International Journal of Computational Intelligence Systems, Vol. 5, No. 6 (November, 2012), 1068-1088

A Relational Model for the Possibilistic Valid-time Approach

Jose Enrique Pons, Nicolás Marín, Olga Pons,[1]
Christophe Billiet, Guy de Tré [2]

[1] Department of Computer Science and Artificial Intelligence, University of Granada
C/ Periodista Daniel Saucedo Aranda, S/N,E-18071
Granada, Spain
E-mail: jpons,nicm,opc@decsai.ugr.es
[2] Department of Telecommunications and Information Processing, Ghent University
Sint-Pietersnieuwstraat 41,
B-9000, Ghent, Belgium.
E-mail: Christophe.Billiet, Guy.De.Tre@ugent.be

Received 16 July 2012
Accepted 18 July 2012

Abstract

In real world, it is very common that some objects or concepts have properties with a time-variant or time-related nature. Modelling this kind of objects or concepts in a (relational) database schema is possible, but time-variant and time-related attributes have an impact on the consistency of the entire database and must be appropriately managed. Therefore, temporal database models have been proposed to deal with this problem in the literature. Time can be affected by imprecision, vagueness and / or uncertainty, since existing time measuring devices are inherently imperfect. Additionally, human beings manage time using temporal indications and temporal notions, which may also be imprecise. However, the imperfection in human-used temporal indications is supported by human interpretation, whereas information systems need appropriate support in order to accomplish this task. Several proposals for dealing with such imperfections when modelling temporal data exist. Some of these proposals transform the temporal data into a compact representation but there is not a formal model for managing and handling uncertainty regarding temporal information. In this work we present a novel model to deal with imprecision in valid-time databases together with the definition and implementation of the data manipulation language, *DML*.

Keywords: fuzzy, temporal, database

1. Introduction

The concept of time is very complex to handle and interpret [27,45], though it is very natural and omnipresent in real world data. As information systems attempt the modelling of natural objects, concepts or processes, they often require modelling temporal aspects or concepts. Thus, several proposals have arisen to obtain theoretical models that allow the modelling or representation of time [3,8].

A very specific type of information systems are database systems. A database contains data representing real objects or concepts. In real world, some aspects or properties of objects are time-variant or time-related. E.g., the moment of a bank transaction and the status of an employee in a company, are time-related and time-variant notions, respectively. A temporal database schema is a database schema

that models objects with time-related or time-variant properties. However, the modelling of temporal aspects has a direct impact on the consistency of the temporal database, because the temporal nature of these aspects imposes extra integrity constraints and suitable ways of interaction with the human user.

For example, let us consider a library database and, concretely, the modelling of the presence of books in the library. Two dates are stored: the loan and the return date. It is clear that a book cannot be loaned again until it is returned. Without further cautions, a library employee could loan the same book several times even if it is not returned. A temporal database model will typically constrain record insertion and prevent similar modelling inconsistencies.

A lot of research concerns temporal database models and their approaches to the modelling of time. The first efforts were towards the representation of historical information related to objects represented by records in a database [7]. Some proposals tried to extend the Entity Relationship Model [28], without impact on any database standards like SQL [43].

An interesting issue in temporal modelling concerns relationships between temporal notions. In this sense, Allen [1] studied temporal relationships between time intervals (and as a special case time points). Among others, the querying of temporal databases has greatly profited from these temporal relationships, because they allow more powerful user-specified temporal query demands, by allowing to express more complex relationships between the temporal notions in the temporal expressions in the query. For example, a query like 'who were the department heads when Thomas worked for the institution' can be evaluated using operators similar to Allen's ones.

Humans handle temporal information using certain temporal notions like time intervals or time points [17], and they often have to deal with imperfections like imprecision, vagueness, uncertainty or inconsistencies possibly contained in the descriptions of these temporal notions. These imperfections in descriptions of temporal notions determine an important issue in temporal modelling. Consider as an example the description of the temporal notion in a sentence like 'The Belfry of Bruges was finished between 01/01/1201 A.D. and 31/12/1300 A.D.'. This sentence contains imperfection because of the uncertainty in the used time-related expression. It is known that the building was finished on a single day, but this day is not precisely known.

To allow information systems to cope with these and similar data imperfections, many approaches adopt fuzzy sets [46] for the representation and management of temporal information [32,33,2,14,12]. The temporal relationships studied by Allen were fuzzified by several authors [35,33,44]. Garrido et al. [21] presented a compact representation for the time and defined different relationships among time intervals by using a combination of regular fuzzy comparisons. Also [21,40] studied uncertainty in temporal expressions concerning time intervals. Other approaches, like [41], use rough sets [37] to represent time intervals.

In addition to temporal modelling, some attention has been paid to temporal reasoning [1]. Although temporal reasoning is not discussed in this paper, it should be noted that, among others, Dubois and Prade et al. [12,16] have dealt with fuzziness and uncertainty in temporal reasoning. Finally, in [5,4], an approach to the linguistic summarization of data with time dimension is presented.

The present work defines and implements a model for properly representing and managing uncertainty in valid-time specification in a relational database. Our work is focused on both the proposal of an appropriate formal framework to suitably manage time in databases and the implementation of a DML that allows to the user the transparent use of this proposal. None of the previous research offers a database model to accomplish this task.

This way, together with the theoretical model, we also present and explain the main operations of the manipulation language for a temporal database which stores the valid-time periods of the objects affected by imprecision. The rest of the work is organized as follows. Section 2 presents some background concepts about both possibility theory and temporal databases. In section 3 the representation of the valid-time intervals in the database is explained. Section 4 explains the main concepts of the temporal Data Manipulation Language (DML) and

its implementation. Finally, Section 5 presents the conclusions and some guidelines for future research work.

2. Preliminaries

In this section, the framework of set evaluation by ill-known constraints [40] is explained. The section also includes a brief introduction to temporal databases.

2.1. Interval Evaluation by Ill-known Constraints

In our case, we need to know if all points in a crisp interval I reside between the boundaries of an ill-known interval $[X,Y]$, where X e Y are possibilistic variables represented by triangular possibility distributions

$$D,a,b$$

, where D is the modal value and a and b are the left and right spreads respectively. To compute that, we introduce the concept of *ill-known constraint*[40].

Definition 1. Given a universe U, an ill-known constraint C on a set $A \subseteq U$ is specified by means of a binary relation $R \subseteq \widetilde{\mathscr{P}}(U)^2$ and a fixed ill-known value denoted by its possibilistic variable V over U, i.e.:

$$C \triangleq (R,V) \tag{1}$$

Set A satisfies the constraint if and only if:

$$\forall a \in A : (a,V) \in R \tag{2}$$

An example of an ill-known constraint is $C_{ex} = (<,X)$. Some set A satisfies C_{ex} if $\forall a \in A : a < X$, given the possibilistic variable X.

The satisfaction of a constraint $C \triangleq (R,V)$ by a set A is still a Boolean matter, but due to the uncertainty about the ill-known value V, it can be uncertain whether C is satisfied by A or not [40]. In fact, this satisfaction now behaves as a proposition. Based on the possibility distribution π_V of V, the possibility and necessity that A satisfies C can be computed. This proposition can thus be seen as a possibilistic variable on $\mathbb{B}$. The required possibility and necessity are:

$$\text{Pos}(A \text{ satisfies } C) = \tag{3}$$
$$\min_{a \in A} \left(\sup_{(a,w) \in R} \pi_V(w) \right)$$
$$\text{Nec}(A \text{ satisfies } C) = \tag{4}$$
$$\min_{a \in A} \left(\inf_{(a,w) \notin R} 1 - \pi_V(w) \right)$$

So far, we have shown how it can be verified whether a set satisfies or not an ill-known constraint. The interval evaluation problem is explained in a more general context in [40].

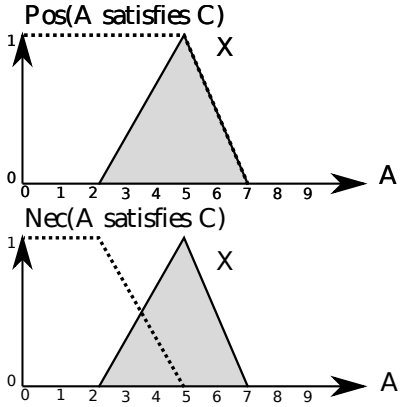

Fig. 1. Example of the evaluation of the ill-known constraint $C \triangleq (\leqslant, X), X = [5,3,2]$.

Example 1. Consider $X = [5,3,2]$ and let $C = (\leqslant, X)$ the ill-known constraint. Then, the evaluation of the possibility and the necessity are obtained from (3) and (4) respectively. (See Figure 1).

$$\text{Pos}(A \text{ satisfies } C) = \tag{5}$$
$$\min_{a \in A} \left(\sup_{a \leqslant w} \pi_X(w) \right)$$
$$\text{Nec}(A \text{ satisfies } C) = \tag{6}$$
$$\min_{a \in A} \left(\inf_{a > w} 1 - \pi_X(w) \right)$$

It is observed that Boolean combinations of constraints are required. For example, the problem of interval evaluation (explained earlier) requires that all the elements of an interval $[a,b]$ are larger than a value X and, at the same time, smaller than a value Y, which implies that a conjunctive Boolean combination of both constraints must be satisfied. To allow Boolean combinations of constraints, the following definitions are introduced.

Definition 2. Consider a universe U, an n-ary vector $\mathbf{C}$ of constraints and a Boolean function $\mathscr{B} : \mathbb{B}^n \to \mathbb{B}$. An evaluation function is defined by:

$$\lambda : \mathscr{P}(U) \to \mathbb{B} : \lambda(A) \mapsto \mathscr{B}\Big(C_1(A), ..., C_n(A)\Big). \tag{7}$$

Definition 2 presents a function that evaluates a Boolean combination of some basic constraints. Informally, it states that a set A passes the evaluation made by λ if the Boolean combination of some propositions equals T. This crisp definition can be generalized to the case of ill-known constraints.

Definition 3. Consider a universe U, an n-ary vector $\mathbf{C}$ of ill-known constraints and a Boolean function $\mathscr{B} : \mathbb{B}^n \to \mathbb{B}$. The uncertainty about the evaluation of a set A by an evaluation function λ is then given by:

$$\forall A \in \mathscr{P}(U) : \pi_{\lambda(A)} = \widetilde{\mathscr{B}}\Big(\pi_{C_1(A)}, ..., \pi_{C_n(A)}\Big) \tag{8}$$

Hereby, $\widetilde{\mathscr{B}}$ is the possibilistic extension of $\mathscr{B}$. It is well known that any Boolean function $\mathscr{B}$ can be cast to a canonical form [29], requiring only logical conjunction $\wedge$, logical disjunction $\vee$ and logical negation. Therefore, only those connectives will be treated within the scope of this paper. By applying the possibilistic extensions of $\wedge$, $\vee$ and $\neg$, concrete equations are obtained for the calculations of uncertainty about the evaluation of a set by means of an evaluation function λ. In the case of conjunction (i.e., $\mathscr{B} = \wedge$), the inference of uncertainty about the evaluation of a set reduces to:

$$\forall A \in \mathscr{P}(U) : \mathrm{Pos}(\lambda(A)) = \min_{i=1...n} \mathrm{Pos}\,(C_i(A)) \tag{9}$$

$$\forall A \in \mathscr{P}(U) : \mathrm{Nec}(\lambda(A)) = \min_{i=1...n} \mathrm{Nec}\,(C_i(A)). \tag{10}$$

In the case of disjunction (i.e. $\mathscr{B} = \vee$), the inference of uncertainty about the evaluation of a set reduces to:

$$\forall A \in \mathscr{P}(U) : \mathrm{Pos}(\lambda(A)) = \max_{i=1...n} \mathrm{Pos}\,(C_i(A)) \tag{11}$$

$$\forall A \in \mathscr{P}(U) : \mathrm{Nec}(\lambda(A)) = \max_{i=1...n} \mathrm{Nec}\,(C_i(A)). \tag{12}$$

Note that by using the functions min and max here, there is an implicit assumption that the possibilistic variables π_{C_i} are mutual min-dependent in the sense of De Cooman (i.e. non-interactive). For an extensive reading on (in)dependency of possibilistic variables, the reader is referred to [22],[23],[24]. In case of $\neg$, we get:

$$\forall A \in \mathscr{P}(U) : \mathrm{Pos}(\neg\lambda(A)) = 1 - \mathrm{Nec}(\lambda(A)) \tag{13}$$

$$\forall A \in \mathscr{P}(U) : \mathrm{Nec}(\neg\lambda(A)) = 1 - \mathrm{Pos}(\lambda(A)). \tag{14}$$

Example 2. Consider that we want to check if the crisp interval $I = [j,k]$ is included in $[X,Y]$. In this situation, two ill-known constraints are constructed.

$$C_1 \triangleq (\geqslant, X) \tag{15}$$

$$C_2 \triangleq (\leqslant, Y) \tag{16}$$

To calculate the possibility and necessity concerning a conjunction of constraints, the min operator can be used. The possibility and necessity of I being included in $[X,Y]$ are now:

$$\mathrm{Pos}(I \text{ satisfies } C_1 \textit{ and } C_2) = \min_{a\in I}\left(\sup_{a\geqslant w} \pi_X(w), \sup_{a\leqslant v} \pi_Y(v)\right) \tag{17}$$

$$\mathrm{Nec}(I \text{ satisfies } C_1 \textit{ and } C_2) = \min_{a\in I}\left(\inf_{a<w} 1-\pi_X(w), \inf_{a>v} 1-\pi_Y(v)\right). \tag{18}$$

There is a special boolean combination of constraints that is of particular interest; let us see it.

Definition 4. *Conjunctive combination of ill-known constraints*. Consider a universe U. Let R_1, R_2 be two binary relations in $\widetilde{\mathscr{P}}(U)$. Let X_1, X_2 be two fixed ill-known values in U. Let $C_1 = (R_1, X_1)$ and $C_2 = (R_2, X_2)$ be two ill-known constraints. A conjunctive combination of both constraints is given by:

$$CC \triangleq \{C_1 \wedge C_2\} \tag{19}$$

In a more general way, it is possible to define the conjunctive combination of an n-ary vector of constraints:

$$CC \triangleq \{C_1 \wedge \ldots \wedge C_n\} \tag{20}$$

Theorem 1. *Consider the conjunctive combination CC of any n-ary vector* $\left\{C_{1_{Z_1}}, \ldots, C_{n_{Z_n}}\right\}$ *of constraints over the ill-known variables* $Z_1, \ldots, Z_n$. *Then, if* $\pi_{C_1(Z_1)}, \ldots, \pi_{C_n(Z_n)}$ *are convex, then* π_{CC} *is also convex.*

Proof. Let $CC \triangleq \left\{C_{1_{Z_1}} \wedge \ldots \wedge C_{n_{Z_n}}\right\}$. Then:

$$\begin{aligned} \pi_{CC}(\lambda x_1 + (1-\lambda)x_2) = \\ \min\big(\pi_{C_1(Z_1)}(\lambda x_1 + (1-\lambda)x_2), \ldots, \\ \pi_{C_n(Z_n)}(\lambda x_1 + (1-\lambda)x_2)\big) \end{aligned} \tag{21}$$

Since $\pi_{C_1(Z_1)}, \ldots, \pi_{C_n(Z_n)}$ are convex:

$$\begin{aligned} \pi_{C_1(Z_1)}(\lambda x_1 + (1-\lambda)x_2) \geqq \\ \min\big(\pi_{C_1(Z_1)}(x_1), \pi_{C_1(Z_1)}(x_2)\big) \\ \vdots \\ \pi_{C_n(Z_n)}(\lambda x_1 + (1-\lambda)x_2) \geqq \\ \min\big(\pi_{C_n(Z_n)}(x_1), \pi_{C_n(Z_n)}(x_2)\big) \end{aligned} \tag{22}$$

Then, by using equation (21):

$$\begin{aligned} \pi_{CC}(\lambda x_1 + (1-\lambda)x_2) \geqq \\ \min\big(\min\big(\pi_{C_1(Z_1)}(x_1), \pi_{C_1(Z_1)}(x_2)\big), \\ \ldots, \min\big(\pi_{C_n(Z_n)}(x_1), \pi_{C_n(Z_n)}(x_2)\big)\big) \end{aligned} \tag{23}$$

Which is equivalent to the following:

$$\begin{aligned} \pi_{CC}(\lambda x_1 + (1-\lambda)x_2) \geqq \\ \min\big(\min\big(\pi_{C_1(Z_1)}(x_1), \ldots, \pi_{C_n(Z_n)}(x_1)\big), \\ \ldots, \min\big(\pi_{C_1(Z_1)}(x_2), \ldots, \pi_{C_n(Z_n)}(x_2)\big)\big) \end{aligned} \tag{24}$$

Finally we obtain:

$$\begin{aligned} \pi_{CC}(\lambda x_1 + (1-\lambda)x_2) \geqq \\ \min\big(\pi_{CC}(x_1), \pi_{CC}(x_2)\big) \end{aligned} \tag{25}$$

□

Sometimes, an ill-known value might be specified by a convex combination of ill-known constraints. This allows to define ill-known values by means of relationships with respect to other ill-known points.

Definition 5. *Ill-known value defined by conjunctive combination of constraints.* Consider a universe U, $CC \triangleq \left\{C_{1_{Z_1}} \wedge \ldots \wedge C_{n_{Z_n}}\right\}$ a conjunctive combination of ill-known constraints over the variables $Z_1, \ldots, Z_n$. The uncertainty about the evaluation of an ill-known value X is given by:

$$X \in \widetilde{\mathscr{P}}(U) : \pi_X = \pi_{CC} \tag{26}$$

The definition of the ill-known value X with respect to the conjunctive combination of ill-known constraints, is written as:

$$X \triangleq CC \tag{27}$$

Note that π_X is convex since π_{CC} is convex as demonstrated in Theorem 1.

Example 3. As an example, consider a historical database containing data about diplomatic medieval documents. The starting and ending dates when a diplomatic document was valid, are not precisely known. Consider now that the time granularity are years. Then $X = [1112, 2, 2]$ is the starting year for the validity of a document. A new diplomatic document was valid in the year $Y = [1118, 2, 1]$. Then, it is possible to obtain Z (the period of time between the starting of both documents) by using a conjunctive combination of constraints.

$$\begin{aligned} CC = \{C_1(>, X) \wedge C_2(\leqslant, Y)\} \\ Z = CC \end{aligned}$$

Z is a fuzzy interval defined by a trapezoidal shape given by [1112, 1114, 1118, 1119]. Figure 2 illustrates the relations among the variables X, Y and Z.

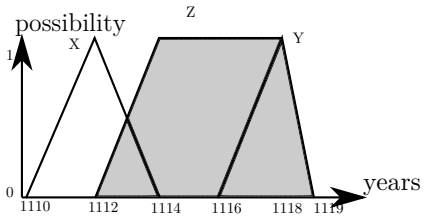

Fig. 2. Ill-known values X and Y. The grey area represents the ill-known value Z defined by the convex combination of the two ill-known constraints C_1 and C_2.

We have seen the main theoretical concepts about ill-known values. Now, we are going to explain the main concepts about the treatment of time and the imperfection related to the time in databases. These two preliminary analysis will be the pillars of our proposal in section 3.

2.2. Time in Databases

The concept of time has been studied in the database framework for a long time. A true standard for adding temporal aspects to relational databases does not exist, but there is a consensus in the literature [17] on what is called a *temporal database*: a temporal database is a database dealing with some aspects of time in its schema. In a temporal DBMS, a **chronon** is the shortest duration of time supported by the system. In temporal databases, some temporal attributes can be managed without treating the attribute differently from non-temporal attributes. The time described by such an attribute is called **user defined time** (*UDT*). In addition to UDT, the following types of time can be discerned in a temporal database, all of which are handled exceptionally by the DBMS:

- **Transaction time** (*TT*) [42,25] denotes the time when the fact (object) is stored in the database. It is usually append-only: as the past cannot be changed, TT cannot be changed either. Furthermore, at the moment of insertion, a TT can be neither in the past nor in the future.
- **Valid time** (*VT*) [26,43,30] denotes the time when the fact (object) is true in the modelled reality. A fuzzy extension has been proposed by [21].
- **Decision time** (*DT*) [34,6,19,36] denotes the time when an event was decided to happen.

For example, consider a database containing employee contract descriptions. The time when the employee's contract is valid, represented by an interval, is the VT. The time when the employee's contract is stored in the database is the TT. The time when the decision for hiring this employee was made is a DT.

When working with these time concepts, the Data Manipulation Language (*DML*, which is part of the standard database querying language SQL) is extended to deal with possible temporal inconsistencies within the data and to handle more complex (temporal) queries. Depending on the time managed, a database is classified as either a **Valid Time Database** (*VTDB*), a **Transaction Time Database** (*TTDB*), a **bi-temporal database** (both valid and transaction time are managed) or a **tri-temporal** or *multitemporal* database (valid time, transaction time and decision time are managed).

2.2.1. Imperfection and time

The representation of imprecision and its semantics when dealing with time has been studied for a long time. Several proposals for representing and handling imprecise time indications can be found in [9,10]. Also, the changes between several granularities can be seen as a source of imprecision [11].

In this paper we will consider two kinds of imperfection:

- **Imperfection in the database**; the knowledge about the temporal data contains some imperfection. E.g., a database record shows that *'The car was in the garage around April.'*
- **Imprecision in the query specification**; it denotes the imprecision in the specification of temporal criteria by the user, when querying. E.g., *'The user wants a car which was in the garage around April.'*

2.2.2. Representation

Several proposals for managing uncertain time in a database exist. Some proposals work with rough sets [41], and some others rely on possibility distributions for representing uncertainty in time [18,21,20]. To compare temporal possibility distributions, the extensions of the classical Allen's operators [1] are defined in [35,33,13,44].

In order to deal with uncertainty in time intervals, several proposals have been made. Here, two approaches are described: the first one, based on *Fuzzy Validity Periods* [21] and the second one, based on *Possibilistic Valid-time Periods* [38].

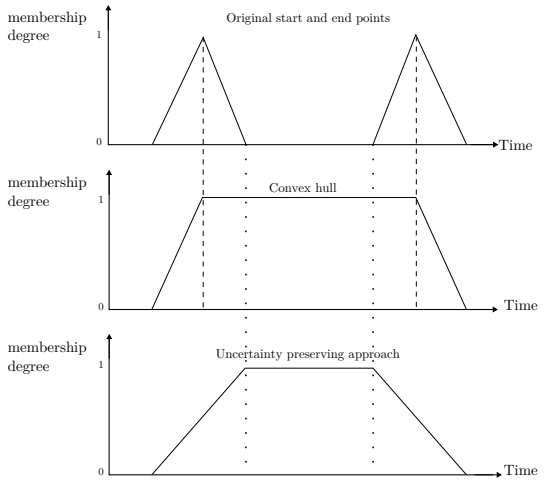

Fig. 3. Transformation to obtain the FVP. The top graph shows the two triangular possibility distributions. The middle graph shows the convex hull validity period, the bottom one shows the result of the second transformation, which maintains the imprecision.

Definition 6. A **Fuzzy Validity Period**[21] (*FVP*) is defined as a fuzzy time interval specifying when the data regarding an object are valid. A fuzzy time interval is then the fuzzification of a crisp time interval.

Several options to transform possibility distributions corresponding to the fuzzy starting point and the fuzzy end point into a consistent FVP exist [21], e.g (Fig. 3):

- The **convex hull** approach is the most intuitive approach. The resulting FVP is the convex hull of the union of both possibility distributions.
- The **uncertainty preserving** approach. The amount of uncertainty is maintained at the edges of the possibility distribution representing the FVP [21].

The main feature for the FVP is the optimization for the storage. The compact representation is the result of a conjunctive semantic. The object is valid within all the time points inside the starting and ending points.

Definition 7. A **Possibilistic Valid-time Period** (*PVP*) is an ill-known interval of time specifying when the data regarding an object are valid.

Note that the PVP represents only one crisp time interval, but for some reason, it is (partially) unknown.

The main advantage for the PVP is that it preserves all the information for both starting and ending points[40,39]. Table 1 is a comparative between PVP and FVP. The following list defines the items in the comparative:

1. Domain: The domain of the possibility distribution modelled by the approach.
2. Implementation of relationships: How to implement a relationship.
3. Allen's relations: Are the Allen's relations defined?

4. Storage: The way the data are stored in the database.

5. Possibility measures: Does the framework provide a possibility measure for any relation between the temporal elements?

6. Necessity measures: Does the framework provide a necessity measure for any relation between the temporal elements?

Table 1. Comparative PVP vs FVP

Item	PVP	FVP
(1)	$\mathscr{P}(\mathbb{R})$	$\mathbb{R}$
(2)	Ill-known constraints.	Ad-hoc operators.
(3)	✓	-
(4)	Two distributions one for each endpoint.	Only one distribution.
(5)	✓	✓
(6)	✓	-

In the rest of the paper we will work only with PVP to represent valid-time intervals.

2.3. *Understanding Valid-time Databases*

This subsection is devoted to describe the behaviour of a crisp valid-time database. For the sake of simplicity, only the three main operations in the Data Manipulation Language (CReate Update, Delete) are shown. Usually the DML operations in a temporal database are re-defined (a typical update sentence in SQL could be expressed by means of a couple of insert and update sentences). Therefore, for the sake of clarity, these high level primitives in the DML for a valid-time database are usually noted as Insert, Modify and Delete. In the following subsections each primitive is defined and explained. Finally, an illustrative example is given. For a more complete information on the behaviour of a bitemporal database, please refer to [26].

Definition 8. *Valid-time relation.* Consider the following definitions and notations:

- A set of non-temporal attributes.

$$A = \{A_1, A_2, \ldots, A_n\} \tag{28}$$

  The domain for each attribute $A_1, \ldots, A_n$ is $D_1, \ldots, D_n$ respectively.
- The original primary key A_K is a subset of the attributes in A.

$$A_K \subseteq A \tag{29}$$

- Two attributes, S and E for the starting and ending points respectively. I defines the valid time interval for the data.

$$I = \{S, E\} \tag{30}$$

  $\mathscr{T}$ is the time domain.
- Then R, the schema for the valid-time relation is:

$$R = A \cup I \tag{31}$$

- The primary key for the valid-time relation R is:

$$PK = A_K \cup I \tag{32}$$

- We will note by r any valid instance of R.

$$r \subseteq D_1 \; x \; \ldots \; x \; D_n x \; T x \; T \tag{33}$$

- $V(t)$ is the set of all the versions for a given tuple t. Formally,

$$V(t) = \{t_i \in r, t_i[A_K] = t[A_K]\} \tag{34}$$

  Obviously, t itself is included in the set.

We will illustrate the definitions with an example.

Example 4. Consider the set of attributes $A = \{A_1, A_2, A_3\}$. The primary key for these attributes is given by $A_K = \{A_1, A_2\}$. Let $I = \{S, E\}$ be the set of temporal attributes that define the validity period of the data. R is the valid-time relation and r is an instance of the relation. The instance r is given by the following elements. $r = \{(a_{11}, a_{12}, a_{13}, s_1, e_1), (a_{21}, a_{22}, a_{23}, s_2, e_2)\ , \ (a_{11}, a_{12}, a_{31}, s_3, e_3)\}$. The instance r is illustrated in Table 2. Consider the tuple

$t_1 = (a_{11}, a_{12}, a_{13}, s_1, e_1)$. Then,

$$
\begin{aligned}
t_1[S] &= s_1 \\
t_1[E] &= e_1 \\
t_1[S,E] &= (s_1, e_1) \\
t_1[A_K] &= (a_{11}, a_{12}) \\
t_1[PK] &= (a_{11}, a_{12}, s_1, e_1) \\
V(t_1) &= \{t_1, t_3\}
\end{aligned}
$$

Table 2. Sample database containing the instance r of the valid-time relation R.

	$\mathbf{A}_1$	$\mathbf{A}_2$	A_3	S	E
t_1	a_{11}	a_{12}	a_{13}	s_1	e_1
t_2	a_{21}	a_{22}	a_{23}	s_2	e_2
t_3	a_{11}	a_{12}	a_{31}	s_3	e_3

In order to simplify the algorithms for the manipulation of data, some auxiliary functions and constants are defined:

Definition 9. *From the beginning (FB).* Consider the elements in definition 8 and a tuple t. We will say $t[S] = FB$ when $t[S] = -\infty$.

Definition 10. *Until changed (UC).* Consider the elements in definition 8 and a tuple t. We will say $t[E] = UC$ when $t[E] = +\infty$.

Definition 11. *Current.* Consider the elements in definition 8. We will say that the tuple t is current in the instance r of the relation R when $t[E] = UC$.

For example, let us consider the last row in Table 3. The value for the time interval is $I = (4/4/2012, UC)$. The meaning is that the document with ID = 3 was valid the 4/4/2012 and it is still valid. The document with ID = 3 is current in the relation.

Example 5. Consider a historical database containing diplomatic documents. The starting and the ending dates say when the diplomatic document is valid. It is possible that a diplomatic document is valid for a period of time and several years later it becomes valid again. The following elements are stored: an identifier of the document (ID), the entity that issues the document and the dates when the document is valid. Table 3 illustrates the first version of the database, after three insertions. In this example, $A =$ (ID, Entity) and $I =$ (Start, End).

Table 3. Sample historical database

ID	Entity	Start	End
3	E.U.	15/3/2012	30/3/2012
4	N.A.T.O.	25/3/2012	4/4/2012
5	C.E.I.	18/3/2012	2/4/2012
3	E.U.	4/4/2012	UC

Definition 12. *Current* (r, a_k). Consider the elements in definition 8. The function Current (r, a_k) returns the crisp time interval of a tuple t with primary key a_k as follows:

$$
\text{Current}(r, a_k) = \tag{35}
$$

$$
\begin{cases}
t[S,E] & \text{if } \exists t \in r : t[E] = UC \text{ and} \\
 & t[A_K] = a_k \\
\emptyset & \text{otherwise}
\end{cases}
$$

For example, the function Current(r, 3) returns the time interval $(4/4/2012, UC)$. Conversely, Current(r, (4,'N.A.T.O.')) returns the empty set.

The Allen's relations between two time intervals are shown in Figure 4. The complete implementation of the relations using only the starting and the

ending points is defined in [33].

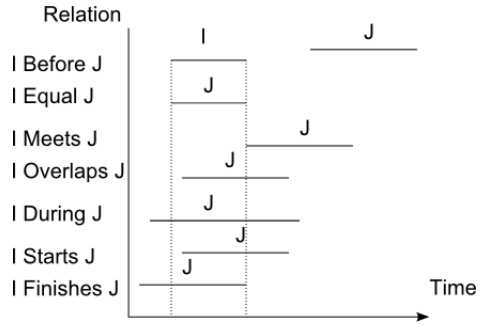

Fig. 4. Allen's relations for two crisp time intervals I and J. Note that the crisp time interval I is fixed and the different positions of the interval J illustrate the Allen's relations.

We will use two of the Allen's relations: Overlaps and During which will be defined as follows.

Definition 13. *Overlaps:* Given two time intervals defined by the couples of values $i_1 = (s_1, e_1)$ and $i_2 = (s_2, e_2)$, it is said that i_1 overlaps i_2 if:

$$i_1 \text{ overlaps } i_2 = (((s_1 < s_2) \wedge (e_1 < e_2)) \vee (((s_1 > s_2) \wedge (e_1 > e_2)) \quad (36)$$

Definition 14. *During:* Given two time intervals defined by the couples of values $i_1 = (s_1, e_1)$ and $i_2 = (s_2, e_2)$, it is said that i_1 during i_2 if:

$$i_1 \text{ during } i_2 = (s_1 > s_2) \wedge (e_1 < e_2) \quad (37)$$

In the rest of the paper, and without losing generality, we will consider that the time granularity are days. Also, the dates will be given in the format *dd/mm/yyyy*.

Definition 15. *CloseR*(i_1, i_2)*:* Consider two crisp intervals defined by the couples of values $i_1 = (s_1, e_1)$ and $i_2 = (s_2, e_2)$. The CloseR(i_1, i_2) function allows to close the right-open interval i_1 with respect to the first value s_2 in i_2:

$$\text{CloseR}(i_1, i_2) = \quad (38)$$

$$\begin{cases} (s_1, s_2 - 1) & \text{if } e_1 = UC \text{ and } i_2 \text{ During } i_1 \\ i_1 & \text{otherwise} \end{cases}$$

For example, consider two time intervals, $i_1 = (4/4/2012, UC)$ and $i_2 = (24/4/2012, UC)$. The result of applying the CloseR(i_1, i_2) is $i_1 = (4/4/2012, 23/4/2012)$.

Now it is possible to close the current version of an entity by using (38) and (35). This functionality is required to add or update new information about an existing entity in the relation.

Definition 16. *Close-current*(r,t). Consider the elements in definition 8. The function Close-current(r,t) closes any current version t_k of the entity given by t and adds the new version t. For the implementation t_{CUR} and t_{UP} variables are defined:

$$\begin{aligned} t_{CUR}[A_k] &= t[A_K] \quad (39) \\ t_{CUR}[S,E] &= \text{Current } (r, t[A_K]) \\ t_{UP}[A_k] &= t[A_k] \\ t_{UP}[S,E] &= \text{CloseR } (t_{CUR}[S,E], t[S,E]) \end{aligned}$$

Then, t_{CUR} is the current version of the entity given by the tuple t, and t_{UP} is the updated version of the tuple t_{CUR}. In this updated version, the time interval given by i_{UP} is closed with respect to the tuple t.

$$\text{Close-current}(r,t) = \quad (40)$$

$$\begin{cases} r - t_{CUR} \cup \{t_{UP}\} \cup \{t\} & \text{if Current } (r, t[A_K]) \neq \emptyset \\ r, \text{ otherwise} \end{cases}$$

For example, consider the document with ID = 3 in Table 3. The current version of the document started on 4/4/2012. The document was not valid anymore but, for some reason, the date was not registered. The document was valid again on 24/4/2012. Then, the Close-current function closes the current version of the document and adds a new version. First, the function CloseR is applied with $i_1 = (4/4/2012, UC)$ and $i_2 = (24/4/2012, UC)$. Hence, the value for the time interval i_1 is $(4/4/2012, 23/4/2012)$. Then, the modifications on the value for the time interval i_1 are stored. Finally a new row with the current values of the document and the time interval i_2 is stored. The result of this operation is illustrated on Table 4.

2.3.1. Modify

This operation adds new information about an existing entity (given by the tuple t) in the instance r of the relation R. The modify operation does not remove any previous value of the entity. It closes the current version and adds a new version.

Definition 17. *Modify(r, t).* Consider the elements in definition 8. The algorithm for the modify operation is defined as follows.

$$\begin{aligned} &\textbf{modify}\,(r,t) = \\ &\text{Close-current}\ (r,t) \end{aligned} \tag{41}$$

2.3.2. Insert

The user wants to store an entity (given by the tuple t) which is valid in the instance r of the relation R during the time interval specified by $i = (s,e)$. There are two main cases when performing a create operation:

1. The entity was never in the relation: The entity is added with the valid-time indicated by the crisp interval i.
2. The entity is in the relation. Depending on the value of the time interval, there are three possibilities:
   (a) Insert t in the instance r of the relation R. If the time interval i does not overlap any other valid-time interval in the instance r relation R for the entity. For example, consider that the document with ID = 3 began again to be valid on 4/4/2012 and it is still currently valid. This is illustrated in Table 3.
   (b) Modify and close the current version of t and insert the new version. For example, consider now that the document with ID = 3 was valid on 24/4/2012. Here the problem is that the document with ID = 3 was valid on 4/4/2012 but, for some reason, the ending date was not stored. If the document is again valid, then it is necessary to set the ending date and add a new row with the new starting date. This is illustrated in Table 4.
   (c) Reject the insertion, if the time interval i does overlap any existing valid-time interval for the entity t in the instance r of the relation R. For example, consider that the document manager wants to introduce a past valid-time for the document with ID = 3. The validity period for the document is [6/4/2012, 25/4/2012]. As the dates do overlap, it is not possible that the document was valid at that time interval. Therefore, the insertion is rejected.

Table 4. Sample historical database

ID	Entity	Start	End
3	E.U.	15/3/2012	30/3/2012
4	N.A.T.O.	25/3/2012	4/4/2012
5	C.E.I.	18/3/2012	2/4/2012
3	E.U.	4/4/2012	23/4/2012
3	E.U.	24/4/2012	UC

Definition 18. *Insert(r, t).* Consider the elements in definition 8. Then, the algorithm for the implementation of the insert operation is defined as follows.

$$\textbf{insert}\,(r,t) = \tag{42}$$

$$\begin{cases} r & \text{if } \exists t_k \in V(t), (t[S,E] \text{ overlaps } t_k[S,E]) \\ r \cup \{t\} & \text{if } V(t) = \emptyset \text{ or} \\ & \forall t_k \in V(t), \neg\,(t[S,E] \text{ overlaps } t_k[S,E]) \\ \text{modify(r, t)} & \text{otherwise} \end{cases}$$

2.3.3. Delete

The delete operation logically removes a current entity t which is valid in the instance r of the relation R.

Definition 19. *Delete(r, t).* Consider the elements in definition 8.The algorithm for the delete operation is defined as follows.

$$\textbf{delete}\,(r,t) = r - V(t)$$

The set $V(t)$ is computed as explained in definition 8 and contains all the versions for the tuple t.

For example, consider that the document manager wants to delete the history for the document with ID = 3. The result of this operation is shown in Table 5.

Table 5. Sample historical database

ID	Entity	Start	End
4	N.A.T.O.	25/3/2012	4/4/2012
5	C.E.I.	18/3/2012	2/4/2012

3. Time Representation

This section is devoted to specify the representation of time within the framework of the possibility theory.

First of all, the specification for a single ill-known temporal point will be explained. Then, the formal specification and the related constraints are given for an ill-known valid-time interval.

3.1. Ill-known time point

An ill-known time point X is an atomic time point that, for some reason, is not fully specified.

Note that X has only one possible value but that value is unspecified.

Definition 20. Ill-known time point.
Consider a time domain $\mathscr{T}$; the uncertainty about the values of the ill-known time point X is given by the possibility distribution π_X:

$$\Pi(X) = \pi_X(t) \in [0,1], t \in \mathscr{T} \tag{43}$$

It is also possible to specify an ill-known time point by a convex combination of ill-known constraints, as shown by equation (26).

Definition 21. Domain for an ill-known time point.
Consider $\widetilde{\mathscr{P}}(\mathscr{T})$ the set of all the possibility distributions over $\mathscr{T}$, and the three fuzzy constants:

- $UNKNOWN = \{1/t, \forall t \in \mathscr{T}\}$,
- $UNDEFINED = \{0/t, \forall t \in \mathscr{T}\}$ and
- $NULL = \{1/\ UNKNOWN, 1/\ UNDEFINED\ \}$.

The domain for an ill-known time point X is given by:

$$\begin{aligned}\mathscr{D}(X) = \{&\widetilde{\mathscr{P}}(\mathscr{T}) \cup UNKNOWN \\ &\cup UNDEFINED \\ &\cup NULL\}\end{aligned} \tag{44}$$

3.1.1. Datatypes

The data type for the representation of an ill-known time point allows the representation of the values shown in Table 6.

Table 6. Values for the time point data type.

DIFFERENT VALUES FOR A TIME POINT		
Subtype	Value	Representation
1	A single time point	$1/x, x \in \mathscr{T}$
2	A possibility distribution in the numeric domain	A fuzzy number or a fuzzy interval.
3	An unknown value	**UNKNOWN**= $\{1/t, \forall t \in \mathscr{T}\}$
4	An undefined value	**UNDEFINED**= $\{0/t, \forall t \in \mathscr{T}\}$
5	A null value	**NULL** $= \{1/\text{Unknown}, 1/\text{Undefined}\ \}$

Example 6. Consider a historical database with data from medieval diplomatic documents.

The following fields are stored: the digital identifier *ID* which is the primary key and the estimated time when the document was sent (field *Date*).

Table 7 contains some example data from this database.

Table 7. Sample of the historical database

ID	Date
23454	Unknown
34563	11/12/1204
12211	$[7/2/1204, 30, 30]$
23455	$[10, 10/6/1204, 20/6/1204, 15]$

In that database, for the document with ID=23454, all the dates in the domain are equally possible. Nevertheless, the document 34563 was sent in the crisp (exact) date 11/12/1204. The time for documents 12211 and 23455 are specified with several possibility distributions. The first one is also known as a fuzzy number whereas the second one is also known as a fuzzy interval, as explained in Section 2.

3.2. Ill-known time interval

An ill-known time interval denoted by $[X,Y]$ is a time interval whose boundaries are not precisely known.

Definition 22. *Ill-known time interval* Let $\mathscr{T}$ be the time domain, and X, Y two ill-known values in the time domain. An ill-known time interval is given by $[X,Y]$. The evaluation of the ill-known time interval is given by equations (3),(4). We will note $\mathscr{I}_{PVP}$ the set of all the ill-known time intervals.

3.2.1. Open ill-known time intervals

Quite often, the user may want to specify time intervals with open boundaries in one or both endpoints. Consider an ill-known time interval $[X,Y]$. Then it is possible to distinguish between the following two types of open intervals:

Definition 23. Completely unknown time interval: Both starting and ending points are unknown, therefore the whole interval is unknown.

Definition 24. Semi-open time interval: Only one of the two ill-known boundaries for the time interval $[X,Y]$ is unknown. Example 7 and Figure 5 illustrates a left open time interval.

3.2.2. Representation of semi-open time intervals

As mentioned before, the problem resides in the representation of this kind of interval.

Because of the ill-known constraints C_1, C_2, a function called *Open* should be defined in order to deal with a proper representation of these intervals.

Definition 25. $Open(C)$
Consider an ill-known value T, a binary relationship $B_r \in \{\leqslant, <, >, \geqslant\}$ and the constraint $C \stackrel{\triangle}{=} (B_r, T)$. The function $\text{Open}(C) = (I_p(C), I_n(C))$ provides both possibility and necessity measures for all the points in the open part of a semi-open ill-known time interval.

The possibility and necessity measures are defined by:

$$I_p(C) = \left(\sup_{r \in \mathscr{T}, r\,\text{Rp}\ w} \pi_T(w) \right) \quad (45)$$

$$I_n(C) = \left(\inf_{r \in \mathscr{T}, r\,\text{Rn}\ w} 1 - \pi_T(w) \right) \quad (46)$$

Where the values for the binary relations Rp and Rn are shown in Table 8.

Table 8. Relations for the $\text{Open}(C)$ function. Depending on the relation $B_r \in \{\leqslant, <, >, \geqslant\}$ in the constraint C, the values for Rp and Rn are shown.

	Relations	
Constraint	Rp	Rn
$C \stackrel{\triangle}{=} (<, T)$	$>$	$\leqslant$
$C \stackrel{\triangle}{=} (\leqslant, T)$	$\geqslant$	$<$
$C \stackrel{\triangle}{=} (>, T)$	$<$	$\geqslant$
$C \stackrel{\triangle}{=} (\geqslant, T)$	$\leqslant$	$>$

As explained before, the constants FB and UC are aliases for the function Open with the following parameters:

$$\text{FB} = \text{Open}(C_2) \quad (47)$$

$$\text{UC} = \text{Open}(C_1) \quad (48)$$

Where the constraints C_1 and C_2 are given in equations (15) and (16).

Example 7. Consider an ill-known time interval given by $[\text{FB}, Y]$. Consider also that, in this case,

$Y = [15/10/2012, 3, 4]$. Figure 5 shows the representation for Y. The user wants to obtain the possibility and the necessity measures for the FB part of the interval.

$$\text{FB} = \text{Open}(C_2) \text{ with } C_2 \stackrel{\triangle}{=} (\leqslant, Y)$$
$$I_p(C_2) = \Big(\sup_{r \in \mathscr{T} \geqslant w} \pi_Y(w) \Big)$$
$$I_n(C_2) = \Big(\inf_{r \in \mathscr{T} < w} 1 - \pi_Y(w) \Big)$$

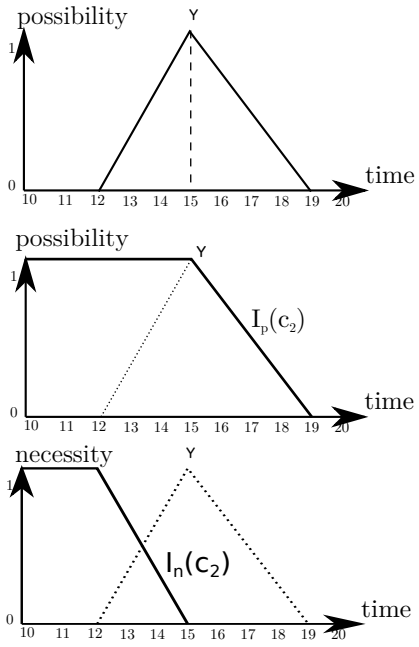

Fig. 5. Possibility distribution for Y, and possibility and necessity measures for the open ill-known point, X

3.2.3. Datatypes

In order to properly represent an ill-known time interval in a database, some datatypes are needed. Because of the ill-known constraints, not all the combinations of datatypes for each ill-known time point (see Table 6) are allowed. Table 9 shows all the possible values that can be used to represent an ill-known time interval denoted by $[X, Y]$.

Table 9. All the possible combination of values for the time interval $[X, Y]$. The subtypes refer to Table 6.

TIME INTERVAL DATA TYPE

Subtype for X	Subtype for Y	Description
1 or 2	1 or 2	An ill-known time interval.
3	3	An unknown time interval.
FB	1 or 2	A left-open time interval.
1 or 2	UC	A right-open time interval.

4. Possibilistic Valid-Time Model for Relational DBs

In this section we will formalize the model for possibilistic valid-time relational databases. The first subsection is devoted to the formalization of the model. Then, a data manipulation language is defined.

4.1. The generalized temporal model

The model is based on GEFRED [31] (Generalized Model of Fuzzy Relational DB) model. This model is extended by adding valid-time support which will be illustrated through the following definitions and examples. The information in the system is defined by the following elements:

Definition 26. *Generalized fuzzy domain.* Let D be the discourse domain, $\widetilde{\mathscr{P}}(D)$ is the set of all possibility distributions defined on D, plus the NULL constant. The generalized fuzzy domain D_G is defined as:

$$D_G \subseteq \widetilde{\mathscr{P}}(D) \cup \text{NULL} \tag{49}$$

The datatypes that can be used to represent D_G are shown in table 10.

Definition 27. Typeof(a). Consider D_G to be a generalized fuzzy domain and the elements in definition 8. Let a be the value for the attribute A. The function typeof(a) returns the datatype associated with the value a and returns a number in $[1, 10]$ as shown in Table 10.

$$\text{Typeof}\,(a) \mapsto [1, 10] \tag{50}$$

Table 10. Data types

No.	Datatype
1	A single scalar.
2	A single number.
3	A set of mutually exclusive possible scalar assignations.
4	A set of mutually exclusive possible numeric assignations.
5	A possibility distribution in a scalar domain.
6	A possibility distribution in a numeric domain.
7	A real number in $[0,1]$ referring to degree of matching.
8	An *UNKNOWN* value.
9	An *UNDEFINED* value.
10	A *NULL* value.

It is possible to define a more specific generalized temporal domain, $\mathscr{T}_G$.

Definition 28. *Generalized fuzzy temporal domain.* Consider $\mathscr{T}$ to be the temporal domain, and let $\widetilde{\mathscr{P}}(\mathscr{T})$ be the set of all *normalized* possibility distributions defined on $\mathscr{T}$. The Generalized Fuzzy Temporal Domain, $\mathscr{T}_G$ is

$$\mathscr{T}_G \subseteq \left\{ \widetilde{\mathscr{P}}(\mathscr{T}) \cup \text{NULL} \right\} \tag{51}$$

Note that $\mathscr{T}_G \subseteq D_G$. The datatypes for this domain have been studied previously in section 3 and are shown in tables 6 and 9.

A generalized fuzzy relation is defined in [31]. Here, we will extend the definition to a generalized fuzzy temporal relation.

Definition 29. *Generalized fuzzy temporal relation.* Consider the elements in definition 8. Some of them will be extended for the fuzzy case.

- An attribute called version identifier, V_{ID}, will be added to the schema. This attribute is a counter for each different version of the entities.
- Then R_{FTG}, the schema for the fuzzy valid-time relation is:

$$R_{FTG} = A \cup V_{ID} \cup I \tag{52}$$

- The primary key for the fuzzy valid-time relation R_{FTG} is:

$$K_{GT} = A_K \cup V_{ID} \tag{53}$$

  A formal definition of the primary key for fuzzy valid-time relations will be given later in Definition 34.
- We will note by r any valid instance of R_{FTG}.

$$r \subseteq D_1 \; x \; \ldots \; x \; D_n x \; \mathbb{N} x \; \mathscr{T}_G \; x \; \mathscr{T}_G \tag{54}$$

- Let K_{GT} be the primary key for the valid-time relation as given in equation (53). Then, k denotes the values for the attributes in the primary key.

$$k = t\,[K_{GT}] \tag{55}$$

Table 11 contains an example instance.

Table 11. Sample database containing the instance r of the fuzzy valid-time relation R_{FTG}.

	$\mathbf{A}_1$	$\mathbf{A}_2$	A_3	V_{ID}	S	E
t_1	a_{11}	a_{12}	a_{13}	001	s_1	e_1
t_2	a_{21}	a_{22}	a_{23}	001	s_2	e_2
t_3	a_{11}	a_{12}	a_{31}	002	s_3	e_3

A generalized fuzzy temporal relation R_{FTG} can be noted also by:

$$R_{FTG} = (\mathscr{H}, \mathscr{B}) \tag{56}$$

Where $\mathscr{H}$ is the Head of the relation and consists on a fixed set of triplets attribute- domain - compatibility with an optional the valid-time attribute:

$$\begin{aligned} \mathscr{H} = \{ & (A_{G1} : D_{G1}\,[, C_{A_{G1}}]), \\ & \ldots, \\ & (A_{Gn} : D_{Gn}\,[, C_{A_{Gn}}]), \\ & \left[(\text{PVP}, D_{\text{PVP}}\,[, C_{A_{\text{PVP}}}]) \right] \} \end{aligned} \tag{57}$$

Note that D_{Gj} $(j = 1, \ldots, n)$ is the domain for the attribute A_{Gj}. $C_{A_{Gj}}$ is the compatibility attribute in the unit interval $[0,1]$.

$\mathscr{B}$ is the body of the relation and it consists on a set of tuples. Each tuple is a set of triplets attribute-value- degree with an optional valid-time attribute:

$$\mathscr{B} = \{\ldots\{\left(A_{G1} : \widetilde{d}_{i1}\,[,c_{i1}]\right), \quad (58)$$
$$\ldots,$$
$$\left(A_{Gn} : \widetilde{d}_{in}\,[,c_{in}]\right),$$
$$\left[\left(\text{PVP}, \widetilde{d}_{\text{PVP}}\,[,C_{A_{\text{PVP}}}]\right)\right]\}\ldots\}$$

The definition in [31] for R_{FTG} shows that classical relations are a particular case of this model.

Example 8. Consider a historical database containing diplomatic documents as explained in example 5. But now, the documents are from the Medieval Ages. Hence, the time is known with imprecision. For simplicity, the ill-known time points are represented as triangular fuzzy numbers. An ill-known time point is given by $[dd/mm/yyyy, a, b]$. The values for a and b are integers. Thus, the date given by $[15/3/1012, 5, 2]$ is a triangular fuzzy number with the left bound the 10/3/1012, the core on the 15/3/1012 and the right bound on 17/3/1012. Table 12 shows the elements Head, $\mathscr{H}$ and Body, $\mathscr{B}$.

$$\mathscr{H} = \{(\text{ID} : D_{\text{ID}}), \quad (59)$$
$$\left(\text{Entity} : D_{\text{Entity}}\right),$$
$$(\text{PVP} : D_{\text{PVP}})\}$$

When the compatibility degree is 1, the component is omitted. The body, $\mathscr{B}$ consists on all the tuples shown in Table 12.

Table 12. Sample historical database

			PVP
$\mathscr{H}$	**ID**	Entity	Start , End
	3	E.U.	[15/3/1012, 5, 2] ,
$\mathscr{B}$			[30/3/1012, 1, 1]
	4	N.A.T.O.	[25/3/1012, 3, 2] ,
			[4/4/1012, 1, 7]
	5	C.E.I.	[18/3/1012, 4, 1] ,
			[2/4/1012, 2, 2]

Definition 30. *Value component.* The value component R^v_{FTG} of a fuzzy temporal relation R_{FTG} is a set with the value components for both the head and the body of the relation:

$$R^v_{FTG} = \{\mathscr{H}^v, \mathscr{B}^v\} \quad (60)$$
Where:
$$\mathscr{H}^v = \{(A_{G1} : D_{G1}), \ldots, (A_{Gn} : D_{Gn})\}$$
$$\mathscr{B}^v = \left\{(A_{G1} : \widetilde{d}_{i1}), \ldots, (A_{Gn} : \widetilde{d}_{in})\right\}$$

For example, in the case of the document with ID = 3:

$$\mathscr{H}^v = \{(\text{ID} : D_{\text{ID}}), \left(\text{Entity} : D_{\text{entity}}\right),$$
$$(\text{PVP} : D_{\text{PVP}})\}$$
$$\mathscr{B}^v = \{\ldots\{(\text{ID} : 3), (\text{Entity} : \text{“E.U.”}),$$
$$t(\text{PVP} : [15/3/1012, 5, 2], [30/3/1012, 1, 1])\}\ldots\}$$

Definition 31. *Compatibility component.* The compatibility component R^c_{FTG} of a fuzzy temporal relation R_{FTG} is a set with the compatibility components for both the head and the body of the relation:

$$R^c_{FTG} = \{\mathscr{H}^c, \mathscr{B}^c\} \quad (61)$$
Where:
$$\mathscr{H}^c = \{[C_{A_{G1}}], \ldots, [C_{A_{Gn}}]\}$$
$$\mathscr{B}^c = \{[c_{i1}], \ldots, [c_{in}]\}$$

For example, in the case of the document with ID = 3:

$$\mathscr{H}^c = \{(C_{\text{ID}}), \left(C_{\text{entity}}\right), (C_{\text{PVP}})\}$$
$$\mathscr{B}^c = \{1, 1, 1, 1\}$$

Definition 32. *Temporal component.* The temporal component R^t_{FTG} of a fuzzy temporal relation R_{FTG} is a set with the temporal components for both the head and the body of the relation:

$$R^t_{FTG} = \left\{\mathscr{H}^t, \mathscr{B}^t\right\} \quad (62)$$
Where:
$$\mathscr{H}^t = \{(\text{PVP}, D_{\text{PVP}}\,[,C_{A_{\text{PVP}}}])\}$$
$$\mathscr{B}^t = \left\{\left[\text{PVP}, \widetilde{d}_{\text{PVP}}\,[,C_{A_{\text{PVP}}}]\right]\right\}$$

For example, in the case of the document with ID = 3:

$$\begin{aligned}\mathscr{H}^t =&\{(\text{PVP}, D_{\text{PVP}})\}\\ \mathscr{B}^t =&\{\ldots(\text{PVP}:[[15/3/1012,5,2],\\ &[30/3/1012,1,1]])\ldots\}\end{aligned}$$

Analogously, it is possible to define both the name value component and the compatibility component for the temporal part.

Definition 33. *Generalized primary key.* Consider D_G to be a fuzzy generalized domain, and let $A_{Gs}:D_{Gs}$ be the attributes and the domain of the attribute for each $s\in S\subseteq(1,\ldots,n)$. A generalized primary key, K_G is a subset of the head:

$$K_G\subseteq\mathscr{H}, K_G=\{(A_{Gs}:D_{Gs})\} \quad (63)$$
$$s\in S\subseteq(1,\ldots,n) \quad (64)$$

Subject to the following constraints:

$$\forall s\in S, \text{Typeof }(D_{Gs})\in\{1,2\} \quad (65)$$
$$\forall i,i'\in\{1,\ldots,m\},\exists s\in S:$$
$$(A_{Gs}:d_{is})\neq(A_{Gs}:d_{i's})$$

For example, consider the database in Table 12. Without any temporal constraint, the primary key K_G is:

$$K_G\subseteq\mathscr{H}, K_G=\{(\text{ID}:D_{\text{ID}})\}$$

In this case, the function *Typeof(ID) = 2* (see Definition 27 and Table 10). The primary key for the table is the attribute *ID*, a unique number. Two different documents have two different values for the *ID* attribute.

In order to add valid-time support, the primary key should be re-defined. E.g., consider the historical database. If the primary key is the *ID* attribute, a document should be valid only during one period of time. To resolve this problem, we extend the given primary key with a version identifier.

Definition 34. *Generalized fuzzy temporal key.* Consider D_G to be a fuzzy generalized domain, and let $A_{Gs}:D_{Gs}$ be the attributes and the domain of the attribute for each $s\in S\subseteq(1,\ldots,n)$. Let V be a new attribute called *version*. A generalized fuzzy temporal key, K_{GT} is a subset of the head.

$$K_{GT}\subseteq\mathscr{H}, K_{GT}=\{(A_{Gs}:D_{Gs})\} \quad (66)$$
$$\cup\{(V_{ID}:\mathbb{N})\}$$
$$s\in S\subseteq(1,\ldots,n)$$

Subject to the following constraints:

$$\forall s\in S, \text{Typeof }(D_{Gs})\in\{1,2\} \quad (67)$$
$$\forall i,i'\in\{1,\ldots,m\},\exists s\in S:$$
$$(A_{Gs}:d_{is})\neq(A_{Gs}:d_{i's})$$

Consider now the database in Table 13. The primary key is now:

$$K_{GT}\subseteq\mathscr{H}, K_{GT}=\{(\text{ID}:D_{\text{ID}}),(V_{\text{ID}}:D_{\text{ID}})\}$$

Table 13. Sample historical database

$\mathscr{H}$	ID	V	Entity	(Start ,End)
	3	001	E.U.	[15/3/1012, 5, 2], [30/3/1012, 1, 1]
$\mathscr{B}$	4	001	N.A.T.O.	[25/3/1012, 3, 2] , [4/4/1012, 1, 7]
	5	001	C.E.I.	[18/3/1012, 4, 1] , [2/4/1012, 2, 2]
	3	002	E.U.	[4/4/1012, 3, 3] , UC

4.2. Data manipulation language

The Generalized Fuzzy Relational Algebra [31] manipulates relations like R_{FTG}. The operations defined are: *Union, Intersection, Difference, Cartesian Product, Projection, Join* and *Selection*. Thus, in this section we will describe and implement the

following operations for temporal databases, as described in 2.3. The operations implemented are: *Insert, Modify* and *Delete*. The semantics of the operations will be the same that those defined for a crisp temporal database, whereas the temporal representation is made by the possibilistic valid-time period and the ill-known constraints (see sections 2 and 3).

It is important to notice that while the result of the evaluation of any comparison between crisp time intervals is boolean, the evaluation of any comparison between *PVPs* is a value in the unit interval.

Since the time intervals are now possibilistic valid-time periods, PVPs, the auxiliary functions defined in equations (38) to (40) are the basis for the following auxiliary functions.

Definition 35. CloseR(i_1,i_2)**.** Consider the elements in definition 29 and ill-known intervals given by $i_1=(s_1,e_1)$ and $i_2=(s_2,e2)$. The CloseR function closes the PVP given by i_1 with a conjunctive combination of ill-known constraints (see section 2.1):

$$\text{CloseR}\,(i_1,i_2)= \quad (68)$$

$$\begin{cases} (s_1,e_z) & \text{if } i_1=(s_1,UC), \\ & e_z \triangleq \{C_1\,(>,s_1)\,,C_2\,(<,s_2)\} \\ i_1 & \text{otherwise} \end{cases}$$

For example, consider $i_1=[[4/4/1012,3,3]\,,UC]$ and $i_2=[[15/4/1012,2,1]\,,UC]$. The result for CloseR$(i_1,i_2)$ is $i_1=[[4/4/1012,3,3]\,,[4,7,13,15]]$. The value that closes i_1 is a trapezoid in the form $[\alpha,\beta,\gamma,\delta]$.

Definition 36. Close-current(r,t)**.** Consider the elements in definition 29. The function Close-current(r,t) closes any current version t_k of the entity given by t if it exists and add the new version t. In order to implement the functionality, the variables in equation (39) are used and the function Current as given by equation (35).

$$\text{Close-current}\,(r,t)= \quad (69)$$

$$\begin{cases} r-t_{CUR}\cup\{t_{UP}\}\cup\{t\} & \text{if Current } (r,t[A_K])\neq\emptyset \\ r, \text{ otherwise} \end{cases}$$

For example, consider the database in Table 13. The function Close-current1$(R_{FTG},(ID=3)\,,$ $[[15/4/1012,2,1]\,,UC])$ closes the current version of for the patient with ID=3 and creates a new version with the specified time interval.

4.2.1. Modify

This operation adds new information about an existing entity (given by the tuple t) in the instance r of the fuzzy temporal relation R_{FTG}. The modify operation does not remove any previous value of the entity. Note that the modify operation is only applicable when the entity is current in the relation; $t\in r,=(s,\text{UC})$.

Definition 37. ***modify*** (r,t). Consider the elements in definition 29. The algorithm for the modify operation is defined as follows.

$$\textbf{modify}\,(r,t)= \quad (70)$$
$$\text{Close-current } (r,t)$$

4.2.2. Insert

The user wants to store an entity (given by the tuple t) which is valid in the instance r of the fuzzy temporal relation R_{FTG} during the time interval specified by the PVP, $i=(s,e)$. There are the following cases when performing an insert operation:

1. The entity was never in the relation: The entity is added with the valid-time indicated by the PVP, i. For example, consider the database given by Table 13. The following sentences correspond with the insertion of the first validity period for each document.

```
Insert(3,'E.U.',
[[15/3/1012, 5, 2],
        [30/3/1012, 1, 1]]);
Insert(4,'N.A.T.O.',
[[25/3/1012, 3, 2],
        [4/4/1012, 1, 7]]);
Insert(5,'C.E.I.',
[[18/3/1012, 4, 1],
        [2/4/1012, 2, 2]]);
```

2. The entity is in the relation. Depending on the value of the time interval i, there are three possibilities:

   (a) Insert t in the instance r of the relation R_{FTG}. If the time interval i does not overlap any other valid-time interval in the relation R_{FTG}. Note that here, the result of the overlaps operator is in the unit interval. For example, the document with ID=3 is still valid. The insert sentence is the following.

   ```
   Insert(3,'E.U.',
   [[4/4/1012, 3, 3], UC]);
   ```

   (b) Modify and close the current version of the entity and insert a new version. For example, consider now that the document with ID=3 was valid around the 24/4/1012 (this is model by a triangular fuzzy number: $[24/4/1012, 1, 1]$). Here the problem is that the document with ID=3 was valid around 4/4/1012, but, for some reason, the ending date was not stored. If the document is again valid, then it is necessary to set the ending date and add a new row with the new starting date.

   ```
   Insert(3,'E.U.',
   [[24/4/1012, 1, 1], UC]]);
   ```

   (c) Reject the insertion, if the time interval i does overlap with a degree of 1 any existing valid-time interval for the entity in the relation. For example, consider now that the document manager wants to introduce a past validity period for the document with ID=3. The validity starting date for the document was around 6/4/12/1012 and the ending date was around 25/4/1012. As this interval does overlaps other time intervals with a degree of 1, it is not possible that the document was valid during two different periods of time. Therefore, the insertion is rejected. The insert sentence is:

   ```
   Insert(3,'E.U.',
   [[6/4/12/1012, 1, 1],
            [25/4/1012, 1, 1]]);
   ```

Definition 38. ***insert*** (r,t). Consider the elements in definition 29. The algorithm for the implementation of the insert operation is defined as follows.

$$\textbf{insert}(r,t) = \tag{71}$$

$$\begin{cases} r & \text{if } \exists t_k \in V(t), \\ & (t[S,E] \text{ overlaps } t_k[S,E] = 1) \\ r \cup \{t\} & \text{if } V(t) = \emptyset \text{ or} \\ & \forall t_k \in V(t), \\ & (t[S,E] \text{ overlaps } t_k[S,E]) < 1 \\ \text{modify(r, t)} & \text{otherwise} \end{cases}$$

4.2.3. Delete

The delete operation logically removes an entity which is valid in the instance r of the relation R_{FTG}.

Definition 39. ***delete*** (r,t). Consider the elements in definition 29. The algorithm for the delete operation is defined as follows.

$$\textbf{delete}(r,t) = r - V(t)$$

For example, consider that the document manager wants to delete the history for the document with ID = 3. The following sentence deletes all the rows for the documents with ID = 3.

```
Delete(3,'E.U.');
```

5. Conclusions

In this work, we presented a complete valid-time model to represent and handle ill-known temporal intervals. The paper includes the formal definition of possibilistic valid-time period in order to manage the time and the formal definition of ill-known constraints to define operators and integrity. This is the first formal model in the literature for possibilistic valid-time in relational databases. The semantics and the implementation of the DML operations are described within this work. As future work , the definition of the Data Definition Language as well as the querying will be considered.

Acknowledgments

Work supported in part by the grant BES-2009-013805 in the project TIN2008-02066: *Fuzzy Temporal Information Treatment in RDBMS* and by the Andalusian Government (Junta de Andalucía, Consejería de Innovación, Ciencia y Empresa) under project P07-TIC-03175, Representación y Manipulación de Objetos Imperfectos en Problemas de Integración de Datos: Una Aplicación a los Almacenes de Objetos de Aprendizaje.

References

1. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM **26**, 832–843 (1983)
2. Billiet, C., Pons, J.E., Matthé, T., De Tré, G., Pons Capote, O.: Bipolar fuzzy querying of temporal databases. In: Lecture Notes in Artificial Intelligence, vol. 7022, pp. 60–71. Springer, Ghent, Belgium (2011)
3. Bolour, A., Anderson, T.L., Dekeyser, L.J., Wong, H.K.T.: The role of time in information processing: a survey. ACM SIGMOD Record **12**, 27–50 (1982)
4. Castillo-Ortega, R., Marín, N., Sánchez, D.: A fuzzy approach to the linguistic summarization of time series. Multiple-Valued Logic and Soft Computing **17**(2-3), 157–182 (2011)
5. Castillo-Ortega, R., Marín, N., Sánchez, D.: Linguistic query answering on data cubes with time dimension. Int. J. Intell. Syst. **26**(10), 1002–1021 (2011)
6. Chakravarthy, S., Kim, S.K.: Resolution of time concepts in temporal databases. Information Sciences **80**(12), 91 – 125 (1994)
7. Clifford, J., Tansel, A.U.: On an algebra for historical relational databases: two views. SIGMOD Rec. **14**, 247–265 (1985). DOI http://doi.acm.org/10.1145/971699.318922. URL `http://doi.acm.org/10.1145/971699.318922`
8. Van der Cruyssen, B., De Caluwe R.and De Tré, G.: A theoretical fuzzy time model based on granularities. EUFIT'97 pp. 1127–1131 (1997)
9. De Caluwe, R., Van der Cruyssen, B., De Tré, G., Devos, F., Maesfranckx, P.: Fuzzy time indications in natural languages interfaces, pp. 163–185. Kluwer Academic Publishers, Norwell, MA, USA (1997)
10. De Tré, G., De Caluwe, R., Van der Cruyssen, B.: Dealing with time in fuzzy and uncertain object-oriented database models. EUFIT'97 pp. 1157–1161 (1997)
11. Devos, F., Maesfranckx, P., De Tré, G.: Granularity in the interpretation of around in approximative lexical time indications. Journal of Quantitative Linguistics **5**, 167–173 (1998)
12. Dubois, D., HadjAli, A., Prade, H.: Fuzziness and uncertainty in temporal reasoning. Journal of Universal Computer Science **9**(9), 1168–1194 (2003)
13. Dubois, D., HadjAli, A., Prade, H.: Fuzziness and uncertainty in temporal reasoning. j-jucs **9**(9), 1168–1194 (2003)
14. Dubois, D., Prade, H.: Ranking fuzzy numbers in the setting of possibility theory. Information Sciences **30**, 183–224 (1983)
15. Dubois, D., Prade, H.: Possibility Theory: An Approach to Computerized Processing of Uncertainty. Plenum Press, New York (1988)
16. DuBois, D., Prade, H.: Processing fuzzy temporal knowledge. IEEE Transactions on Systems, Man, and Cybernetics **19**, 729–744 (1989). DOI 10.1109/21.35337
17. Dyreson, C., Grandi, F.e.a.: A consensus glossary of temporal database concepts. SIGMOD Rec. **23**, 52–64 (1994)
18. Dyreson, C.E., Snodgrass, R.T.: Supporting valid-time indeterminacy. ACM Trans. Database Syst. **23**, 1–57 (1998)
19. Etzion, O., A, G., Segev., A.: Temporal support in active databases. In: Proceedings of the Workshop on Information Technologies and Systems (WITS), pp. 245–254 (1992)
20. Galindo, J., Medina, J.M.: Ftsql2: Fuzzy time in relational databases. In: EUSFLAT Conf. '01, pp. 47–50 (2001)
21. Garrido, C., Marin, N., Pons, O.: Fuzzy intervals to represent fuzzy valid time in a temporal relational database. Int. J. Uncertainty Fuzziness Knowlege-Based Syst. **17**, 173–192 (2009)
22. Gert De Cooman: Possibility theory 1: The measure- and integral-theoretic groundwork. International Jour-

nal of General Systems **25**(4), 291–323 (1997)
23. Gert De Cooman: Possibility theory 2: Conditional possibility. International Journal of General Systems **25**(4), 325–351 (1997)
24. Gert De Cooman: Possibility theory 3: Possibilistic independence. International Journal of General Systems **25**(4), 353–371 (1997)
25. Jensen, C.S., Mark, L., Roussopoulos, N.: Incremental implementation model for relational databases with transaction time. IEEE Trans. Knowl. Data Eng. **3**, 461–473 (1991)
26. Jensen, C.S., Snodgrass, R.T., Soo, M.D.: The tsql2 data model (1994)
27. Klein, W.: Time in Language. Routledge, London, U.K. (1994)
28. Klopprogge, M.R., Lockemann, P.C.: Modelling information preserving databases: Consequences of the concept of time. In: Proceedings of the 9th International Conference on Very Large Data Bases, pp. 399–416. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1983)
29. McCluskey, E.: Introduction to the Theory of Switching Circuits. McGraw-Hill Book Company (1965)
30. McKenzie, J.E., Snodgrass, R.T.: Supporting valid time in an historical relational algebra: Proofs and extensions. (1981)
31. Medina, J., Pons, O., Cubero, J.: Gefred. a generalized model of fuzzy relational databases. Information Sciences **76**, 87–109 (1994)
32. Mitra, D., et al.: A possibilistic interval constraint problem: Fuzzy temporal reasoning. In: Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proc. of the Third IEEE Conference on, vol. 2, pp. 1434–1439 (1994)
33. Nagypál, G., Motik, B.: A fuzzy model for representing uncertain, subjective, and vague temporal knowledge in ontologies. In: On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, *LNCS*, vol. 2888, pp. 906–923. Springer, Heidelberg (2003)
34. Nascimento, M.A., Eich, M.H.: Decision time in temporal databases. In: Proceedings of the Second International Workshop on Temporal Representation and Reasoning, pp. 157–162 (1995)
35. Ohlbach, H.J.: Relations between fuzzy time intervals. International Symposium on Temporal Representation and Reasoning **0**, 44–51 (2004)
36. Ozsoyoglu, G., Snodgrass, R.: Temporal and real-time databases: a survey. Knowledge and Data Engineering, IEEE Transactions on **7**(4), 513 – 532 (1995)
37. Pawlak, Z., Grymala-Busse, J., Slowinski, R., Ziarko, W.: Rough Sets. Communications of the ACM **38**(6) (1995)
38. Pons, J., Billiet, C., Pons, O., de Tré, G.: A possibilistic valid-time model. In: Proceedings of the 14th International Conference on Information Processing and Management of Uncertainty on Knowledge-Based Systems (2012)
39. Pons, J., Bronselaer, A., Pons, O., de Tre, G.: Possibilistic evaluation of fuzzy temporal intervals. In: G. Sainz, J. Alcalá (eds.) Proceedings of the 14th International Conference on Information Processing and Management of Uncertainty on Knowledge-Based Systems. Valladolid, Spain (2012)
40. Pons, J.E., Bronselaer, A., De Tré, G., Pons, O.: Possibilistic evaluation of sets (2011). Submitted to the International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems
41. Qiang, Y., Asmussen, K., Delafontaine, M., De Tré, G., Stichelbaut, B., De Maeyer, P., Van de Weghe, N.: Visualising rough time intervals in a two-dimensional space. In: 2009 IFSA World Congress / EUSFLAT Conference, Proceedings (2009)
42. Rowe, L.A., Stonebraker, M.: The Postgres Papers. University of California at Berkeley, Berkeley, CA, USA (1987)
43. Sarda, N.L.: Extensions to sql for historical databases. IEEE Trans. Knowl. Data Eng. **2**, 220–230 (1990)
44. Schockaert, S., De Cock, M., Kerre, E.: Fuzzifying allen's temporal interval relations. Fuzzy Systems, IEEE Transactions on **16**(2), 517 –533 (2008)
45. Shackle, G.: Decision, order and time in human affairs. Cambridge University Press (1961)
46. Zadeh, L.: Fuzzy sets. Information and Control **8**, 338–353 (1965)