado {conjuntos,.N} {de,.PREP} {documentos,.N} {a partir de,.PREP} los {códigos,.N} {de,.PREP} {clasificación,.N} {utilizados,.PA} {en,.PREP} la base {de,.PREP} {datos,.N} {ISOC,.N}.{S} {dichas,.N} relaciones {se,.PRO} {representan,.V} {mediante,.A} {matrices,.N} {de,.PREP} {coocurrencia,.N} {de,.PREP} {números,.N} {de,.PREP} {clasificación,.N}.{S} Las {matrices,.N} {se,.PRO} {forman,.V} {siguiendo,.V} la estructura {jerárquica,.A} {de,.PREP} la {propia,.A} {clasificación,.N}.{S} Estas {matrices,.N}, una {vez,.N} {normalizadas,.PA}, {constituyen,.V} la entrada {de,.PREP} {un,.DET} proceso {de,.PREP} {red,.N} {neuronal,.A} que {se,.PRO} {basa,.V} {en,.PREP} los {mapas,.N} afltoorganizativos {de,.PREP} Kohonen ({SOM,.N}).{S} {de,.PREP} las {distintas,.A} {salidas,.N} que {produce,.V} {el,.DET} {simulador,.N} {de,.PREP} la {red,.N} {neuronal,.A} {se,.PRO} {utiliza,.V} la {matriz,.N} {de,.PREP} {tasas,.N} {de,.PREP} {activación,.N} como entrada {de,.PREP} una {aplicación,.N} {ad hoc,.NL} que {genera,.V} los {mapas,.N} {cuyas,.ARE} {topologías,.N} {representan,.V} {el,.DET} {conocimiento,.N} extraído {de,.PREP} la base {de,.PREP} {datos,.N}.{S} El resultado {de,.PREP} la {aplicación,.N} {de,.PREP} la {metodología,.N} {descrita,.PA} {es,.V} {un,.DET} {árbol,.N} {de,.PREP} {mapas,.N} que {permite,.V} {a,.PREP} {el,.DET} {usuario,.N} {navegar,.V} {a través de,.PREP} {el,.DET} {conocimiento,.N} extraído {de,.PREP} la base {de,.PREP} {datos,.N}.{S} {de,.PREP} esta forma, {se,.PRO} {genera,.V} una {interfaz,.N} que {expresa,.V} la {topología,.N}, {entendida,.PA} como conjunto {de,.PREP} {vecindades,.N}, {de,.PREP} las {distintas,.A} {categorías,.N} {temáticas,.A} {codificadas,.PA} {en,.PREP} esta base {de,.PREP} {datos,.N}.

4. Etiquetado de las unidades léxicas vinculando las formas flexionadas a las **`Formas Léxicas`** almacenadas en el diccionario, como se muestra en la Tabla 7.7.

TABLA 7.7: Resultado de la *etiquetación* de las unidades léxicas de un registro según las *formas léxicas* del diccionario

{S}{Núm,número.N:ms}. Registro:{S} 408834

{S}{autores,autor.N13:mp}:{S} Moya Anegón, Félix {de,.PREP};{S}Moscoso, Purificación;{S}Olmeda, Carlos;{S}Ortiz Repiso, Virginia;{S}Herrero, Víctor;{S}Guerrero, Vicente
{S}{título,.N4:ms}:{S} {Neurolsoc,.N+PR:ms}:{S} {un,.DET1+Dind:ms} modelo {de,.PREP} {red,.N21:fs} {neuronal,.A2:ms:fs} para la {representación,.N3:fs} {de,.PREP} {el,.DET2+Ddef:ms} {conocimiento,.N4:ms}
{S}{lugar,.N2:ms} {de,.PREP} trabajo:{S} {Univ. Granada,Universidad de Granada.N:fs}, {España,.N+Top:ms};{S}{Univ. Alcalá de Henares,Universidad de Alcalá de Henares.N:fs}, {Madrid,.N+Top:ms}, {España,.N+Top:ms};{S}{Univ. Carlos III,Universidad Carlos III.N:fs}, {Fac,facultad.N:fs}. {humanidades,humanidad.N21:fp}, {comunicación,.N3:fs} {y,.CONJC} Documentación, {Dep,departamento.N:mp}. Biblioteconomía {y,.CONJC} Documentación, {Getafe,.N+Top:ms}, {Madrid,.N+Top:ms}, {España,.N+Top:ms};{S}{Univ. Extremadura,Universidad de Extremadura.N:fs}, {España,.N+Top:ms}
{S}{ISBN,.N10:fs}:{S} 84-699-0289-X
{S}{congreso,.N4:ms}:{S} {congreso,.N4:ms} {ISKO,.N102:ms}-{España,.N+Top:ms}:{S} La {representación,.N3:fs} {y,.CONJC} la {organización,.N3:fs} {de,.PREP} {el,.DET2+Ddef:ms} {conocimiento,.N4:ms} {en,.PREP} {sus,suyo.POS5:3mp:fp} {distintas,distinto.A1:fp} {perspectivas,perspectiva.N5:fp}.{S} IV. 1999.{S} {Granada,.N+Top:ms}
{S}{datos,dato.N4:mp} {fuente,.N5:fs}:{S} 1999, : 151-156, 8 {ref,referencia.N:fs}
{S}{tipo,.N4:ms} documento:{S} {actas,acta.N5:fp} {de,.PREP} {congresos,congreso.N4:mp}
{S}{modo,.N4:ms} documento:{S} {ponencia,.N5:fs}.{S} {comunicación,.N3:fs}
{S}{lengua,.N5:fs}:{S} {español,.A3:ms}
{S}{localización,.N3:fs}:{S} {ISOC,.N102:ms}
{S}{editor,.N13:ms}:{S} {Granada,.N+Top:ms}:{S} S.{n,nombre.N:ms}., 1999
{S}{descriptores,descriptor.N2:mp}:{S} Bases {de,.PREP} {datos,dato.N4:mp};{S}{producción,.N3:fs} científica;{S}{representación,.N3:fs} {de,.PREP} {el,.DET2+Ddef:ms} {conocimiento,.N4:ms};{S}{recuperación,.N3:fs} {de,.PREP} la {información,.N3:fs}
{S}{identificadores,identificador.N13:mp}:{S} {ISOC,.N102:ms} (Base {de,.PREP} {datos,dato.N4:mp})
{S}{clasificación,.N3:fs}:{S} 200304 Bases {de,.PREP} {datos,dato.N4:mp}.
{S}Resumen:{S} El {propósito,.N4:ms} {de,.PREP} esta {ponencia,.N5:fs} {es,ser.V211:P3s} {presentar,.V1:W} {un,.DET1+Dind:ms} modelo {de,.PREP} {red,.N21:fs} {neuronal,.A2:ms:fs} que se {ha,haber.V212:P3s} desarrollado {con el fin de,.PREP} {representar,.V1:W} {el,.DET2+Ddef:ms} {conocimiento,.N4:ms} expresado {a través de,.PREP} la {producción,.N3:fs} científica {en,.PREP} {el,.DET2+Ddef:ms} {campo,.N4:ms} {de,.PREP} las {ciencias,ciencia.N5:fp} {sociales,social.A2:mp:fp} {y,.CONJC} las {humanidades,humanidad.N21:fp}.{S} Dicho modelo se {ha,haber.V212:P3s} aplicado {a,.PREP} {el,.DET2+Ddef:ms} {caso,.N4:ms} concreto {de,.PREP} la base {de,.PREP} {datos,dato.N4:mp} {ISOC,.N102:ms}, {producida,producido.PA:fs} {y,.CONJC} {distribuida,distribuido.PA:fs} {por,.PREP} {el,.DET2+Ddef:ms} {Consejo Superior de Investigaciones Científicas,.N:ms}.{S} Esta {aplicación,.N3:fs} forma parte {de,.PREP} {un,.DET1+Dind:ms} proyecto {de,.PREP} {investigación,.N3:fs} {cuyo,.ARE:ms} objetivo {principal,.A2:ms:fs} {es,ser.V211:P3s} {el,.DET2+Ddef:ms} {desarrollo,desarrollar.V1:P1s} {de,.PREP} una {interfaz,.N:ms} {de,.PREP} {realidad,.N21:fs} {virtual,.A2:ms:fs} {basada,basado.PA:fs} {en,.PREP}

{motores,motor.N2:mp} {de,.PREP} {clasificación,.N3:fs} que {utilizan,utilizar.V126:P3p} técnicas {multivariantes,multivariante.A6:mp:fp} {o,.CONJC} {redes,red.N21:fp} {neuronales,neuronal.A2:mp:fp} para {posibilitar,.V1:W} {el,.DET2+Ddef:ms} {acceso,.N4:ms} {mediante,.A6:ms:fs} {browsing,.N7:ms} {a,.PREP} los {registros,registro.N4:mp} contenidos {en,.PREP} una base {de,.PREP} {datos,dato.N4:mp}.{S} {con el fin de,.PREP} {representar,.V1:W} las relaciones {existentes,existente.A6:mp:fp} entre las {distintas,distinto.A1:fp} {materias,materia.N5:fp} que {conforman,conformar.V1:P3p} {el,.DET2+Ddef:ms} {área,.N4:ms} {de,.PREP} las {ciencias,ciencia.N5:fp} {sociales,social.A2:mp:fp} {y,.CONJC} las {humanidades,humanidad.N21:fp}, se {han,haber.V212:P3p} formado {conjuntos,conjunto.N4:mp} {de,.PREP} {documentos,documento.N4:mp} {a partir de,.PREP} los {códigos,código.N4:mp} {de,.PREP} {clasificación,.N3:fs} {utilizados,utilizado.PA:mp} {en,.PREP} la base {de,.PREP} {datos,dato.N4:mp} {ISOC,.N102:ms}.{S} {dichas,dicha.N5:fp} relaciones se {representan,representar.V1:P3p} {mediante,.A6:ms:fs} {matrices,matriz.N15:fp} {de,.PREP} {coocurrencia,.N5:fs} {de,.PREP} {números,número.N4:mp} {de,.PREP} {clasificación,.N3:fs}.{S} Las {matrices,matriz.N15:fp} se {forman,formar.V1:P3p} {siguiendo,seguir.V302:G} la estructura {jerárquica,jerárquico.A1:fs} {de,.PREP} la {propia,propio.A1:fs} {clasificación,.N3:fs}.{S} Estas {matrices,matriz.N15:fp}, una {vez,.N15:fs} {normalizadas,normalizado.PA:fp}, constituyen la entrada {de,.PREP} {un,.DET1+Dind:ms} proceso {de,.PREP} {red,.N21:fs} {neuronal,.A2:ms:fs} que se {basa,basar.V1:P3s:Y2s} {en,.PREP} los {mapas,mapa.N4:mp} afltoorganizativos {de,.PREP} Kohonen ({SOM,.N102:ms}).{S} {de,.PREP} las {distintas,distinto.A1:fp} {salidas,salida.N5:fp} que produce {el,.DET2+Ddef:ms} {simulador,.N13:ms} {de,.PREP} la {red,.N21:fs} {neuronal,.A2:ms:fs} se {utiliza,utilizar.V126:P3s:Y2s} la {matriz,.N15:fs} {de,.PREP} {tasas,tasa.N5:fp} {de,.PREP} {activación,.N3:fs} como entrada {de,.PREP} una {aplicación,.N3:fs} {ad hoc,.NL:NL} que {genera,generar.V1:P3s:Y2s} los {mapas,mapa.N4:mp} {cuyas,cuyo.ARE:fp} {topologías,topología.N5:fp} {representan,representar.V1:P3p} {el,.DET2+Ddef:ms} {conocimiento,.N4:ms} extraído {de,.PREP} la base {de,.PREP} {datos,dato.N4:mp}.{S} El resultado {de,.PREP} la {aplicación,.N3:fs} {de,.PREP} la {metodología,.N5:fs} {descrita,descrito.PA:fs} {es,ser.V211:P3s} {un,.DET1+Dind:ms} {árbol,.N2:ms} {de,.PREP} {mapas,mapa.N4:mp} que {permite,permitir.V3:P3s:Y2s} {a,.PREP} {el,.DET2+Ddef:ms} usuario {navegar,.V103:W} {a través de,.PREP} {el,.DET2+Ddef:ms} {conocimiento,.N4:ms} extraído {de,.PREP} la base {de,.PREP} {datos,dato.N4:mp}.{S} {de,.PREP} esta forma, se {genera,generar.V1:P3s:Y2s} una {interfaz,.N:ms} que {expresa,expresar.V1:P3s:Y2s} la {topología,.N5:fs}, {entendida,entendido.PA:fs} como conjunto {de,.PREP} {vecindades,vecindad.N21:fp}, {de,.PREP} las {distintas,distinto.A1:fp} {categorías,categoría.N5:fp} {temáticas,temático.A1:fp} {codificadas,codificado.PA:fp} {en,.PREP} esta base {de,.PREP} {datos,dato.N4:mp}.

Sin embargo, en el etiquetado lineal no se puede apreciar la ambigüedad en las categorías POS, porque sólo se asignan etiquetan a las unidades léxicas no-ambiguas. Como se aprecia en las tablas anteriores, los términos que pueden pertenecer a más de una categoría morfosintáctica no reciben ninguna etiqueta –como «*modelo*», «*que*», «*desarrollado*»,

«*expresado*», «*científica*», …–. Teniendo en cuenta que las entradas al *parsing* sintáctico son precisamente etiquetas sin ambigüedad, este problema se tiene que resolver antes de aplicar los analizadores sintácticos.

Como ya se indicó en el Capítulo 4, para la eliminación de la ambigüedad en las asignación de etiquetas se pueden emplear métodos estadísticos, o métodos basados en reglas. La solución que vamos a adoptar en este trabajo es utilizar procedimientos basados en reglas, para ello vamos a representar el texto en forma de *Expresiones Regulares*, o en forma de *FST*, y sobre estas representaciones realizar el análisis sintáctico. El inconveniente de la representación del texto en forma de *Expresiones Regulares* es que las secuencias del *corpus de verificación* se vuelven demasiado extensas, casi ilegibles, porque precisan muchas copias de las propias expresiones que la componen, tal y como se muestra en la Tabla 7.8.

TABLA 7.8: Resultado de la representación de las sentencias del *corpus* en forma de *Expresiones Regulares*

```
[...]

{S}{Título,título.N4:ms}
:
{S}
{Neurolsoc,.N+PR:ms}
:
{S}
{Un,un.DET1+Dind:ms}

({modelo,modelar.V1:P1s} + {modelo,.N4:ms})

{de,.PREP}

{red,.N21:fs}

{neuronal,.A2:ms:fs}

({para,.PREP} + {para,parar.V1:P3s:Y2s} )

({la,el.DET2+Ddef:fs} + {la,.PRO3:3s})

{representación,.N3:fs}

{de,.PREP}

{el,.DET2+Ddef:ms}

{conocimiento,.N4:ms}

[...]

{S}({Resumen,resumen.N27:ms} + {Resumen,resumir.V3:P3p})
```

```
:
{S}
({El,el.DET2+Ddef:ms} + {El,él.PRO3:3s})

{propósito,.N4:ms}

{de,.PREP}

({esta,.PRO8:fs} + {esta,este.DEM1:fs})

{ponencia,.N5:fs}

{es,ser.V211:P3s}

{presentar,.V1:W}

{un,.DET1+Dind:ms}

({modelo,modelar.V1:P1s} + {modelo,.N4:ms})

{de,.PREP}

{red,.N21:fs}

{neuronal,.A2:ms:fs}

({que,.CONJS} + {que,.PRORE})

({se,.PRO3:3s} + {se,.PRO6:3p})

{ha,haber.V212:P3s}

({desarrollado,.PA:ms} + {desarrollado,desarrollar.V1:P})

(
{con el fin de,.PREP}

{representar,.V1:W}

{el,.DET2+Ddef:ms}

{conocimiento,.N4:ms}

({expresado,.PA:ms} + {expresado,expresar.V1:P})

(
{a través de,.PREP}

({la,el.DET2+Ddef:fs} + {la,.PRO3:3s})

{producción,.N3:fs}

({científica,científico.A1:fs} + {científica,científico.N1:fs})

{en,.PREP}

{el,.DET2+Ddef:ms}

{campo,.N4:ms}

{de,.PREP}

({las,el.DET2+Ddef:fp} + {las,.PRO6:3p})

{Ciencias,ciencia.N5:fp}
```

```
{Sociales,social.A2:mp:fp}

{y,.CONJC}

({las,el.DET2+Ddef:fp} + {las,.PRO6:3p})

{Humanidades,humanidad.N21:fp}
.
+
{a,.PREP}

{través,.?}

{de,.PREP}

({la,el.DET2+Ddef:fs} + {la,.PRO3:3s})

{producción,.N3:fs}

({científica,científico.A1:fs} + {científica,científico.N1:fs})

{en,.PREP}

{el,.DET2+Ddef:ms}

{campo,.N4:ms}

{de,.PREP}

({las,el.DET2+Ddef:fp} + {las,.PRO6:3p})

{Ciencias,ciencia.N5:fp}

{Sociales,social.A2:mp:fp}

{y,.CONJC}

({las,el.DET2+Ddef:fp} + {las,.PRO6:3p})

{Humanidades,humanidad.N21:fp}
.
)
+
{con,.PREP}

{el,.DET2+Ddef:ms}

{fin,.N2:ms}

{de,.PREP}

{representar,.V1:W}

{el,.DET2+Ddef:ms}

{conocimiento,.N4:ms}

({expresado,.PA:ms} + {expresado,expresar.V1:P})

(
{a través de,.PREP}

({la,el.DET2+Ddef:fs} + {la,.PRO3:3s})

{producción,.N3:fs}
```

```
({científica,científico.A1:fs} + {científica,científico.N1:fs})
{en,.PREP}
{el,.DET2+Ddef:ms}
{campo,.N4:ms}
{de,.PREP}
({las,el.DET2+Ddef:fp} + {las,.PRO6:3p})
{Ciencias,ciencia.N5:fp}
{Sociales,social.A2:mp:fp}
{y,.CONJC}
({las,el.DET2+Ddef:fp} + {las,.PRO6:3p})
{Humanidades,humanidad.N21:fp}
.
+
{a,.PREP}
{través,.?}
{de,.PREP}
({la,el.DET2+Ddef:fs} + {la,.PRO3:3s})
{producción,.N3:fs}
({científica,científico.A1:fs} + {científica,científico.N1:fs})
{en,.PREP}
{el,.DET2+Ddef:ms}
{campo,.N4:ms}
{de,.PREP}
({las,el.DET2+Ddef:fp} + {las,.PRO6:3p})
{Ciencias,ciencia.N5:fp}
{Sociales,social.A2:mp:fp}
{S}
[...]
```

La otra solución, que es la que definitivamente vamos a adoptar, es representar las sentencias del *corpus* en forma de transductores gráficos, en los que cada término, dentro de una determinada sentencia delimitada, se asocia de forma perceptible a las distintas etiquetas,

hipótesis lingüísticas, como se puede comprobar en la *Sentencia 12* (Fig. 7.4), o en la *Sentencia 54* (Fig. 7.5)

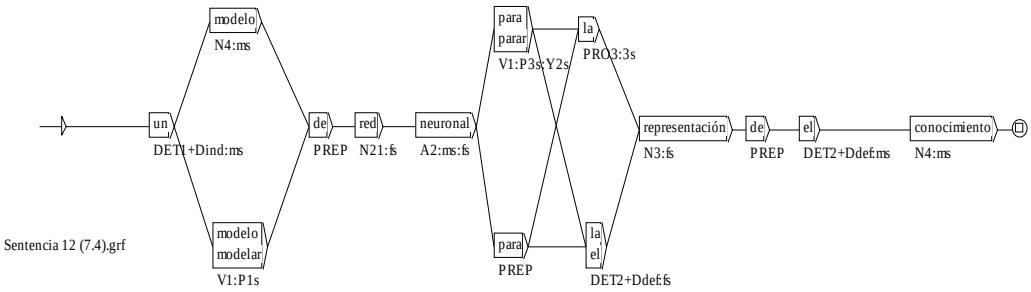


Fig. 7.4: Representación de la *Sentencia 12* en un FST gráfico

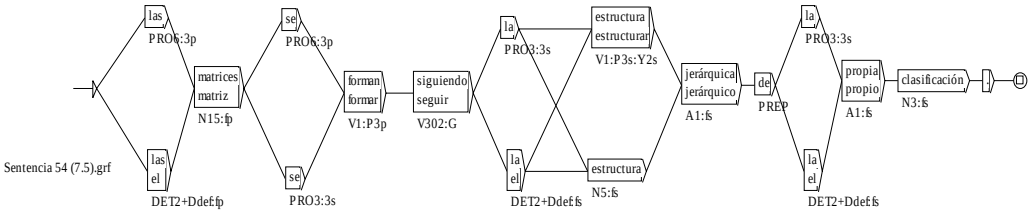


Fig. 7.5: Representación de la *Sentencia 54* en un FST gráfico

Sobre la representación de las sentencias en forma de transductores gráficos se van a aplicar los analizadores sintácticos. De esta forma, las salidas etiquetadas de los FST, que representan las distintas secuencias del *corpus* se configurarán como las entradas a los FST, que representan las gramáticas reconocedoras de los SSNN; se trata, por tanto, de una aplicación en *cascada* de los transductores gráficos. En consecuencia, como ya se ha mencionado, el proceso de identificación de estructuras lingüísticas, sobre etiquetas sin ambigüedad, va a consistir en la equiparación de las *observaciones* de los transductores, que representan las secuencias textuales, con las *entradas* de los transductores, que representan las estructuras de los SSNN.

7.3. Aplicación de las herramientas de análisis sintáctico

Después de la segmentación de los elementos lógicos y de la asignación a los términos de las categorías POS, el siguiente paso es identificar las estructuras sintagmáticas. Con este objetivo se utilizan los analizadores sintácticos que se han construido previamente y que están representados por: *1) Gramáticas Electrónicas Parciales compiladas en FST gráficos*; y *2) Estructuras de SSNN* representadas directamente en *FST gráficos*. Las bases de información sintácticas desarrolladas con estos formalismos están integradas por *137 herramientas de análisis sintáctico* que representan y reconocen un número *infinito* de estructuras sintagmáticas. La composición de estos recursos de análisis se distribuye de la forma siguiente:

a. *Recursos de análisis para la identificación de SSNN de estructura simple* (Tabla 7.9 y Tabla 7.10).

TABLA 7.9: Analizadores sintácticos de SSNN de estructura simple

Gramáticas Electrónicas Parciales	*55*
FST (agrupan las variantes de los SSNN)	*2*
Número total de analizadores	***57***

TABLA 7.10: Analizadores sintácticos de SSNN de estructura simple con *iteración* de constituyentes

Gramáticas Electrónicas Parciales	*14*
FST (agrupan las variantes de los SSNN con iteración de constituyentes)	*14*
Número total de analizadores	***28***

b. *Recursos de análisis para la identificación de SSNN de estructura compleja* (Tabla 7.11 y Tabla 7.12).

TABLA 7.11: Analizadores sintácticos de SSNN de estructura compleja

FST (agrupan las variantes de los SSNN con constituyentes preposicionales)	*8*
FST (agrupan las variantes de los SSNN con constituyentes oracionales)	*11*
Número total de analizadores	***19***

TABLA 7.12: Analizadores sintácticos de SSNN de estructura compleja con *recursividad* de constituyentes

FST (agrupan las variantes de los SSNN con recursividad de constituyentes preposicionales)	*16*
FST (agrupan las variantes de los SSNN con recursividad de constituyentes oracionales)	*17*
Número total de analizadores	***33***

Tanto las *Gramáticas Electrónicas Parciales*, compiladas a su vez en FST, y los *FST Sintácticos* anteriores se pueden aplicar al *corpus de verificación* con dos objetivos diferenciados:

1. *Localizar las estructuras de los SSNN especificados.*
2. *Realizar un análisis sintáctico de las estructuras de los SSNN reconocidos.*

En el primer caso, los recursos sintáctico se utilizan simplemente para localizar en los textos las estructuras de los SSNN especificados. En este proceso se efectúa una correspondencia entre las unidades del *corpus*, representadas *linealmente*, y la información lingüística producida por las gramáticas, representadas en transductores. Esa correspondencia puede adoptar tres *rutinas*: *a)* equiparación más corta, *shortest matches*, en la que el transductor se detiene cuando reconoce la cadena más corta; *b)* equiparación más larga, *longest matches*, en la que el transductor se detiene cuando reconoce la cadena más larga; y *c)* todas las equiparaciones, *all matches*, en la que el transductor sólo se detiene cuando reconoce todas las cadenas posibles.

La rutina que vamos a seguir en esta aplicación es la *equiparación más larga*, porque nos interesa que el transductor identifique la estructura *completa* especificada en la gramática. Simultáneamente, como las gramáticas que generan las estructuras de los SSNN están compiladas en transductores, las *observaciones*, o *salidas*, de estos transductores, cuanto se confrontan con el texto *lineal*, a partir de la *equiparación más larga*, pueden adoptar tres formas según si se asocian o no a las cadenas de entradas. Estos tipos de equiparaciones se muestran en el caso concreto de la localización de la estructura sintagmática SN_{72}, que da lugar a los siguientes resultados posibles:

- Las observaciones de los FST gráficos *no se incorporan* a las cadenas de entrada, como se muestra en la Tabla 7.13.
- Las observaciones de los FST gráficos *se incorporan* a las cadenas de entrada, como se muestra en la Tabla 7.14.
- Las observaciones de los FST gráficos *sustituyen* a las cadenas de entrada, como se muestra en la Tabla 7.15.

TABLA 7.13: Resultado de la intersección del FST SN72 con las secuencias lineales del *corpus de verificación*, en el que *no se anexan* las observaciones del transductor

acceso a información periodística
algunas regularidades de el conocimiento latinoamericano
algunos portales de la escena internacional
ámbitos de actuación prioritarios
análisis de la producción científica
bloques de información concernientes
browsing a los registros contenidos
búsqueda sobre el texto completo
cada disciplina en su carga docente
calidad de el proceso documental
Centro de Documentación Europea

centros de documentación dependientes

centros de documentación musical

centros de información deportiva

clave para la gestión integral

complejos de el Centro Hospitalar

conclusiones referentes a el sistema empleado

conocimiento de los fondos existentes

consolidación de la Red Iberoamericana

cualificación de las series documentales

datos de documentación deportiva

datos de información estructurada

descenso de la capacidad productiva

descriptores en un lenguaje documental

digitalización de documentación administrativa

dispersión de la literatura científica

documentación en derechos humanos

el ámbito de las ciencias experimentales

el análisis a texto completo

el aprendizaje de la metodología científica

el área de la vigilancia tecnológica

el conocimiento con fines documentales

conocimiento en las grandes organizaciones

el empleo de técnicas inteligentes

el mercado de contenidos digitales

el simulador de la red neuronal

el Tesauro de creación propia

el tratamiento de la información audiovisual

elaboración de dossiers personalizados

enseñanza en materia turística

escenarios de la sociedad mediática

escenarios para las industrias digitales

[…]

TABLA 7.14: Resultado de la intersección del FST SN72 con las secuencias lineales del *corpus de verificación*, en el que *se anexan las* observaciones del transductor

(SN72 (SN55 acceso)(Modif_a(SN56_información periodística)))
(SN72 (SN55 algunas regularidades)(Modif_de(SN56_el conocimiento latinoamericano)))
(SN72 (SN55 algunos portales)(Modif_de(SN56_la escena internacional)))
(SN72 (SN55 ámbitos)(Modif_de(SN56_actuación prioritarios)))
(SN72 (SN55 análisis)(Modif_de(SN56_la producción científica)))
(SN72 (SN55 bloques)(Modif_de(SN56_información concernientes)))
(SN72 (SN55 browsing)(Modif_a(SN56_los registros contenidos)))
(SN72 (SN55 búsqueda)(Modif_sobre(SN56_el texto completo)))
(SN72 (SN55 cada disciplina)(Modif_en(SN56_su carga docente)))
(SN72 (SN55 calidad)(Modif_de(SN56_el proceso documental)))
(SN72 (SN55 centros)(Modif_de(SN56_documentación dependientes)))
(SN72 (SN55 centros)(Modif_de(SN56_documentación musical)))
(SN72 (SN55 centros)(Modif_de(SN56_información deportiva)))
(SN72 (SN55 clave)(Modif_para(SN56_la gestión integral)))
(SN72 (SN55 complejos)(Modif_de(SN56_el Centro Hospitalar)))
(SN72 (SN55 conclusiones referentes)(Modif_a(SN56_el sistema empleado)))
(SN72 (SN55 conocimiento)(Modif_de(SN56_los fondos existentes)))
(SN72 (SN55 consolidación)(Modif_de(SN56_la Red Iberoamericana)))
(SN72 (SN55 cualificación)(Modif_de(SN56_las series documentales)))
(SN72 (SN55 datos)(Modif_de(SN56_documentación deportiva)))
(SN72 (SN55 datos)(Modif_de(SN56_información estructurada)))
(SN72 (SN55 descenso)(Modif_de(SN56_la capacidad productiva)))
(SN72 (SN55 descriptores)(Modif_en(SN56_un lenguaje documental)))
(SN72 (SN55 digitalización)(Modif_de(SN56_documentación administrativa)))
SN72 (SN55 Dispersión)(Modif_de(SN56_la literatura científica)))
(SN72 (SN55 documentación)(Modif_en(SN56_derechos humanos)))
(SN72 (SN55 el ámbito)(Modif_de(SN56_las ciencias experimentales)))
(SN72 (SN55 el análisis)(Modif_a(SN56_texto completo)))
(SN72 (SN55 el aprendizaje)(Modif_de(SN56_la metodología científica)))
(SN72 (SN55 el área)(Modif_de(SN56_la vigilancia tecnológica)))
(SN72 (SN55 el conocimiento)(Modif_con(SN56_fines documentales)))

(SN72 (SN55 el conocimiento)(Modif_en(SN56_las grandes organizaciones)))
(SN72 (SN55 el empleo)(Modif_de(SN56_técnicas inteligentes)))
(SN72 (SN55 el mercado)(Modif_de(SN56_contenidos digitales)))
(SN72 (SN55 el simulador)(Modif_de(SN56_la red neuronal)))
(SN72 (SN55 el Tesauro)(Modif_de(SN56_creación propia)))
(SN72 (SN55 el tratamiento)(Modif_de(SN56_la información audiovisual)))
(SN72 (SN55 elaboración)(Modif_de(SN56_dossiers personalizados)))
(SN72 (SN55 enseñanza)(Modif_en(SN56_materia turística)))
(SN72 (SN55 escenarios)(Modif_de(SN56_la sociedad mediática)))
(SN72 (SN55 escenarios)(Modif_para(SN56_las industrias digitales)))
[…]

TABLA 7.15: Resultado de la intersección del FST SN72 con las secuencias lineales del *corpus de verificación*, en el que se *sustituyen las cadenas de entrada* por las observaciones del transductor

(SN72 (SN55)(Modif (SN56_)))
(SN72 (SN55)(Modif (SN56_)))
(SN72 (SN55)(Modif (SN56_)))
(SN72 (SN55)(Modif (SN56_)))
(SN72 (SN55)(Modif (SN56_)))
(SN72 (SN55)(Modif (SN56_)))
(SN72 (SN55)(Modif (SN56_)))
(SN72 (SN55)(Modif (SN56_)))
(SN72 (SN55)(Modif (SN56_)))
(SN72 (SN55)(Modif (SN56_)))
(SN72 (SN55)(Modif (SN56_)))
(SN72 (SN55)(Modif (SN56_)))
[…]

Por otra parte, sobre los resultados de la información lingüística que aportan los transductores se pueden volver a utilizar nuevos transductores, y con los resultados obtenidos re-procesar

nuevamente el *corpus de verificación*, y así sucesivamente. La utilización en *cascada* de los transductores es una de sus propiedades más poderosas para el análisis de las unidades lingüísticas, aunque en este trabajo nos vamos a limitar sólo a las *intersección* de dos transductores: el que representa la gramática y el que representa cada sentencia del *corpus*, como vamos a describir a continuación.

Lo anterior está en relación con el primer objetivo, localizar las estructuras de los SSNN. Con respecto al segundo objetivo, los recursos sintáctico se pueden utilizar para realizar un *análisis* de las estructuras de los SSNN. Esta posibilidad nos va a permitir efectuar una correspondencia entre las secuencias del *corpus de verificación* representadas en FST, y la información lingüística producida por las gramáticas, que a su vez están representadas en transductores, en otras palabras se produce una *intersección* de transductores. La ventaja de este procedimiento es que los transductores, que representan las gramáticas se equiparan siempre a unidades no-ambiguas, y las *observaciones* de estos transductores permiten construir una *representación estructurada* de los SSNN en forma de *árboles de derivación*, en los que es posible distinguir cuál es el *núcleo* y el *modificador* de estas construcciones. A su vez, la correspondencia entre los transductores que representan las gramáticas y las sentencias del *corpus* se puede establecer de diferentes formas:

a. Los transductores gráficos se equiparan con sentencias completas.
b. Los transductores gráficos se equiparan con el principio de las sentencia.
c. Los transductores gráficos se equiparan con el final de las sentencias.
d. Los transductores gráficos se equiparan con cualquier factor dentro de las sentencias.

En este trabajo, vamos a utilizar siempre los transductores en el modo *cualquier factor de una sentencia*, porque para nuestros objetivos no es relevante distinguir si las estructuras reconocidas se localizan al principio, o final las sentencias. Con este planteamiento, el resultado de la aplicación de un transductor concreto, como SN_{72}, al texto consistirá en la aportación de la información lingüística necesaria para obtener una representación

estructurada de los sintagmas identificados, tal y como se muestra en el *árbol de derivación* (Fig. 7.6).

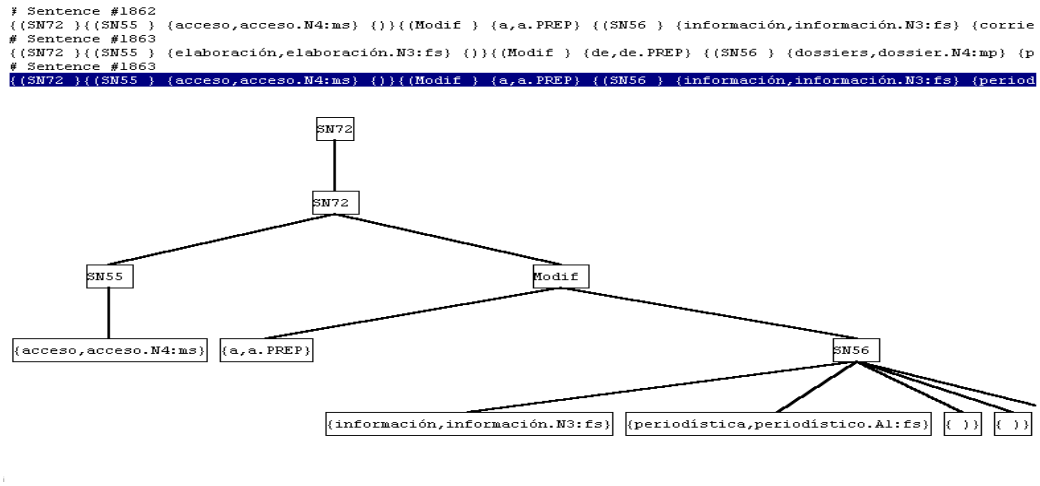


Fig. 7.6: Intersección del FST SN_{72} con las secuencias del *corpus* representadas en FST

Después de establecer cómo se pueden aplicar las bases de información lingüísticas, el paso siguiente es comprobar las *hipótesis explicativa*, que hemos propuesto para los datos lingüísticos, y evaluar los resultados de esta aplicación.

7.4. Métrica de evaluación

El método de evaluación que vamos a utilizar es una adaptación de la métrica clásica habitualmente empleada en los sistemas de RI. Este método se ha aplicado previamente para la evaluación de otro sistema de reducción de variantes lingüísticas, semejante al realizado en este trabajo. En el mencionado sistema el parámetro de *precisión* se redefine como *la proporción de variantes correctas de entre el total de variantes extraídas por el sistema* (`Correct / Total identified`), y el parámetro de *exhaustividad* se redefine como la

proporción de variantes correctas identificadas de entre el total de variantes posibles (`Correct / Total possible`) (Tzoukermann et al. 1997). Incorporando esta métrica de evaluación, nuestro objetivo ahora es medir la efectividad de las herramientas de análisis para generar índices empleando técnicas lingüísticas, esto es, pretendemos obtener resultados concluyentes sobre la validez de los analizadores léxicos y sintácticos que hemos construido, y para ello vamos a:

1. Calcular la eficacia de los analizadores léxicos para identificar y agrupar las distintas variantes léxicas en *formas regularizadas*, *lemas* o *estructuras canónicas*.

2. Calcular la eficacia de los analizadores sintácticos para identificar y agrupar las distintas variantes sintácticas en *estructuras regularizadas*, o *estructuras canónicas*.

Pero antes de realizar estos cálculos es preciso delimitar dos aspectos previos: *a)* determinar la nueva composición léxica del *corpus de verificación*, después de aplicar las herramientas de pre-procesamiento y los diccionarios electrónicos, sobre el que vamos a evaluar los parámetros de *precisión* y *exhuastividad*; y *b)* acotar las estructuras sintácticas sobre las que vamos a evaluar esos mismos parámetros.

En relación con la primera cuestión, la nueva composición del *corpus* después de someterse al pre-procesamiento y a la aplicación de los diccionarios quedaría establecida según los datos de la Tabla 7.16. En esta nueva distribución se añaden los *tokens* que surgen por la separación de las formas contractas, se aporta el *número de secuencias delimitadas*, *el numero de formas léxicas simples*, *el número de formas léxicas desconocidas*, y *el número de unidades compuestas no-ambiguas*, además del *número de dígitos* y *delimitadores*.

TABLA 7.16: Composición del *corpus de verificación* después de aplicar los diccionarios electrónicos

	tokens	*tokens differentes*
Formas léxicas simples	*18470*	*3291*
Formas léxicas simples		*2763*
Formas léxicas desconocidas		*528*
Formas léxicas compuestas	*270*	
Formas etiquetadas	*0*	*0*
Dígitos	*4082*	*10*
Delimitadores	*5508*	*16*
Secuencias delimitadas	*2916*	
Número total de tokens	***28060***	***3317***

En relación con la segunda cuestión, la delimitación de las estructuras sintácticas que vamos a evaluar, se va a localizar en el *corpus* el número de estructuras correspondientes a los SSNN. Con este objetivo se va a establecer el número de equiparaciones de las *Gramáticas Electrónicas Parciales* y los *Transductores Sintácticos* –que representan y agrupan las estructuras de los SSNN– con el *corpus de verificación*. El procedimiento que vamos a adoptar es la *equiparación más larga, longest matches,* y el resultado es el que se muestra a continuación:

- Estructuras identificadas por la equiparación de las *Gramáticas Parciales* que representan SSNN de estructura simple (Tabla 7.17).

TABLA 7.17: Número de SSNN de estructura simple

Sintagmas Nominales	Expresiones Regulares	Estructuras identificadas
SN0	DET N	1838
SN1	CUANT N	146
SN2	N	8354
SN3	N N	648
SN4	N A	852
SN5	A N	253
SN6	N PA	123
SN7	PA N	11
SN8	POS N	59
SN9	DEM N	99
SN10	CARD N	21
SN11	N CARD	0
SN12	ORD N	25
SN13	N ORD	0
SN14	DET N A	314
SN15	DET A N	103
SN16	CUANT N A	25
SN17	CUANT A N	8
SN18	POS N A	11
SN19	POS A N	26
SN20	DEM N A	4
SN21	DEM A N	5
SN22	DET N PA	60
SN23	DET PA N	1
SN24	CUANT N PA	7
SN25	CUANT PA N	0
SN26	POS N PA	0
SN27	POS PA N	0
SN28	DEM N PA	1
SN29	DEM PA N	0
SN30	DET CARD N	5
SN31	DET N CARD	0
SN32	POS CARD N	0
SN33	POS N CARD	0
SN34	DEM CARD N	0
SN35	DEM N CARD	0
SN36	DET ORD N	15
SN37	DET N ORD	0
SN38	POS ORD N	1
SN39	POS N ORD	0
SN40	DEM ORD N	0
SN41	DEM N ORD	0
SN42	DET A N A	19
SN43	DET PA N PA	0
SN44	DET A N PA	1
SN45	DET PA N A	0
SN46	CUANT DET N	6
SN47	CUANT DEM N	0
SN48	CUANT POS ORD N	0
SN49	CUANT POS N ORD	0
SN50	CUANT POS CARD N	0
SN51	DET ADV A N	1
SN52	DET ADV PA N	0
SN53	CARD ORD N	0
SN54	DET CARD ORD N	0

- Estructuras identificadas por la equiparación de FST gráficos que representan y agrupan SSNN de estructura simple (Tabla 7.18 y Tabla 7.19).

TABLA 7.18: Número de variantes de SSNN de estructura simple, agrupados en el FST SN55

Sintagmas Nominales	Expresiones Regulares	Variantes sintácticas
SN0	DET N	1838
SN1	CUANT N	146
SN2	N	8354
SN3	N N	648
SN8	POS N	59
SN9	DEM N	99
SN10	CARD N	21
SN11	N CARD	0
SN12	ORD N	25
SN13	N ORD	0
SN30	DET CARD N	5
SN31	DET N CARD	0
SN32	POS CARD N	0
SN33	POS N CARD	0
SN34	DEM CARD N	0
SN35	DEM N CARD	0
SN36	DET ORD N	15
SN37	DET N ORD	0
SN38	POS ORD N	1
SN39	POS N ORD	0
SN40	DEM ORD N	0
SN41	DEM N ORD	0
SN46	CUANT DET N	6
SN47	CUANT DEM N	0
SN48	CUANT POS ORD N	0
SN49	CUANT POS N ORD	0
SN50	CUANT POS CARD N	0
SN53	CARD ORD N	0
SN54	DET CARD ORD N	0
SN55		***7809***

TABLA 7.19: Número de variantes de SSNN de estructura simple, agrupados en el FST SN56

Sintagmas Nominales	Expresiones Regulares	Variantes sintácticas
SN4	N A	852
SN56	A N	253
SN6	N PA	123
SN7	PA N	11
SN14	DET N A	314
SN15	DET A N	103
SN16	CUANT N A	25
SN17	CUANT A N	8
SN18	POS N A	11
SN19	POS A N	26
SN20	DEM N A	4
SN21	DEM A N	5
SN22	DET N PA	60
SN23	DET PA N	1
SN24	CUANT N PA	7
SN25	CUANT PA N	0
SN26	POS N PA	0
SN27	POS PA N	0
SN28	DEM N PA	1
SN29	DEM PA N	0
SN42	DET A N A	19
SN43	DET PA N PA	0
SN44	DET A N PA	1
SN45	DET PA N A	0
SN51	DET ADV A N	1
SN52	DET ADV PA N	0
SN56		***1180***

- Estructuras identificadas por la equiparación de FST gráficos que representan y agrupan SSNN de estructura simple con *iteración* de constituyentes (Tabla 7.20).

TABLA 7.20: Número de variantes de SSNN de estructura simple con *iteración* de constituyentes

Sintagmas Nominales	Expresiones Regulares	Variantes sintácticas
SN57	DET N^{+}	7700
SN58	DET N^{+} A	967
SN59	DET N A^{+}	967
SN60	DET N^{+} A^{+}	967
SN61	DET A N^{+}	869
SN62	DET A^{+} N	263
SN63	DET A^{+} N^{+}	263
SN64	DET A N^{+} A	31
SN65	DET A^{+} N A	29
SN66	DET A N A^{+}	29
SN67	DET A^{+} N^{+} A	31
SN68	DET A N^{+} A^{+}	31
SN69	DET A^{+} N A^{+}	29
SN70	DET A^{+} N^{+} A^{+}	31

- Estructuras identificadas por la equiparación de FST gráficos que representan y agrupan SSNN de estructura compleja con constituyentes preposicionales (Tabla 7.21).

TABLA 7.21: Número de variantes de SSNN con constituyentes preposicionales

Sintagmas Nominales	Expresiones Regulares	Variantes sintácticas
SN71	N PREP N	1334
SN72	N PREP N A	376
SN73	N A PREP N A	101
SN74	N A PREP N	371
SN75	DET N^{+} PREP DET N^{+}	1328
SN76	DET N^{+} PREP DET N^{+} A	392
SN77	DET N^{+} A PREP DET N^{+} A	111
SN78	DET N^{+} A PREP DET N^{+}	403

- Estructuras identificadas por la equiparación de FST gráficos que representan y agrupan SSNN de estructura compleja con constituyentes oracionales (Tabla 7.22).

TABLA 7.22: Número de variantes de SSNN con constituyentes oracionales

Sintagmas Nominales	Expresiones Regulares	Variantes sintácticas
SN79	N PRORE V N	93
SN80	N A PRORE V N	34
SN81	DET N^+ PRORE V DET N^+	93
SN82	DET N^+ PRORE V DET N^+ A	86
SN83	DET N^+ A PRORE V DET N^+ A	35
SN84	N PREP N PRORE V N PREP N	49
SN85	DET N^+ PREP DET N^+ PRORE V DET N^+ PREP DET N^+	28
SN86	DET N^+ PREP DET N^+ PRORE V DET N^+ A PREP DET N^+ A	27
SN87	DET N^+ PREP DET N^+ A PRORE V DET N^+ PREP DET N^+	7
SN88	DET N^+ A PREP DET N^+ A PRORE V DET N^+ A PREP DET N^+ A	4
SN89	DET N^+ A PREP DET N^+ PRORE V DET N^+ PREP DET N^+	5

- Estructuras identificadas por la equiparación de FST gráficos que representan y agrupan SSNN de estructura compleja con *recursividad* de constituyentes preposicionales (Tabla 7.23).

TABLA 7.23: Número de variantes de SSNN con *recursividad* de constituyentes preposicionales

Sintagmas Nominales	Expresiones Regulares	Variantes sintácticas
SN93	N (PREP N)*	6135
SN94	N (PREP (N A) N)*	5948
SN95	N A (PREP (N A) N A)*	1076
SN96	N A (PREP (N) N A)*	1028
SN97	DET N^+ (PREP (DET N^+) DET N^+)*	6028
SN98	DET N^+ (PREP (DET N^+ A) DET N^+)*	5855
SN99	DET N^+ A (PREP (DET N^+ A) DET N^+)*	1525
SN100	DET N^+ A (PREP (DET N^+) DET N^+)*	1467
SN101	N (PREP N)* PRORE V N (PREP N)*	93
SN102	N (PREP (N A) N)* PRORE V N (PREP (N A) N)*	103
SN103	N A (PREP (N A) N A)* PRORE V N A (PREP (N A) N A)*	33
SN104	N A (PREP (N) N A)* PRORE V N A (PREP (N) N A)*	41
SN105	DET N^+ (PREP (DET N^+) DET N^+)* PRORE V DET N^+ (PREP (DET N^+) DET N^+)*	93
SN106	DET N^+ (PREP (DET N^+ A) DET N^+)* PRORE V DET N^+ (PREP (DET N^+ A) DET N^+)*	104
SN107	DET N^+ A (PREP (DET N^+ A) DET N^+)* PRORE V DET N^+ A (PREP (DET N^+ A) DET N^+)*	35
SN108	DET N^+ A (PREP (DET N^+) DET N^+ A)* PRORE V DET N^+ A (PREP (DET N^+) DET N^+ A)*	45

- Estructuras identificadas por la equiparación de FST gráficos que representan y agrupan SSNN de estructura compleja con *recursividad* de constituyentes oracionales (Tabla 7.24).

TABLA 7.24: Número de variantes de SSNN con *recursividad* de constituyentes oracionales

Sintagmas Nominales	Expresiones Regulares	Variantes sintácticas
SN109	N PRORE V (SN109)	85
SN110	N A PRORE V (SN110)	33
SN111	DET N^+ PRORE V (SN111)	85
SN112	DET N^+ A PRORE V (SN112)	35
SN113	N PREP N PRORE V (SN113)	45
SN114	DET N^+ PREP DET N^+ PRORE V (SN114)	25
SN115	DET N^+ PREP DET N^+ A PRORE V (SN115)	15
SN116	DET N^+ A PREP DET N^+ A PRORE V (SN116)	7
SN117	DET N^+ A PREP DET N^+ PRORE V (SN117)	8
SN118	N (PREP N)* PRORE V (SN118)	84
SN119	N (PREP (N A) N)* PRORE V (SN119)	95
SN120	N A (PREP (N A) N A)* PRORE V (SN120)	33
SN121	N A (PREP(N) N A)* PRORE V (SN121)	41
SN122	DET N^+ (PREP (DET N^+) DET N^+)* PRORE V (SN122)	84
SN123	DET N^+ (PREP (DET N^+ A) DET N^+)* PRORE V (SN123)	95
SN124	DET N^+ A (PREP (DET N^+ A) DET N^+)* PRORE V (SN124)	35
SN125	DET N^+ A (PREP (DET N^+) DET N^+ A)* PRORE V (SN125)	44

Una vez determinada la composición del *corpus de verificación* y el número de estructuras sintácticas identificadas, vamos a evaluar las herramientas de análisis léxico y sintáctico con los métodos cuantitativos de *precisión* y *exhaustividad*. En el caso de la evaluación de las estructuras léxicas, estos parámetros se van a aplicar al *corpus* completo, pero en el caso de las estructuras sintácticas es necesario acotar las unidades objeto de valoración, porque el número de patrones sintácticos es demasiado extenso. Para delimitar las variantes de las estructuras sintagmática que vamos a evaluar hemos adoptado el criterio de seleccionar una muestra representativa de cada tipo de estructura y que dichas estructuras tengan al menos treinta equiparación en el *corpus*:

- Variantes de *sintagmas de estructura simple* (SN_{56}).

- Variantes de *sintagmas de estructura simple con iteración de constituyentes* (SN_{70}).
- Variantes de *sintagmas de estructura compleja con constituyentes preposicionales* (SN_{73}).
- Variantes de *sintagmas de estructura compleja con constituyentes oracionale*s (SN_{81}).
- Variantes de *sintagmas de estructura compleja con recursividad de constituyentes preposicionales* (SN_{102}).
- Variantes de *sintagmas de estructura compleja con recursividad de constituyentes oracionales* (SN_{109}).

7.5. Resultados

Con el objetivo de medir la *precisión* y *exhaustividad* de las variantes léxicas identificadas con las herramientas de análisis léxico necesitamos adquirir los siguientes datos:

- El *total de las variantes léxicas reconocidas y agrupadas en lemas*. Para obtener los datos del total de variantes hemos decidido aplicar los analizadores léxicos en el modo **`Lemas`**, en el que cada palabra del *corpus* se agrupa, o relaciona, con su lema correspondiente.
- El total de las *variantes léxicas posibles reconocidas y agrupadas en lemas*. Para obtener los datos del *total de las variantes posibles* hemos decidido aplicar los analizadores léxicos en el modo **`Formas flexionadas`** + **`categorías POS`**, en el que las palabra del *corpus* se identifican con las variantes flexionadas y se les asigna la categoría POS a la que pertenecen.
- Las *variantes léxicas correctas reconocidas y agrupadas en lemas*. Para obtener los datos de las *variantes correctas* hemos optado por aplicar los analizadores léxicos en

el modo **Lemas** + **categorías POS**, en el que cada palabra del *corpus* se agrupa, o relaciona, a su correspondiente lema y a su categoría POS.

Los datos obtenidos con el procedimiento anterior son los que nos van a permitir realizar la evaluación de la efectividad de las herramientas de análisis léxico para el reconocimiento y agrupación de las variantes léxicas. La distribución de estos datos se expone a continuación:

1. *Total de variantes léxicas agrupadas* en *lemas*, según los datos de la Tabla 7.25 obtenidos de la aplicación de los analizadores léxicos en el modo **Lemas**.

TABLA 7.25: Composición del *corpus* después de sustituir cada término por el correspondiente *lema*

	tokens	*tokens differentes*
Formas léxicas sin lematizar	*4277*	*771*
Formas léxicas		*312*
Formas desconocidas		*459*
Formas léxicas lematizadas	***13786***	***1689***
Dígitos	*4082*	*10*
Delimitadores	*5499*	*16*
Número total de tokens	*27644*	*2486*

Los *tokens* de la tabla anterior están integrados básicamente por *Formas léxicas sin lematizar* y *Formas léxicas lematizadas*. Dentro de las *Formas léxicas sin lematizar* se encuentran: *a)* Formas léxicas, o términos que se pueden agrupar en diferentes lemas, o diferentes formas canónicas –como *aportes,aportar.V aportes,aporte.N proyecto,proyectar.V proyecto,proyecto.N*–; *b)* Formas desconocidas, como nombres personales, anglicismos, o errores ortográficos. Dentro de las *Formas léxicas lematizadas* se encuentra: el total de los términos que se agrupan en un único lema, aunque ese lema pueda tener asignado diferentes categorías POS –como *científico,científico.A científico,científico.N documental,documental.A documental,documental.N avance,avance.N avance,avanzar.V consulta,consulta.N consultas,consulta.N consulta,consultar.V consultas,consultar.V*–

2. *Total de las variantes léxicas posibles agrupadas en lemas*, según los datos de la Tabla 7.26 obtenidos de la aplicación de los analizadores léxicos en el modo **`Formas flexionadas`** + **`categorías POS`**.

TABLA 7.26: Composición del *corpus* después de sustituir cada término por la correspondiente *forma flexionada más la categoría POS*

	tokens	*tokens differentes*
Formas léxicas flexionadas sin lematizar ni etiquetar	*4471*	*824*
Formas léxicas		*365*
Formas desconocidas		*459*
Formas léxicas flexionadas y etiquetadas	***13592***	***2216***
Dígitos	*4082*	*10*
Delimitadores	*5499*	*16*
Número total de tokens	*27644*	*3066*

Los *tokens* de la tabla anterior están integrados por *Formas léxicas flexionadas sin etiquetar* y *Formas léxicas flexionadas y etiquetadas*, que corresponden a *Formas léxicas* y *Formas desconocidas*. Dentro de las Formas léxicas flexionadas sin etiquetar se encuentran: *a)* Términos flexionados, o variantes léxicas, a las que se les puede asignar más de una categoría POS, –como *empleados,empleado*.N *empleados,emplear*.PA *estructuras,estructura*.N *estructuras,estructurar*.V–, y *b)* Formas desconocidas, como nombres personales, anglicismos, o errores ortográficos. Dentro de las *Formas léxicas flexionadas y etiquetadas con categorías POS* se encuentra: el total de los términos flexionados o variantes léxicas posibles que se asignan a una sola categoría POS –como *neuronal,*.A *neuronales,*.A *red,*.N *redes,*.N–.

3. *Variantes léxicas correctas agrupadas en lemas*, según los datos de la Tabla 7.27, obtenidos de la aplicación de los analizadores en el modo **`Lemas + categorías POS`**.

TABLA 7.27: Composición del *corpus* después de sustituir cada término por el correspondiente *lema más la categoría POS*

	tokens	*tokens differentes*
Formas léxicas sin lematizar ni etiquetar	*4852*	*875*
Formas léxicas		*416*
Formas desconocidas		*459*
Formas léxicas lematizadas y etiquetadas con categorías POS	***13211***	***1632***
Dígitos	*4082*	*10*
Delimitadores	*5499*	*16*
Número total de tokens	*27644*	*2533*

En la tabla anterior se distinguen *Formas léxicas sin lematizar ni etiquetar* y *Formas léxicas lematizadas y etiquetadas con categorías POS*. Dentro de las *Formas léxicas sin lematizar ni etiquetar* se encuentran: *a)* Formas léxicas, o términos que se pueden agrupar en diferentes lemas, o diferentes formas canónicas, como –*aportes,aportar*.V *aportes,aporte*.N *proyecto,proyectar*.V *proyecto,proyecto*.N– y términos que se agrupan a un único lema, pero que ese lema puede tener asignado diferentes categorías POS –como *científico,científico*.A *científicos,científico*.N *documental,documental*.A *documental,documental*.N–, y *b)* Formas desconocidas, como nombres personales, anglicismos, o errores ortográficos. Dentro de las *Formas léxicas lematizadas y etiquetadas con categorías POS* se encuentran: los términos que se agrupan correctamente en un único lema y a una sola categoría POS –como *propósito,propósito*.N *motor,motor*.N *desarrollar,desarrollar*.V–.

Los resultados de la evaluación de los analizadores léxicos desarrollados en este trabajo con los datos anteriores se exponen en la Tabla 7.28, en la que se muestra que: *a) la precisión de los diccionarios representados en FST* para la identificación y agrupación de variantes léxicas es del *96.6%*, y b) *la exhaustividad de los diccionarios representados en FST* para la identificación y agrupación de variantes léxicas es del *73%.*

TABLA 7.28: Resultados de la evaluación de las variantes léxicas

Total de variantes léxicas identificadas	*1689*
Variantes correctas identificadas	*1632*
Variantes incorrectas identificadas	*57*
Total de variantes léxicas posibles	*2216*
Precisión	***96.6 %***
Exhaustividad	***73.6 %***

Con el propósito de la evaluación de las variantes sintácticas, capaces de ser identificadas con los analizadores sintácticos desarrollados, necesitamos adquirir los siguientes datos:

- El *total de las variantes sintácticas* reconocidas y agrupadas a determinadas *estructuras sintácticas canónicas*. Para obtener los datos del *total de las variantes sintácticas* hemos decidido aplicar los analizadores sintácticos en el modo *localizar patrón sintáctico por medio de FST gráficos*. La aplicación de los FST gráficos, que representan las estructuras regularizadas de los SSNN, nos va a permitir identificar y agrupar las estructuras del *corpus*, que se correspondan con las estructuras representadas en los FST. A su vez, las estructuras identificadas se indizan en la rutina *longest matches* y las *observaciones* de los FST no se van a añadir a las secuencias reconocidas.

- *El total de las variantes sintácticas posibles* reconocidas y agrupadas a determinadas *estructuras sintácticas canónicas*. Para obtener los datos del *total de las variantes sintácticas posibles* hemos decidido realizar un análisis sintáctico aplicando los FST al

texto representado también en un FST. De esta forma, vamos a obtener todas las variantes de las estructuras sintácticas que son susceptibles de análisis con los FST gráficos correspondientes. Por otra parte, las estructuras del *corpus* se van a analizar en la rutina *longest matches* y los FST se van equiparar en el modo *cualquier factor dentro de las sentencias*.

- Las *variantes sintácticas correctas* reconocidas y agrupadas a determinadas *estructuras sintácticas canónicas*. Para obtener los datos de las *variantes sintácticas correctas* aplicamos los FST sobre las secuencias del *corpus* correctamente etiquetadas, para ello aplicamos previamente los analizadores léxicos en el modo **`Formas flexionadas`** + **`categorías POS`**. Sin embargo, en el etiquetado lineal no se pueden apreciar todas las ambigüedades en la asignación de categorías, por esta razón volvemos a la representación del *corpus* por medio de transductores y sobre esta representación realizamos la intersección de las *Gramáticas Locales* (GL), que se encargan de eliminar la ambigüedad y que están representadas también en transductores. Como ya se mencionó en el Capítulo 4, las GL podrían dar lugar a errores si se aplicaran de forma indiscriminada, teniendo en cuenta esa limitación de uso se han construido sólo seis GL que son las que hemos utilizado para la desambiguación de los SSNN, objeto de esta evaluación. Con la intersección de transductores conseguimos obtener secuencias etiquetadas correctamente, sobre las que procedemos a aplicar nuevamente los analizadores sintácticos en el modo *localizar patrón sintáctico por medio de FST gráficos*, pero en este caso sobre cadenas sin ambigüedad. Todo este proceso nos llevará finalmente a la localización de los patrones sintácticos sobre cadenas sin ambigüedad y que nos van a permitir identificar las variantes *sintácticas correctas*.

Los datos obtenidos con el procedimiento anterior van a constituir la base para realizar la evaluación de la efectividad de las herramientas de análisis sintáctico para la localización y control de las variantes sintácticas, a partir de una muestra representativa de cada estructura sintagmática. La distribución de estos datos se expone a continuación:

1. *Total de variantes sintácticas* reconocidas y agrupadas en *estructuras sintácticas regularizadas*, según los datos de la Tabla 7.29 obtenidos de la aplicación de los transductores, que representan las estructuras de los SSNN, sobre las secuencias del *corpus*. La equiparación se establece en el modo *localizar patrón sintáctico por medio de FST* y en la rutina *longest matches*

TABLA 7.29: Número de variantes sintácticas identificadas

FST sintácticos	SN56	SN70	SN73	SN81	SN102	SN109
Total de variantes sintácticas identificadas	*1168*	*30*	*101*	*93*	*103*	*85*

2. *Total de variantes sintácticas posibles* reconocidas y agrupadas en *estructuras sintácticas regularizadas*, según los datos de la Tabla 7.30 obtenidos a partir de un análisis sintáctico de las estructuras de los SSNN sobre el texto representado también en un FST. Con este procedimiento vamos a obtener todas las variantes de las estructuras sintácticas que se pueden analizar con los FST gráficos. Por otra parte, las estructuras del *corpus* se van a analizar en la rutina *longest matches* y los FST se van a equiparar en el modo *cualquier factor dentro de las sentencias.*

TABLA 7.30: Número de variantes sintácticas posibles

FST sintácticos	SN56	SN70	SN73	SN81	SN102	SN109
Total de variantes sintácticas posibles	*1772*	*53*	*178*	*177*	*361*	*138*

3. *Total de variantes sintácticas correctas* reconocidas y agrupadas en *estructuras sintácticas regularizadas*, según los datos de la Tabla 7.31 obtenidos de la aplicación de los transductores, que representan las estructuras de los SSNN, sobre las secuencias del *corpus*. La equiparación se establece en el modo *localizar patrón sintáctico por medio de FST* y en la rutina *longest matches*. Sin embargo, en este caso la equiparación se

establece sin ambigüedad porque hemos realizado un análisis léxico previo en el modo **`Formas flexionadas`** + **`categorías POS`** y se ha eliminando la ambigüedad por medio de GL.

TABLA 7.31: Número de variantes sintácticas correctas

FST sintácticos	SN56	SN70	SN73	SN81	SN102	SN109
Total de variantes sintácticas correctas	*1160*	*30*	*98*	*87*	*91*	*80*

Los resultados de la evaluación con los datos anteriores se exponen en la Tabla 7.32 , en la que se muestra la tasa de *precisión* y *exhaustividad* de los transductores sintácticos. A continuación, se calcula el *promedio* de ambas tasas y el resultado de este cálculo nos permite hacer la siguiente valoración: *a) la precisión de los FST sintácticos* para reconocer y agrupar variantes sintácticas es de un *promedio* de *0.95*, y *b) la exhaustividad de los FST sintáctivos* para reconocer y agrupar variantes sintácticas es de un *promedio* del *0.51*, tal y como se expone en la Tabla 7.33.

TABLA 7.32: Tasas de *precisión* y *exhaustividad* de los FST sintácticos

	Precisión	***Exhaustividad***
SN56	*0.99*	*0.65*
SN70	*1*	*0.56*
SN73	*0.97*	*0.55*
SN81	*0.93*	*0.49*
SN102	*0.88*	*0.25*
SN109	*0.94*	*0.58*

TABLA 7.33: Resultados de la evaluación de los FST sintácticos

	Precisión	***Exhaustividad***
promedio	***0.95***	***0.51***

7.6. Discusión

Los resultados de la evaluación de los analizadores léxicos y sintácticos muestran que las variantes correctamente identificadas son muy elevadas, pero también ponen de manifiesto que estos analizadores no son capaces de seleccionar cuál es el análisis correcto para un determinado término, o una determinada estructura sintagmática, en los casos de ambigüedad. Además de la ambigüedad, la falta de cobertura, el *infraanálisis* y el *sobreanálisis* son las deficiencias mayores que se han presentado después de haber realizado esta evaluación. Con el objetivo de realizar un análisis pormenorizado de las cuestiones anteriores, a continuación vamos a tratar de interpretar los resultados obtenidos, distinguiendo, en primer lugar, los errores de los analizadores léxicos y, en segundo lugar, los errores de los analizadores sintácticos. Por último, a la vista de estos resultados, vamos a proponer, en cada caso, posibles soluciones que mejorarían en el futuro las herramientas que hemos desarrollado.

En la aplicación de los analizadores léxicos es fundamental tener en cuenta que sólo se lematizan las variantes que se corresponden con un lema. Dicho de otro modo, cuando una mismas variante se puede agrupar a lemas distintos –como *proyecto,proyectar.V proyecto,proyecto.N*– no se lematiza. Aún así, la tasa de *precisión* de los analizadores léxicos muestra un pequeño porcentaje de errores del *3.4%*. Estos errores se deben a que, aunque dos o más variantes se correspondan con un único lema, se trata de variantes distintas porque tienen asignadas etiquetas distintas –como científico,científico.A científicos,científico.N –. Según esto, la primera deficiencia del sistema es que existen variantes léxicas que se lematizan erróneamente, provocando que determinadas variantes se vinculen al mismo lema, cuando en realidad pertenecen a lemas distintos, porque cada uno de ellos está en relación con una etiqueta POS. A su vez, como los analizadores, compilados en autómatas, pueden trabajar en el modo de análisis o en el modo de generación, a este tipo de errores de los sistemas que manejan PLN se les denomina comúnmente *errores de sobreanálisis*, o *de sobregeneración*.

La evaluación de la *exhaustividad* de los analizadores léxicos ha dado como resultado que podamos afirmar que estas herramientas consiguen reconocer correctamente el *73.6%* de las

variantes léxicas posibles. Esto significa que si el total de las variantes posibles susceptibles de agruparse en lemas es de *2216*, con los analizadores basados en técnicas de estado-finito conseguimos reducir correctamente las variantes a *1632*, con lo cual se consigue reducir un *26.4%* del total de variantes. La interpretación de este resultado es que, aunque pueda parecer una contradicción, en este caso la tasa de *exhaustividad* mejoraría si el índice fuera más *bajo*, justo lo contrario que sucede cuando se utiliza esta métrica para la evaluación de otros sistemas. Por esta razón, en la evaluación de los sistemas que utilizan modelos lingüísticos más que de *exhaustividad* se debería de hablar de *cobertura* de los analizadores, pero para no crear confusión, porque el término *cobertura* es demasiado amplio, vamos a seguir utilizando el término *exhaustividad*.

Teniendo en cuenta la observación anterior, la interpretación estricta de este parámetro supone que cuanto más *alta* sea la tasa de *exhaustividad* esto significará que menos variantes léxicas se reducen del total de variantes posibles. En otras palabras, si el total de las variantes posibles agrupadas en lemas es de *2216* y los analizadores fueran capaces de reducirlas hipotéticamente a *2216*, el resultado sería del *100%*, lo cual se traduciría en que los analizadores léxicos no conseguirían reducir las variantes léxicas en absoluto. El hecho de que estas herramientas reduzcan las variantes a un *26.4%* se puede considerar un resultado satisfactorio, más teniendo en cuenta que muchas variantes sólo se reducen en la *proporción* $1 \Rightarrow 1$, como preposiciones o adverbios, en la *proporción* $2 \Rightarrow 1$ [`singular/plural ⇒ singular`], o en la *proporción* $4 \Rightarrow 2$ [`singular/plural-masculino/femenino ⇒ masculino-singular`]; aunque algunas variantes, como las verbales se reduzcan en la *proporción* $61 \Rightarrow 1$.

Los resultados que hemos obtenido se podrían mejorar en el caso de la *precisión*, aplicando los analizadores léxicos en el modo **`Lemas`** + **`categorías POS`**, con lo cual se lograría atenuar la degradación de la *precisión* eliminando los errores que antes hemos mencionada y que están desencadenados por la agrupación de dos o más variantes a un único lema, cuando en realidad se trata de variantes distintas, porque cada una tiene asignada una etiqueta POS. Frente a esto, los resultados obtenidos en la *exhaustividad* no se pueden mejorar, o por ahora

no somos capaces de hacerlo. El hecho de que no podamos *reducir* el porcentaje obtenido, se debe a que no se trata de errores de los analizadores de estado-finito sino de limitaciones propias de estas herramientas: *no son capaces de reducir más porque están diseñados para reducir simplemente al lema*. A pesar de lo anterior, como hemos mencionado, se puede considerar que estos resultados son bastante satisfactorios.

Sin embargo, el gran problema de los analizadores léxicos de estado-finito es otro: cuando hemos evaluado los datos obtenidos lo hemos hecho siempre sobre variantes léxicas que se lemantizan, esto es, hemos evaluado los datos a partir de las variantes que se pueden agrupar a un lema. Pero, en los casos de ambigüedad en los que las variantes se puedan agrupar a distintos lemas, los analizadores no lematizan y, por tanto, no son capaces de reducir las variantes. El número de *formas sin lematizar* se expone nuevamente en la Tabla 7.34, en la que no hemos tenido en cuenta otras unidades que no se lematizan, como son las *formas desconocidas* –integradas por errores ortográficos, nombres personales, o términos en otras lenguas–. Todo esto nos lleva a constatar que la principal deficiencia de los analizadores, desarrollados con técnicas de estado-finito, para identificar y agrupar variantes son las unidades que precisamente no son capaces de analizar: el *infraanálisis*. Para este problema no hay solución con estas técnicas, no obstante algo a favor de este procedimiento es que no es tan frecuente, como pueda parecer, que una misma variante léxica se vincule a lemas distintos.

TABLA 7.34: Formas léxicas del *corpus* que no se reducen a lemas

Formas léxicas sin lematizar	*312*
Formas léxicas flexionadas sin lematizar ni etiquetar	*365*
Formas léxicas sin lematizar ni etiquetar	*416*

En relación con la evaluación de las herramientas de análisis sintácticos, los resultados muestran que el promedio de *precisión* de los analizadores para agrupar las variantes sintácticas es del *0.95*. Este elevado índice de *precisión* se debe a que hemos realizado dos procesos:

1. Para identificar las variantes sintácticas correctas hemos procedido a etiquetar las secuencias de entrada en el modo **Formas flexionadas** + **categorías POS**, con lo cual se ha eliminado la ambigüedad entre forma simples y compuestas, o dicho de otro modo, cadenas de entrada como «*Consejo Superior de Investigaciones Científicas*» se etiquetan como formas simples y no como una secuencia de formas simples. Por lo tanto, el analizador sintáctico identifica una forma simple y no lo que hipotéticamente constituiría la estructura compleja de un SN.

2. Para eliminar la ambigüedad en el etiquetado hemos recurrido al diseño de *Gramáticas Locales* (GL) que se *intersectan* al texto convertido previamente en FST –como es el caso de una de las estructuras recuperadas con el FST SN56 (Fig. 7.7), y para la que se ha construido una GL específica (Fig. 7.8)–.

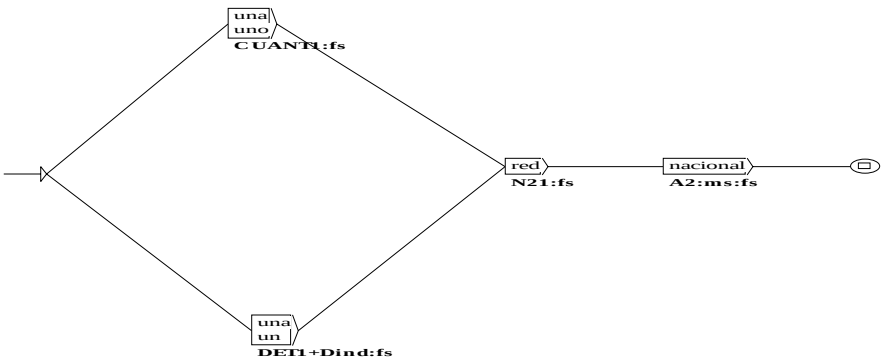


Fig. 7.7: Estructura ambigua recuperada con el FST SN56

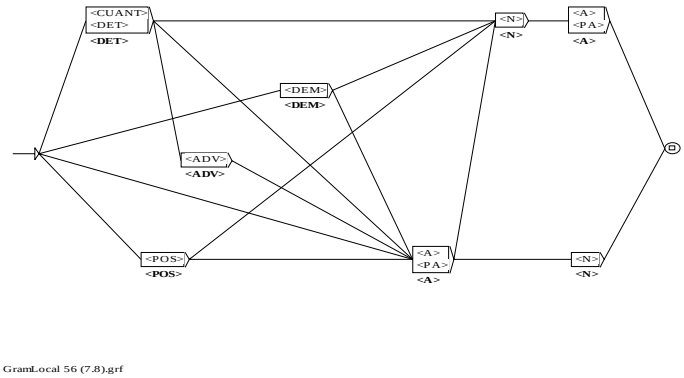


Fig. 7.8: GL construida para eliminar la ambigüedad del FST SN56

Se puede deducir que el alto índice de *precisión* que hemos obtenido se basa fundamentalmente en la correcta etiquetación de las estructuras –de ahí el éxito de otros anotadores automáticos de *corpus*, como los etiquetadores estocásticos, basados en técnicas estadísticas–. Frente a esto, el procedimiento que hemos utilizado es recurrir a una estrategia, o *heurística*, para solucionar este problema consistente en: una vez localizada una estructura, las GL se encargan de asociar una *restricción* a las cadenas reconocidas, por medio de *reglas de dos-partes* representadas también de forma gráfica en un transductor. Los resultados han demostrado que las GL son bastante eficaces como estrategia para la eliminación de la ambigüedad, y junto con la anotación automática con técnicas de estado-finito, constituyen un procedimiento menos complejo que los que utilizan modelos estocásticos.

Por otra parte, los resultados obtenidos en la evaluación de la exhaustividad no son nada favorables, el promedio es sólo del *0.51*. Sin embargo, esto no quiere decir que los FST sintácticos sólo sean capaces de recuperar el *49%* de las estructuras posibles, sino que hay deficiencias que, en este caso, sí creemos que se pueden mejorar. Los causas que degradan la *exhaustividad* son de dos tipos, aunque tienen las mismas consecuencias:

1. Tras esta evaluación hemos comprobado que en el lexicón hay entradas en las que coinciden algunas de sus variantes flexionadas y etiquetas POS, aunque con lemas distintos –como {técnicas,técnica.N5:fp} y {técnicas,técnico.N1:fp}, o {informática,informática.N10:fs} y {informática,informático.N1:fs})–. Teniendo en cuenta que las entradas a los analizadores sintácticos son las etiquetas POS, y en este caso es la misma con distintas especificaciones, el comportamiento de las herramientas de análisis ante este hecho es recuperar o identificar varias estructuras posibles para una misma estructura sintáctica. Esto ha dado como resultado un *sobreanálisis* de las secuencias identificadas y, en consecuencia que aumente del número de estructuras posibles. No obstante, consideramos que ésta no es la principal causa de la degradación de la *exhaustividad*, porque este fenómeno lingüístico tampoco es muy frecuente y, de cualquier forma, esta deficiencia se podría corregir en el futuro simplemente suprimiendo una de las entradas del lexicón.

2. Debido al funcionamiento inherente de los analizadores de estado-finito, basados en la identificación de las *posibles combinaciones válidas en un espacio de búsqueda*, esto puede provocar en muchas ocasiones que una misma estructura tenga múltiples análisis correctos. En el caso concreto de la identificación de las variantes sintácticas del FST SN70 se recuperaría una misma estructura que se podría analizar de distintas formas posibles:

$SN_{70} \rightarrow$ DET A^+ N^+ A^+

los principales sitios web académicos

(SN70 {los,.DET} {principales,.A} {sitios,.N} {web,.N} {académicos,.A})

(SN70 {principales,.A} {sitios,.N} {web,.N} {académicos,.A})

Este fenómeno es el que realmente degrada el promedio de *exhaustividad*, pero nos encontramos ante la misma situación anterior: no se trata de un error de los analizadores sintácticos sino de una característica propia de las técnicas de estado-finito. La particularidad de los analizadores desarrollados con estas técnicas es que las búsquedas se

establecen en un *espacio combinatorio*, provocando que en muchas veces se encuentren distintas elecciones válidas. Todo esto tiene como consecuencia que se multipliquen las estructuras válidas, o dicho de otro modo, que se produzca *sobreanálisis*.

En relación con el problema del *sobreanálisis*, hay que señalar que, además de que una misma estructura se pueda analizar de distinta forma por un solo FST, se puede dar el caso de que una misma estructura se analice con distintos FST. Es necesario aclarar que nuestro objetivo no ha sido construir herramientas de análisis sintáctico que se puedan aplicar simultáneamente en el reconocimiento de estructuras, sino que hemos creado FST gráficos que sean capaces de analizar y agrupar determinados *patrones sintácticos*, correspondientes a las estructuras de los SSNN. Si estas herramientas no se utilizaran de forma aislada, y se aplicaran de forma simultánea agravarían el problema del *sobreanálisis*.

En el supuesto de que se aplicaran simultáneamente FST que analizaran las mismas estructuras, el problema del *sobreanálisis* se habría desencadenado no sólo porque se hubiera aumentado el *espacio de búsqueda* sino porque se hubiera producido una *explosión de análisis*, con combinaciones todas correctas y posibles:

los principales sitios web académicos

(SN70 {los,.DET} {principales,.A} {sitios,.N} {web,.N} {académicos,.A})
(SN70 {principales,.A} {sitios,.N} {web,.N} {académicos,.A})

(SN55 {sitios,.N} {web,.N})
(SN56 {web,.N} {académicos,.A})
(SN58 {sitios,.N} {web,.N} {académicos,.A})
(SN62 {los,.DET} {principales,.A} {sitios,.N})
/.../

Para mejorar el promedio de *exhaustividad*, y evitar de paso que se produzca esa hipotética *explosión de análisis*, tendríamos que utilizar un procedimiento que consiguiera reducir las combinaciones posibles. Un método para restringir la profusión de análisis válidos y posibles

podría ser calcular las probabilidades de transición de las distintas secuencias de categorías y determinar la secuencia más probable con la que se puede etiquetar una estructura. Estas probabilidades se incorporarían a un *Autómata de Estado-Finito Probabílistico*, o *Modelo de Markov*, con el cual conseguiríamos reducir el espacio de búsqueda. A su vez, todo este proceso conseguiría mejorar el índice de *exhaustividad* de los analizadores sintácticos que hemos construido.

Aunque el desarrollo del procedimiento expuesto anteriormente exceda los objetivos de este trabajo, sí creemos conveniente exponer brevemente cómo se podría llevar a cabo, más después de la valiosa información estadística que nos puede aportar ahora el análisis léxico del *corpus de verificación*, por medio de los diccionarios electrónicos que hemos desarrollado, y que nos van a permitir trabajar con datos reales. Así, para determinar la secuencias de etiquetas mas probables para el análisis de una estructura como «los principales sitios web académicos», teniendo en cuenta que se puede analizar con las siguientes secuencias de etiquetas válidas:

DET A N N A
A N N A
N N
N A
N N A
DET A N
/.../

tendríamos que obtener una *matriz de probabilidad de transición* entre etiquetas que se incorporarían a un *Autómata Probabilístico*. Una forma de calcular la probabilidad de transición podría ser mediante el cálculo de la *probabilidad condicionada*, o *modelo n-gramas*, como se expuso en el Capítulo 4. En este caso, para obtener los *bigramas* localizamos en el *corpus*, previamente etiquetado, cada uno de las posibles combinaciones por parejas ($n = 2$) que se pueden formar con las etiquetas anteriores, y registramos las frecuencias con la que aparecen (Tabla 7.35).

TABLA 7.35: Parejas de etiqueta-etiqueta

	frecuencias
(N, N)	*644*
(N, A)	*847*
(DET, A)	*160*
(A, N)	*243*

Además, sería preciso obtener la probabilidad de que una estructura se inicie con una determinada categoría, para ello el delimitador de inicio de sentencia, < ^ >, se debe considerar una unidad del recuento (Tabla 7.36). De la misma forma, sería necesario obtener la frecuencia total de aparición de cada etiqueta (Tabla 7.37)

TABLA 7.36: Parejas de delimitador-etiqueta

	frecuencias
(^ , N)	*2316*
(^ , A)	*41*
(^, DET)	*68*

TABLA 7.37: Frecuencia de etiquetas

	frecuencias
(N)	*8245*
(A)	*1552*
(DET)	*2073*

Con todos estos datos se podría calcular la *probabilidad condicionada* de que dos etiquetas *co-ocurran* en el *corpus* por medio de la siguiente fórmula:

$$P\left(t_i / t_{i-1}\right) = \frac{f\left(t_{i-1}, t_i\right)}{f\left(t_{i-1}\right)}$$

$$P\left(N \mid \wedge\right) = \frac{f\left(\wedge, N\right)}{f\left(N\right)} = 0.28$$

$$P\left(A \mid \wedge\right) = \frac{f\left(\wedge, A\right)}{f\left(A\right)} = 0.026$$

$$P\left(DET \mid \wedge\right) = \frac{f\left(\wedge, DET\right)}{f\left(DET\right)} = 0.033$$

$$P\left(N \mid N\right) = \frac{f\left(N, N\right)}{f\left(N\right)} = 0.078$$

$$P\left(A \mid N\right) = \frac{f\left(N, A\right)}{f\left(N\right)} = 0.1$$

$$P\left(A \mid DET\right) = \frac{f\left(DET, A\right)}{f\left(DET\right)} = 0.077$$

$$P\left(N \mid A\right) = \frac{f\left(A, N\right)}{f\left(A\right)} = 0.16$$

Si incorporamos estas probabilidades a un AFD obtenemos un *Autómata Probabilístico*, o *cadena de Markov* (Fig. 7.9)

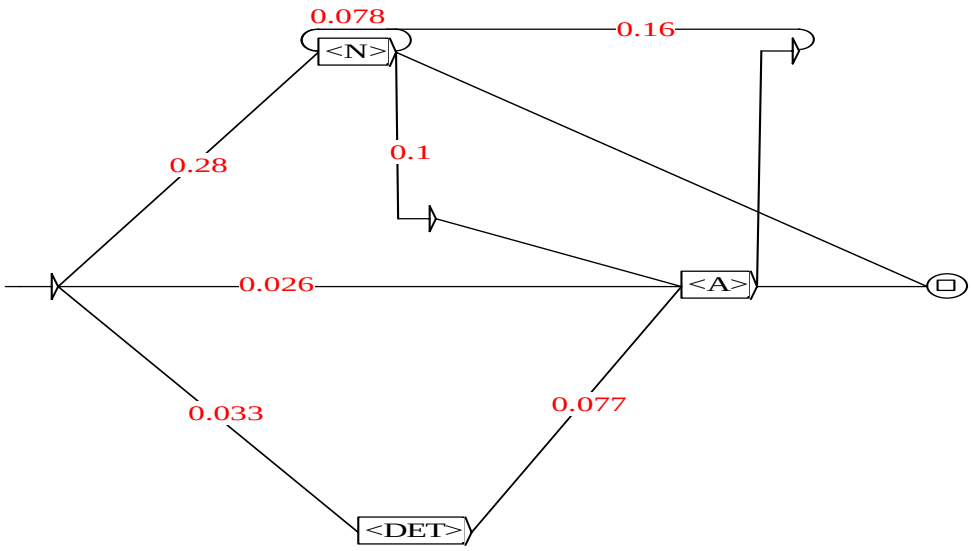


Fig. 7.9: Incorporación de *probabilidades de transición* a un AFD

Para determinar la probabilidad de cada secuencia de categorías bastaría con multiplicar las probabilidades de transición, tal y como aparece en la Tabla 7.38, y seleccionar la secuencia con la probabilidad mayor.

TABLA 7.38: Probabilidades de transición de etiquetas

	probabilidades de transición
DET A N N A	*0.033 * 0.077* 0.16 * 0.078 * 0.1= 0.0000031*
A N N A	*0.026 * 0.16 * 0.078* 0.1=0.0000032*
N N	*0.28 * 0.078=0.02*
N A	*0.28 * 0.1=0.028*
N N A	*0.28 * 0.078 * 0.1=0.0021*
DET A N	*0.033 * 0.077 * 0.16=0.0004*

Según los resultados anteriores la combinación de etiquetas más probable sería N A y el FST que reconoce esta estructura sería el que tendríamos que seleccionar, en el caso de que la misma estructura se pudiera analizar por varios transductores, esto es, en caso de ambigüedad. De esta forma, la utilización de *modelos probabilísticos* nos permitiría obtener las estructuras sintácticas más frecuentes y podríamos reducir el *espacio de búsqueda combinatoria* de los analizadores sintácticos que hemos construido, lo que nos permitiría reducir a su vez el número de estructuras válidas y posibles susceptibles de análisis, y en consecuencia conseguiríamos mejorar el índice de *exhaustividad*.

Capítulo 8

CONCLUSIONES Y DESARROLLOS FUTUROS

Al comienzo de este trabajo nos propusimos dos objetivos básicos: *a)* Crear bases de información lingüísticas: *Diccionarios* y *Gramáticas electrónicas*, que se pudieran utilizar por mecanismos automáticos de análisis para reconocer y agrupar variantes léxicas y sintácticas, y *b)* Comprobar las *hipótesis* formuladas para explicar los datos lingüísticos y, en relación con esto, evaluar la aplicación de las bases de información desarrolladas. Para la representación de las bases de información hemos utilizado métodos de la *Teoría de Lenguajes y Gramáticas Formales*, y como mecanismo automático de reconocimiento hemos utilizado una aplicación informática desarrollada a partir de *Técnicas de Estado-Finito*. Por su parte, la comprobación empírica de las *hipótesis* se ha realizado aplicando los analizadores, diseñados con las bases de información, sobre un *corpus de verificación* y, a continuación, los resultados de esa aplicación se han evaluado por medio de una adaptación de la métrica tradicional empleada en los sistemas de RI.

Las conclusiones que se han extraído del *primer objetivo*, relacionado con el diseño de las bases de conocimiento, son las siguientes:

1. Se han creado bases de información léxicas consistentes en: *Diccionarios electrónicos*, representados en FST gráficos. La composición de los *Diccionarios electrónicos* es la siguiente: *61659 entradas en total*. Estas entradas se distribuyen del modo siguiente: *4772 Formas canónicas*, que se expanden automáticamente en *60511 Formas flexionadas simples*, y *1148 Formas compuestas*, que se expanden automáticamente en *1148 Formas flexionadas compuestas*. La construcción de las bases de información léxicas nos ha permitido cumplir los siguientes objetivos específicos: (*i*) vincular *Formas flexionadas* a *Formas canónicas*; (*ii*) crear diccionarios de *Formas compuestas*; (*iii*) asignar etiquetas POS a las *Formas léxicas*; (*iv*) resolver los problemas de las *irregularidades* entre *Formas flexionadas* y *Formas canónicas*, por medio de la representación directa de las irregularidades en FST gráficos.

2. Se han creado bases de información sintácticas consistentes en *137 herramientas de análisis* representadas en FST gráficos, que generan todas las estructuras posibles de los *Sintagmas Nominales*, a modo de *patrones sintácticos*. La composición de las bases de información sintácticas es la siguiente: *85 Gramáticas electrónicas*, configuradas como *Gramáticas electrónicas parciales*; y *52 Transductores de Estado-Finito*. La construcción de las bases de información sintácticas nos ha permitido cumplir los siguientes objetivos específicos: (*i*) vincular las *variantes de las estructuras sintácticas* a *estructuras sintácticas canónicas*; (*ii*) resolver el problema de la *recursividad* de determinadas estructuras sintagmáticas, que no se pueden representar con mecanismos de estado-finito, por medio de la construcción de *Gramáticas Regulares* equivalentes; (*iii*) resolver el problema de la *ambigüedad* por medio de la construcción de *Gramáticas Locales*, que se encargan de asociar restricciones a las cadenas reconocidas.

Las conclusiones extraídas a partir del *segundo objetivo*, relacionado con la comprobación empírica de las hipótesis por medio de la aplicación de los analizadores y con la evaluación de los resultados de esa aplicación, son las siguientes:

3. La aplicación de los analizadores léxicos y sintácticos sobre el *corpus de verificación* nos ha permitido comprobar las *hipótesis explicativas* que hemos propuesto para la representación de las variantes lingüísticas. De este modo, la aplicación de los *Diccionarios electrónicos* sobre el *corpus de verificación* nos ha permitido (*i*) comprobar empíricamente cómo los analizadores léxicos identifican y agrupan las *Formas flexionadas* en *Formas canónicas*. A su vez, la aplicación de las *Gramáticas electrónicas parciales* sobre el *corpus de verificación* nos ha permitido (*ii*) comprobar empíricamente cómo los analizadores sintácticos identifican las *estructuras sintácticas* y las agrupan a *estructuras sintácticas canónicas*.

4. La evaluación de la aplicación de analizadores léxicos para el reconocimiento y agrupación de las variantes léxicas nos ha llevado a las siguientes conclusiones: (*i*) los analizadores léxicos son muy precisos, *96.6%*; (*ii*) los analizadores léxicos consiguen fusionar, o *conflactar*, las variantes en un *26.4%* del total de las variantes posibles, y este resultado se puede considerar satisfactorio; (*iii*) los analizadores basados en técnicas de estado-finito tienen una limitación: sólo agrupan las variantes que se puedan vincular a una sola *forma controlada*, y en caso de ambigüedad no lematizan, en consecuencia, la principal deficiencia de los analizadores léxicos es la falta de cobertura, o *infraanálisis*, y esta deficiencia no se puede mejorar porque se trata de una limitación inherente de este procedimiento: *los analizadores no son capaces de reducir las mismas variantes que se puedan agrupar a distintos lemas*.

5. La evaluación de la aplicación de los analizadores sintácticos para la identificación y agrupación de las variantes sintácticas nos ha llevado a las siguientes conclusiones: (*i*) los analizadores sintácticos son muy precisos, *95%*; (*ii*) el promedio de

exhaustividad de los analizadores es bajo, *51%*, este índice está provocado porque una misma estructura puede tener múltiples análisis correctos, lo cual aumenta el número de *estructuras posibles* que se pueden analizar correctamente por los transductores; (*iii*) el problema del *sobreanálisis* se puede solucionar reduciendo el *espacio de búsqueda combinatoria* de los analizadores sintácticos mediante *modelos probabilísticos*, este proceso consistiría en *incorporar probabilidades* a los resultados obtenidos con los analizadores sintácticos, lo cual reduciría el número de estructuras válidas y posibles susceptibles de análisis, que se traduciría en un aumento del índice de *exhaustividad*

Teniendo en cuenta que el objetivo potencial de la construcción de los analizadores desarrollados con técnicas lingüísticas es identificar variantes lingüísticas que se puedan usar como entradas a los índices de los sistemas de RI, podemos concluir a modo de síntesis que: (*i*) los analizadores léxicos basados en técnicas de estado-finito constituyen una *técnica de conflación* de variantes adecuada para la generación de índices, aunque hemos comprobado que tienen un problema de *infraanálisis;* (*ii*) los analizadores sintácticos basados en técnicas de estado-finito constituyen una t*écnica de conflación* de variantes adecuada, si se resuelve el problema del *sobreanálisis* aplicando modelos probabilísticos.

Además de las conclusiones anteriores, podemos añadir que la notación de *Expresiones Regulares* ha resultado ser una metodología muy eficaz para la descripción exacta de patrones lingüísticos. La falta de un procedimiento adecuado en este punto provoca que la descripción de las expresiones se realice muchas veces de forma vaga e imprecisa , y esto tiene como consecuencia que nunca se pueda llegar a una descripción lo suficientemente precisa. En este trabajo hemos utilizado la tecnología de Expresiones Regulares para intentar describir correctamente las variantes léxicas y sintácticas, las posibilidades de esta tecnología nos han permitido contar con una metodología eficaz para la descripción correcta de expresiones lingüísticas y su posterior implementación en autómatas y transductores. En el futuro intentaremos explotar estas posibilidades en tareas como: *control de términos de dominio*, *control de nombres personales*, o *descripción de términos especializados*.

Además, el desarrollo de los analizadores con esta técnica nos ha permitido obtener una valiosa herramienta de etiquetación de cadenas. Como hemos comprobado, los cálculos estadísticos de las estructuras léxicas y sintácticas se hacen siempre sobre *corpora etiquetados*, por esta razón los lexicones computacionales constituyen la base de cualquier investigación sobre la distribución de dichas estructuras. En relación con esto, un objetivo a corto plazo, será ampliar la cobertura de los analizadores léxicos, con lo cual conseguiremos etiquetar un número mayor de términos. Así mismo, pretendemos aumentar las entradas de los *Diccionarios electrónicos* para que se puedan utilizar en dominios de conocimiento amplios, e intentaremos desarrollar diccionarios en otras lenguas. Por último, en trabajos posteriores nos propondremos desarrollar herramientas de análisis semántico, que nos permitan comprobar la eficacia de este tipo de analizadores para localizar y agrupar *variantes semánticas*.

Bibliografía

1. Abney, S. 1996. "Partial Parsing via Finite-State Cascades." *Proceedings of the ESSLLI'96 Robust Parsing Workshop*. Praga.

2. Aho, A. V., R. Sethi, y J. Ullman. 1990. *Compiladores: principios, técnicas y herramientas*. México: Addison Wesley Iberoamericana.

3. Aït-Mokhtar, S. y J. L. Rodrigo Mateos. 1995. "Segmentación y análisis morfológico de textos en español utilizando el sistema SMORPH." *Procesamiento del Lenguaje Natural* 17:29-41.

4. Alcoba, S. 1991. "Morfología del verbo español." Pp. 87-119 en *Lenguajes Naturales y Lenguajes Formales*, vol. VII, C. Martin Vide (ed.). Barcelona: Publicaciones de la Universidad.

5. Ambadiang, T. 1990. "Contribución al estudio del verbo español: un análisis morfosemántico." *ALH* VI:29-63.

6. ———. 1994. *La morfología flexiva*. Madrid: Taurus.

7. Antworth, E. L. 1990. "PC-KIMMO: a Tow-Level Processor for Morphological Analysis." *Occasional Publications in Academic Computing nº* 16. Dallas, Texas: Summer Institute of Linguistics.

8. Antworth, E. L. 1995. "User's Guide to PC-KIMMO Version 2" [Página web]. Disponible en http://www.sil.org/pckimmo/v2/doc/guide.html.

9. Appelt, D., J. Hobbs, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Myers, y M. Tyson. 1995. "SRI International FASTUS System: MUC-6 Test Results and Analysis." Pp. 237-48 en *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. San Mateo, CA: Morgan Kaufmann.

10. Arampatzis, A. T., T. Tsoris, C. H. A. Koster., y P. van der Weide. 1998. "Phrase-Based Information Retrieval." *Information Processing & Management* 34(6):693-707.

11. Beesly, K. R. y L. Karttunen. 2000. "Finite-State Non-Concatenative Morphotactics." Pp. 1-12 en *SIGPHON-2000, Proceedings of the Fifth Workshop of the ACL Special Interest Group in Computational Phonology*.

12. Bikel, D. M., S. Miller, R. Schwartz, y R. Weischedel. 1997. "Nymble: a High-Performance Learning Name-Finder." Pp. 194-201 en *Proceedings of ANLP-97*.

13. Black, A. 1989. "Finite State Machines from Feature Grammars." *International Workshop on Parsing Technologies*. Pittsburgh, PA.: Carnegie Mellon University.

14. Booth, T. L. 1967. *Sequential Machines and Automata Theory*. New York: John Wiley.

15. Brill, E. 1992. "A Simple Rule Based Part-of-Speech Tagger." Pp. 152-55 en *Third Conference on Applied Natural Language Proceedings*. Trento, Italia.

16. ———. 1994. "Some Advances in Transformation-Based Part of Speech Tagging." *Proceedings of AAAI*.

17. Brookshear, J. G. 1993. *Teoría de la Computación*. Wilmington, Delaware, E.U.A.: Addison Wesley Iberoamericana.
Notas: versión en español de la obra *Theory of Computation. Formal Languages, Automata, and Complexity*, publicada originariamente en inglés por The Benjamin/Cummings Publishing Company, Inc., Redwood City, California, E.U.A, 1989.

18. Buckley, C., J. Alland, y G. Salton. 1995. "Automatic Routing and Retrieval Using SMART: TREC-2." *Information Processing & Management* 31(3):315-26.

19. Chanod, J. P. y P. Tapanainen. 1995. "Comparing a Statistical and a Contraint-Based Method." *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL'95)*. Dublín.

20. Chen, H., Y. Zhang, y A. L. Houston. 1998. "Semantic Indexing and Searching Using a Hopfield Net." *Journal of Information Science* 24.

21. Chomsky, N. 1965. *Aspects of the Theory of Syntax*. Cambridge, Massachusetts: Massachusetts Institute of Technology.
Notas: Trad. esp. *Aspectos de la teoría de la sintaxis*. Madrid, Aguilar, 1976.

22. ———. 1957. *Syntactic Structures*. La Haya: Mouton.

23. Chomsky, N. y M. Halle. 1968. *The Sound Pattern of English*. New York: Harper and Row.

24. Church, K. 1988. "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text." *Second Conference on Applied Natural Language Processing*. Austin, Texas.

25. Church, K., S. Young, y G. Bloothcroft (eds.). 1996. *Corpus-Based Methods in Language and Speech*. Dordrecht: Kluwer Academic Publishers.

26. Church, K., W. Gale, P. Hanks, y D. Hindle. 1991. "Using Statistics in Lexical Analysis." Pp. 165-78 en *Lexical Acquisition: Using On-Line Resources to Build a Lexicon*, U. Zernik (ed.). Hillsdale, New Jersey: Lawrence Earlbaum.

27. Cohen, D. I. A. 1991. *Introduction to Computer Theory*. New York: John Wiley.

28. Coseriu, E. 1981. *Lecciones de lingüística general*. Madrid: Gredos.

29. Croft, W. B., H. R. Turtle, y D. D. Lewis. 1991. "The Use of Phrases and Structured Queries in Information Retrieval." *Proceedings, SIGIR 1991*.

30. Croft, W. B. y J. Xu. 1995. "Corpus-Specific Stemming Using Word Form Co-Occurence." *Proceedings for the Fourh Annual Symposium on Document Analysis and Information Retrieval*. Las Vegas, Nevada.

31. Cutting, D., J. Kupiec, J. Pedersen, y P. Sibun. 1992. "A Practical Part-of-Speech Tagger." *Third Conference on Applied Natural Language Processing*. Trento, Italia.

32. Dawson, J. L. 1974. "Suffix Removal for Word Conflation." *Bulletin of the Association for Literary & Linguistic Computing* 2(3):33-46.

33. Defense Advanced Research Projects Agency. 1998. "Proceedings of the Seventh Message Understanding Conference (MUC-7)" [Página web]. Disponible en http://www.saic.com.

34. DeGroot, M. H. 1989. *Probabilidad y estadística*. 2ª ed. México: Addison-Wesley Iberoamericana.

35. Doszkocs, T. E., J. Reggia, y X. Lin. 1990. "Connectionist Models and Information Retrieval." *Annual Review of Information Science and Technology (ARIST)* 25:209-60.

36. Evans, D. A. y C. Zhai. 1996. "Noun-Phrase Analysis in Unrestricted Text for Information Retrieval." Pp. 17-24 en *Proceedings of the 34th Annual Meeting of Association for Computational Linguistics*. Santa Cruz: University of California.

37. Evans, D. A., N. Milic-Frayling, y R. G. Lefferts. 1996. "CLARIT TREC-4 Experiments." *The Fourth Text REtrieval Conference (TREC-4)*, D. K. Harman (ed.). National Institute of Standards and Technology (NIST) Special Publication 500-236.

38. Fagan, J. L. 1989. "The Effectiveness of a Nonsyntactic Approach to Automatic Phrase Indexing for Document Retrieval." *Journal of the American Society for Information Science* 40(2):115-32.

39. Faitelson-Weiser, S. 1993. *Les suffixes formateurs d'djectifs en Espagnol moderne. Rapport de Recherche (2 Vol.).* Université Laval, Quebec: Departement de langues et linguistique.

40. Farwell, D., S. Helmreich, y M. Casper. 1995. "SPOST: a Spanish Part of Speech Tagger." *Procesamiento del Lenguaje Natural* 17:42-53.

41. Floyd, R. W. y R. Beigel. 1993. *The Language of Machines: an Introduction to Computability and Formal Languages*. New York: Computer Science Press.

42. Frakes, W. B. 1992. "Stemming Algorithms." Pp. 131-61 en *Information Retrieval: Data Structures and Algorithms*, W. B. Frakes y R. Baeza-Yates (eds.). Englewood Cliffs, NJ: Prentice-Hall.

43. Frakes, W. B. y R. Baeza-Yates. 1992. *Information Retrieval: Data Structures and Algorithms*. Englewood Cliffs,NJ: Prentice-Hall.

44. Gaizauskas, R., T. Wakao, K. Humphreys, H. Cunningham, y Y. Wills. 1995. "University of Sheffield: Description of the LaSIE System as Used for MUC-6." Pp. 207-20 en *Proceedings of the Sixth Message Understanding Conference (MUC-6).* San Mateo, CA*:* Morgan Kaufmann.

45. Gaizauskas, R. y Y. Wilks. 1998. "Information Extraction: Beyond Document Retrieval." *Journal of Documentation* 54(1):70-105.

46. Gala, N. 1999. "Using the Incremental Finite-State Architecture to Create a Spanish Shallow Parser." *Procesamiento del Lenguaje Natural* 25:75-82.

47. Glickman, O. y R. Jones. 1999. "Examining Machine Learning for Adaptable End-to-End Information Extraction Systems." *Proceedings of the AAAI-99 Workshop on Machine Learnig for Information Extraction*. American Association for Artificial Intelligence.

48. Grishman, R. 1986. *Computational Linguistics*. Cambridge: Cambridge University Press.

49. ———. 1997. "Information Extraction: Techiques and Challenges." *Information Extraction*, M. T. Pazienza (ed.). Roma: Springer-Verlag.

50. ———. 1995. *TIPSTER Architecture Design Document Version 1.52 (Tinman Architecture)*. New York University: Department of Computer Science.

51. Grishman, R. y J. Sterling. 1993. "Description of the Proteus System As Used for MUC-5." Pp. 181-94 en *Proceedings of the Fifth Message Understanding Conference (MUC-5).* San Mateo, CA: Morgan Kaufmann.

52. Guerrero Bote, V. P., F. de Moya Anegón, y V. Herrero Solana. 2002. "Document Organization Using Kohonen's Algorihm." *Information Processing and Management* 38:79-89.

53. Harman, D. K. 1997. *The Sixth Text REtrieval Conference (TREC-6)*. National Institute of Standards and Technology (NIST) Special Publication 500-240.

54. Harris, J. W. 1987. "The Accentual Patterns of Verb Paradigms in Spanish." *NLLT* 5:61-90. Notas: Trad. esp. *La estructura silábica y el acento en español*. Madrid, Visor, 1991

55. Harris, Z. S. 1951. *Methods in Structural Linguistics*. Chicago: University of Chicago Press.

56. Hobbs, J. R. 1991. "Description of the TACITUS System as Used for MUC-3." Pp. 200-206 en *Proceedings of the Third Message Understanding Conference (MUC-3).* San Mateo, CA: Morgan Kaufmann.

57. *Proceedings of the Fifth Message Understanding Conference (MUC-5).* 1993.San Mateo, CA: Morgan Kaufman.

58. Hobbs, J. R., D. E. Appelt, M. Tyson, B. Mabry, y D. Israel. 1992. "SRI International: Description of the FASTUS System Used for MUC-4." Pp. 268-75 en *Proceedings of the Fourth Message Understanding Conference (MUC-4).* San Mateo, CA: Morgan Kaufmann.

59. Hockett, C. 1961. "Linguistic Elements and Their Relations." *Language* 37:29-53.

60. Hopcroft, J. E. y J. D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Massachusetts: Addison-Wesley.

61. Hull, D. 1996. "Stemming Algorithms - A Case Study for Detailed Evaluation." *Journal of the American Society for Information Science* 47(1):70-84.

62. Humphreys, K., G. Demetriou, y R. Gaizauskas. 2000. "Two Applications of Information Extraction to Biological Science Journal Articles: Enzyme Interactions and Protein Structures." Pp. 505-16 en *Proceedings of the Pacific Symposium on Biocomputing (PSB-2000).*

63. Humphreys, K., R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, y Y. Wilks. 1999. "Description of the University of Sheffield LaSIE-II System As Used for MUC-7." *Proceedings of the Seventh Message Understanding Conference (MUC-7).* San Mateo, CA: Morgan Kaufmann.

64. Jacobs, P. F., G. Krupka, L. R. Rau, M. Mauldin, T. Mitamura, T. Kitani, I. Sider, y L. Childs. 1993. "The TIPSTER/SHOGUN Project." *Proceedings of the TIPSTER Phase I Final Meeting* . San Mateo, CA: Morgan Kaufmann.

65. Jacobs, P. S. 1995. "Text Interpretation: Extracting Information." *Survey of the State of the Art in Human Language Technology*, R. A. Cole. National Science Foundation.

66. Jacquemin, C. 1996. "What Is the Tree That We See Through the Windows: a Linguistic Approach to Windowing and Term Variation." *Information Processing & Management* 32(4):445-58.

67. Jacquemin, C. y E. Tzoukermann. 1999. "NLP for Term Variant Extraction: Synergy Between Morphology, Lexicon, and Syntax." Pp. 25-74 en *Natural Language Information Retrieval* , T. Strzalkowski (ed.). Dordrecht: Kluwer.

68. Jacquemin, C., J. L. Kavans, y E. Tzoukermann. 1997. "Expansion of Multi-Word Terms for Indexing and Retrieval Using Morphology and Syntax." *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL'97).* Madrid.

69. Johnson, C. D. 1972. *Formal Aspects of Phonological Description.* La Haya: Mouton.

70. Jurafsky, D. y J. H. Martin. 2000. *Speech and Language Processing: an Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. New Jersey: Prentice-Hall.

71. Kaplan, R. M. 1995. "Finite State Technology." *Survey of the State of the Art in Human Language Technology*, R. A. Cole (ed.). National Science Foundation.

72. Kaplan, R. M. y M. Kay. 1981. "Phonological Rules and Finite-State Transducers." *Linguistic Society of America Meeting Handbook, Fifty-Sixth Annual Meeting.* New York.

73. ———. 1994. "Regular Models of Phonological Rule Systems." *Computational Linguistics* 20(3):331-78.

74. Karttunen, L. 2000. "Applications of Finite-State Transducers in Natural-Language Processing." *Proceedings of CIAA-2000. Lecture Notes in Computer Science*. Springer Verlag.

75. ———. 1994. "Constructing Lexical Transducers." *Proceedings of the Fifteenth International Conference on Computational Linguistics*. Kyoto, Japan: Coling 94.

76. ———. 1996. "Direted Replacement." *The Proceedings of the 34rd Annual Meeting of the Association for Computational Linguistics.* Santa Cruz, California: ACL-96.

79. ———. 1991. "Finite-State Constraints." *Proceedings of the International Conference on Current Issues in Computational Linguistics*. Penang, Malaysia: Universiti Sains Malaysia.

80. ———. 1983. "KIMMO: a General Morphological Processor." *Texas Linguistics Forum* 22:217-28.

81. ———. 1995. "The Replace Operator." *The Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics* . Boston. Massachusetts: ACL-95.

82. Karttunen, L., J. P. Chanod, G. Grefenstette, y A. Schiller. 1996. "Regular Expressions for Language Engineering." *Natural Language Engineering* 2(4):305-28.

83. Karttunen, L., R. M. Kaplan, y A. Zaenen. 1992. "Two-Level Morphology With Composition." *Proceedings of the 14th International Conference on Computational Linguistics (COLING'92).*

84. Katz, J. J. y J. A. Fodor. 1963. "The Structure of a Semantic Theory." *Language* 39:170-210.

85. Katz, J. J. y P. M. Postal. 1964. *An Integrated Theory of Linguistic Descriptions*. Cambridge, Massachusetts: M. I. T. Press.

86. Kelley, D. 1995. *Teoría de autómatas y lenguajes formales*. Madrid: Prentice Hall.

87. Klenee, S. C. 1956. "Representation of Events in Nerve Nets and Finite Automata." *Automata Studies*. Princeton , N.J.: Princeton University Press.

88. Koskenniemi, K. 1983. "Two-Level Morphology: a General Computational Model for Word-Form Recognition and Production." Department of General Linguistics, University of Helsinki.

89. Krovetz, R. 1993. "Viewing Morpholoy As an Inference Process." Pp. 191-202 en *Proceedings of the 16th ACM/SIGIR Conference*, R. Korfhage et al. (ed.). New York: Association for Computing Machinery.

90. Krupka, G. R. 1995. "Description of the SRA System As Used for MUC-6." Pp. 221-36 en *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. San Mateo, CA: Morgan Kaufmann.

91. Kupiec, J. 1992. "Robust Part-of-Speech Tagging Using a Hidden Markov Model." *Computer Speech and Language* 6:225-42.

92. Leek, T. R. 1997. "Information Extraction Using Hidden Markov Model." Master's thesis, UC, San Diego.

93. Lehnert, W. 1996. "Information Extraction" [Página web]. Disponible en http://www-nlp.cs.umass.edu.

94. Lennon, M., D. S. Peirce, B. D. Tarry, y P. Willett. 1981. "An Evaluation of Some Conflation Algorithms for Information Retrieval." *Journal of Information Science* 3:177-83.

95. Lewis, D. D. 1992. "Text Filtering in MUC-3 and MUC-4." Pp. 51-66 en *Proceedings of the Fourth Message Understanding Conference (MUC-4).* San Mateo, CA: Morgan Kaufmann.

96. Lin, D. 1995. "Description of the PIE System As Used for MUC-6." Pp. 113-26 en *Proceedings of the Sixth Message Understanding Conference (MUC-6).* San Francisco: Morgan Kaufmann.

97. Lovins, J. B. 1968. "Development of a Stemming Algorithm." *Mechanical Translation and Computational Linguistics* 11:22-31.

98. Marcus, M. P., B. Santorini, y M. A. Marcinkiewicz. 1993. "Building a Large Annotated Corpus of English: the Penn Treebank." *Computational Linguistics* 19(2):313-30.

99. Márquez, L. y L. Padró. 1997. "A Flexible POS Tagger Using an Automatically Acquired Language Model." *Proceedings of the Association for Computational Linguistics (ACL'97)* .

100. Matthews, P. H. 1965. "The Inflection Component of a Word-and-Paradigm Grammar." *JL* 1:139-71.

101. ———. 1974. *Morphology. An Introduction to the Theory of Word-Structure*. Cambridge: University Press.
Notas: Trad. esp. *Morfología: una introducción a la teoría de la estructura de palabra*. Madrid, Paraninfo, 1980

102. McCallum, A., D. Freitag, y F. Pereira. 2000. "Maximum Entropy Markov Models for Information Extraction and Segmentation." *ICML-2000*.

103. McCallum, A., K. Nigam, J. Rennie, y K. Seymore. 1999. "Building Domain-Specific Search Engines With Machine Learning Techniques." *Proceedings of AAAI Spring Symposium on Intelligent Agents in Cyberspace* .

104. Mealy, G. H. 1955. "A Method for Synthesizing Sequential Circuits ." *Bell System Technical Journal* 34:1045-79.

105. Mighetto, D. 1992. "Notas sobre la noción de aspecto en un marco de clasificación de verbos (Vb) y sustantivos verbales (Sv)." *Voz y Letra* 3(1):69-100.

106. Miller, G. 1990. "Wordnet: An On-Line Lexical Database." *International Journal of Lexicography* 3(4).

107. Mohri, M. 1997. "Finite-State Transducers in Language and Speech Processing." *Computational Linguistics* 23(2):1-42.

108. ———. 1996. "On Some Applications of Finite-State Automata Theory to Natural Language Processing." *Journal of Natural Language Engineering* 2:1-20.

109. ———. 1997. "String-Matching With Automata." *Nordic Journal of Computing* 4(2):217-31.

110. Mohri, M., F. Pereira, y M. Riley. 1996. "Weighted Automata in Text and Speech Processing." *Proceedings of the 12th Biennial European Conference on Artificial Intelligence (ECAI-96), Workshop on Extended Finite State Models of Language*. Budapest, Hungría: ECAI.

111. Mohri, M. y R. Sproat. 1996. "An Efficient Compiler for Weighted Rewrite Rules." *The Proceedings of the 34rd Annual Meeting of the Association for Computational Linguistics*. Santa Cruz, California: ACL-96.

112. Moore, E. F. 1956. "Gedankenexperiments on Sequential Machines." Pp. 129-53 en *Automata Studies*, C. E. Shannon y J. McCarthy (eds.). Princeton, New York: Princeton University Press.

113. Moya Anegón, F. de., V. Herrero Solana, y V. Guerrero Bote. 1998. "La aplicación de redes neuronales artificiales (RNA) a la Recuperación de Información." *Anuario SOCADI de Documentación e Información* 42(2).

114. Mulbregt, P. V., I. Carp, L. Gillick, y J. Yamron. 1998. "Text Segmentation and Topic Tracking on Broadcast News Via a Hidden Markov Model Approach." *Proceedings of the International Conference on Spoken Language Processing* .

115. Multi-Lingual Theory and Technology Group (MLTT). 1994. *Xerox Finite-State Morphology Tools*. Meylan, France: Xerox Research Centre Europe (XRCE). Notas: http://www.xrce.xerox.com/research/mltt/home.en.html

116. Muslea, I. 1999. "Extraction Patterns for Information Extractions Task: A Survey." *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*. American Association for Artificial Intelligence.

117. Naur, P. 1963. "Revised Report on the Algorithmic Language Algol 60." *Communications of the A.C.M.* 6(1):1-17.

118. Paice, C. D. 1990. "Another Stemmer." *ACM SIGIR Forum* 24(3):56-61.

119. ———. 1996. "Method for Evaluation of Stemming Algorithms Based on Error Counting." *Journal of the American Society for Information Science* 47(8):632-49.

120. Pereira, F. 1990. "Finite-State Aproximations of Grammars." Pp. 20-25 en *Proceedings of the Third DARPA Speech and Natural Language Workshop*. Hidden Vallery, Pennsylvania: Defense Advanced Research Projects Agency, Morgan Kaufmann.

121. Pereira, F. y D.H.D. Warren. 1980. "Definitive Clause Grammars for Language Analysis: a Survey of the Formalism and a Comparison With Augmented Transition Networks." *Artificial Intelligence* 13(3):231-78.

122. Popovic, M. y P. Willett. 1992. "The Effectiveness of Stemming for Natural-Language Access to Slovene Textual Data." *Journal of the American Society for Information Science* 43(5):384-90.

123. Porter, M. F. 1980. "An Algorithm for Suffix Stripping." *Program* 14:130-137.

124. Pustejovsky, J. 1992. "The Adquisition of Lexical Semantic Knowledge From Large Corpora." *Proceedings of the Fifth DARPA Speech Recognition*. San Mateo, CA: Morgan Kaufmann.

125. Rabiner, L. 1989. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." *Proceedings of the IEEE* 77 (2):257-85.

126. Rau, L. F., P. J. Jacobs, y U. Zernik. 1989. "Information Extraction and Text Summarization Using Linguistic Knowledge Acquisition." *Information Processing & Management* 35(4):419-28.

127. Rich, E. y K. Knight. 1992. *Artificial Intelligence*. 2ª ed. New York: McGraw-Hill.

128. Riloff, E. y R. Jones. 1999. "Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping." *Proceedings of the Sixteenth National Conference on Artificial Intelligence*. American Association for Artificial Intelligence.

129. Riloff, E. y J. Shepherd. 1997. "A Corpus-Based Approach for Building Semantic Lexicons." Pp. 117-24 en *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing* .

130. Ritchie, G. D., G. J. Russell, A. W. Black, y S. G. Pulman. 1991. *Computational Morphology Practical Mechanisms for the English Lexicon*. Cambridge, MA: The MIT Press.

131. Robertson, A. M. y P. Willett. 1998. "Applications of N-Grams in Textual Information Systems." *Journal of Documentation* 54(1):48-69.

132. Roche, E. 1993. "Analyse syntaxique transformationelle du français par transducteurs et lexique-grammaire." Université, Paris.

133. ———. 1996. "Finite-State Transducers: Parsing Free and Frozen Sentences." *Proceedings of the 12th Biennial European Conference on Artificial Intelligence (ECAI-96), Workshop on Extended Finite State Models of Language*. Budapest, Hungría: ECAI.

134. Roche, E. y Y. Schabes. 1995. "Deterministic Part-Of-Speech Tagging With Finite State Transducers." *Computational Linguistics* 21(2):227-53.

135. ———. 1997. *Finite State Language Processing*. Cambridge, Massachusetts: MIT Press.

136. Ross, B. J. 2000. "Probabilistic Pattern Matching and the Evolution of Stochastic Regular Expressions." *Applied Intelligence* 13(3):285-300.

137. Rumelhart, D. 1977. *Introduction to Human Information Processing*. New York: Wiley.

138. Salton, G. 1989. *Automatic Text Processing: the Transformation, Analysis and Retrieval of Information Computer*. Reading, MA: Addison-Wesley.

139. ———. 1980. "The SMART System 1961-1976: Experiments in Dynamic Document Processing." Pp. 1-36 en *Encyclopedia of Library and Information Science*, vol. 28.

140. Salton, G., C. Buckley, y M. Smith. 1990. "On the Application of Syntactic Methodologies in Automatic Text." *Information Processing & Management* 26(1):73-92.

141. Salton, G. y M. J. McGill. 1983. *Introduction to Modern Information Retrieval*. New York: McGraw-Hill.

142. Sánchez-León, F. 1995. "Desarrollo de un etiquetador morfosintáctico para el español." *Procesamiento del Lenguaje Natural* 17:14-28.

143. Saussure, F. d. 1916. *Cours de linguistique générale*. Paris. Notas: Trad. esp. *Curso de lingüística general*. Buenos Aires, Losada, 1945

144. Savoy, J. 1993. "Stemming of French Words Based on Grammatical Categories." *Journal of the American Society for Information Science* 44(1):1-9.

145. Schwarz, C. 1990. "Automatic Syntactic Analysis of Free Text." *Journal of the American Society for Information Science* 41(6):408-17.

146. Seymore, K., A. McCallum, y R. Rosenfeld. 1999. "Learning Hidden Markov Model Structure for Information Extraction." *AAAI'99 Workshop on Machine Learning for Information Extraction* .

147. Shannon, C. E. y W. Weaver. 1949. *The Mathematical Theory of Communication*. Urbana. Notas: Trad. esp. Teoría Matemática de la Comunicación. Forja, Madrid, 1981

148. Sheridan, P. y A. F. Smeaton. 1992. "The Application of Morpho-Syntactic Language Processing to Effective Phrase Matching." *Information Processing & Management* 28(3):349-69.

149. Silberztein, M. 2000. "INTEX: an FST Toolbox." *Theorical Computer Science* 231(1):33-46.

150. ———, (ed.) 1996. *Proceedings of the First INTEX User's Workshop*. Paris: LADL. Notas: Laboratoire d'Automatique Documentaire et Linguistique, University of Paris

151. ———. 1999. "Text Indexation With INTEX." *Computers and the Humanities* 33(3):265-80.

152. Smadja, F. 1993. "Retrieving Collocations From Text: XTRACT." *Computational Linguistics* 19(1).

153. Smadja, F. y K. McKeown. 1990. "Automatically Extracting and Representing Collocations for Language Generation." *Proceedings of the 28th Annual Meeting of the ACL*. Pittsburgh, PA: Association for Computational Linguistics.

154. Soderland, S. y W. Lehnert. 1994. "Wrap-Up: a Trainable Discourse Module for Information Extraction." *Journal of Artificial Intelligence Research* 2:131-58.

155. Sparck Jones, K. y J. I. Tait. 1984. "Automatic Search Term Variant Generation." *Journal of Documentation* 40(1):50-66.

156. Strzalkowski, T. 1996. "Natural Language Information Retrieval." *Information Processing & Management* 31(3):397-417.

157. ———, (ed.) 1999. *Natural Language Information Retrieval*. Dordrecht: Kluwer.

158. ———. 1997. "Robust Text Processing in Automated Information Retrieval." *Readings in Information Retrieval*, K. Sparck Jones y P. Willett (eds.). San Francisco, CA: Morgan Kauffmann Publishers.

159. Strzalkowski, T., F. Lin, J. Wang, y J. Pérez-Carballo. 1999. "Evaluating Natural Language Processing Techniques in Information Retrieval: a TREC Perspective." Pp. 113-45 en *Natural Language Information Retrieval*, T. Strzalkowski (ed.). Dordrecht: Kluwer Academic Publishers.

160. Strzalkowski, T. y J. Perez Carballo. 1995. "Natural Language Information Retrieval: TREC-4 Report." *Proceedings of the FourthText REtrieval Conference (TREC-4)*. NIST Special Publication 500-236.

161. Tzoukermann, E. y D. R. Radev. 1995. "Use of Weighted Finite State Transducers in Part of Speech Tagging." *Natural Language Engineering* 1(1).

162. Tzoukermann, E., J. L. Kavans, y C. Jacquemin. 1997. "Effective Use of Natural Language Processing Techniques for Automatic Conflation of Multi-Word Terms: the Role of Derivational Morphology, Part of Speech Tagging, and Shallow Parsing." *Proceedings 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*. Philadelphia, Pennsylvania.

163. Viterbi, A. J. 1967. "Error Bounds for Convolutional Codes and an Asymtotically Optimum Decoding Algorithm." *IEEE Transactions on Information Theory IT* 13:260-267.

164. Voutilainen, A. 1997. "A short introduction to NPtool" [Página web]. Disponible en http://www.lingsoft.fi/doc/nptool/intro/.

165. Weischedel, R. 1995. "Description of the PLUM System As Used for MUC-6." Pp.55-70 en *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. San Francisco, CA: Morgan Kaufmann.

166. Wilks, Y. 1997. "Information Extraction As a Core Language Technology." *Information Extraction*, M. T. Pazienza (ed.). Berlin: Springer.

167. Winograd, T. 1983. *Language as a Cognitive Process: Syntax*. Reading, MA: Addison-Wesley.

168. Woods, W. A. 1970. "Transition Network Grammars for Natural Language Analysis." *Communications of the A.C.M.* 13:391-606.

169. Xu, J. y B. Croft. 1998. "Corpus-Based Stemming Using Co-Occurrence of Word Variants." *ACM Transactions on Information Systems* 16(1):61-81.

170. Yangarber, R. y R. Grishman. 1998. "NYU: Description of the Proteus/PET System As Used for MUC-7 ST." *Proceedings of the Seventh Message Understanding Conference (MUC-7)* Morgan Kaufmann.