Dealing with temporal indeterminacy in relational databases: An AI methodology

Dealing with Temporal Indeterminacy in Relational Databases: an AI methodology

Luca Anselma[1], Luca Piovesan[2], and Paolo Terenziani[2]

[1]Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10139 Torino, Italy
[2]DISIT, Univ. Piemonte Orientale "A. Avogadro", Alessandria, Italy
luca.anselma@di.unito.it, luca.piovesan@uniupo.it, paolo.terenziani@uniupo.it

Abstract. Time is pervasive of the human way of approaching reality, so that it has been widely studied in many research areas, including AI and relational Temporal Databases (TDB). While temporally imprecise information has been widely studied by the AI community, only few approaches have faced temporal indeterminacy (in particular, "don't know exactly when" indeterminacy) in TDBs. Indeed, as we will show in this paper, the treatment of time in general, and of temporal indeterminacy in particular, involves the introduction of *implicit* forms of data representation in TDBs. As a consequence, we propose a *new AI-style methodology* to cope with temporal indeterminacy in TDBs. Specifically, we show that typical AI notions and techniques, such as making explicit the *semantics* of the representation formalism, and adopting *symbolic manipulation* techniques based on such a semantics, can be fruitfully exploited in the development of a "*principled*" treatment of indeterminate time in relational databases.

Keywords: Temporal data, data representation and semantics, query semantics, symbolic manipulation

1 Introduction

Time is pervasive of reality, and plays a fundamental role in many intelligent tasks, so that it has been widely investigated by the AI community, which, from its early years, has developed many different methodologies for temporal representation and temporal reasoning. Starting from the middle 80's, also the scientific DB community has started to recognize that time has a special status with respect to the other data, so that its treatment within a *relational* database context requires dedicated techniques [Snodgrass, 00; Snodgrass et al., 95]. Such a consideration has led to the development of many different approaches to cope with time in the area of temporal *relational* databases (TDB in the following; see, e.g., [Wu et al., 98], [Liu & Özsu, 2009]). Just as an example, the 1998 cumulative bibliography about TDBs refers more than 2000 papers [Wu et al., 98]. Such approaches pointed out a quite wide range of solutions, proposing, e.g., different data models, and algebraic operations to query them.

However, TDB approaches have developed quite completely in an independent way with respect to AI methodologies. This is probably due to the fact that, to the best of our knowledge, no TDB approach has explicitly taken into account the fact that, while adding time to a relational DB, one adds ***implicit knowledge*** (i.e., the *semantics* of time) in it. This is particularly true in case temporal indeterminacy is considered (i.e., "*don't know exactly when*" indeterminacy [Dyreson, 09]), since indeterminacy give rise to many different alternative possibilities, and, for the sake of space and computational efficiency, no TDB approach makes all of them explicit. In this paper we argue that, since a high degree of ***implicit information*** is present in temporally indeterminate DB data, a temporal indeterminate DB is indeed quite close to a (simplified) *knowledge base*, so that AI techniques can be exploited to properly cope with it. In this paper, we propose an AI-based methodology to cope with temporal indeterminacy:

(i) We formally define and extend the *snapshot semantics* for temporal data [Snodgrass et al., 95] to cope also with temporal indeterminacy,

(ii) We propose a 1-Normal-Form ***representation model*** for "*interval-based*" temporal indeterminacy

(iii) We analyse the ***semantics*** of the representation model, showing that (at least) two alternatives are possible

(iv) We face the task of defining the relational algebraic operators (which perform ***symbolic manipulation*** on the model) to query the representational model, for both the alternative semantics, showing that only with one of them it is possible to devise a relational algebra which is both *closed* with respect to the model and *correct* with respect to the semantics.

Result (iv) enforces the core message of our approach: in TDBs, the representational model contains *implicit* (temporal) information. Thus, AI techniques could\should be used to analyse its *semantics*, and to devise *algebraic operators* that perform *symbolic manipulation* on the representational model, *consistently* with the devised semantics. In other words, in this paper we propose the first (to the best of our knowledge) AI approach in which an AI methodology is developed to be applied to the relational DB context, to cope with time and temporal indeterminacy.

The paper is organized as follows. In Section 2, we briefly overview the related work, focusing on the TDB approaches to temporal indeterminacy. In Section 3, informally introduce our AI-style methodology to cope with temporal indeterminacy in TDBs, motivating it. In Section 4, we start with the technical contributions, proposing a "functional" (*data* and *query*) semantics for determinate time in TDB. In Section 5, we extend such a semantics to cover temporal indeterminacy. In Section 6, we propose a compact representation (in First Normal Form - 1NF for short) for an important class of temporally indeterminate data, and define temporal algebraic operators to query it, consistently with the semantics of the representation. In Section 7, we provide experimental results, showing that our treatment of temporal indeterminacy only adds a negligible overhead to the "standard" TSQL2 treatment of determinate temporal data. Finally, Section 8 contains conclusions and future work. Appendix.1 briefly presents the "consensus" BCDM semantics for determinate time, elaborated by the TDB community [Snodgrass et al., 95].

2 TDB approaches to valid time temporal indeterminacy

Many different approaches have been devoted to the treatment of time in TDBs. One of the first milestone was the distinction between the time when facts are inserted\deleted into\from the DB (termed *transaction time*), and the time when such facts occurred in the modelled mini-world (termed *valid time*) (consider, e.g., [Snodgrass & Ahn 86]). In the following, we only consider the latter. Despite their variety, until now most TDB approaches have focused on individual occurrences of facts, whose valid time is *exactly known* (i.e., with *determinate* time). However, as well known in the AI field, in many real-world cases the exact time of occurrence of facts is not known, and can only be approximated, so that *temporal indeterminacy* (i.e., "*don't know exactly when*" indeterminacy [Dyreson, 09]) has to be faced. Temporal indeterminacy is so important that "support for temporal indeterminacy" was already one of the eight explicit goals of the data types in TSQL2 [Snodgrass et al., 95], the milestone "consensus" approach devised by the TDB community. In effect, temporal indeterminacy in TDBs has various possible sources, including scale, dating techniques, future planning, unknown or imprecise event times, clock measurements (this list is not exhaustive, and is taken from TSQL2 book [Snodgrass et al., 95]). Due to the fact that temporal indeterminacy is pervasive in many application domains, since the 80's AI a plethora of approaches has been devised to cope with it (just to mention few examples, consider the early surveys in [Vila, 94; Allen, 91; Emerson, 90]). However, in the area of relational databases, the number of approaches coping with temporal indeterminacy is more restricted (see e.g. the surveys in [Jensen & Snodgrass, 96; Dyreson, 09]) and current approaches have several limitations.

In the earliest TDB work on temporal indeterminacy, an indeterminate instant was modeled with a set of possible chronons [Snodgrass, 82]. Dutta [Dutta, 89] introduced a fuzzy set approach. Gadia et al. [Gadia et al., 92] proposed a model to support value and temporal incompleteness. In the TSQL2 "consensus" book [Snodgrass et al., 95], Chapter 18 presents a 1NF data model for temporal indeterminacy and an extension of SQL, while it does not provide a relational algebra. Dyreson and Snodgrass [Dyreson & Snodgrass, 98] and Dekhtyar et al. [Dekhtyar et al., 01] have proposed probabilistic approaches coping with different forms of temporal indeterminacy. Dyreson and Snodgrass cope with valid-time indeterminacy by associating a period of indeterminacy with a tuple. A period of indeterminacy is a period between two indeterminate instants, each one consisting of a range of granules and of a probability distribution over it. However, in such an approach, no relational algebra is proposed to query temporally indeterminate data. Dekhtyar et al. introduce temporal probabilistic tuples to cope with a quite specific for of temporal indeterminacy, concerning instantaneous events only (i.e., with data such as "the tuple d is in relation r at some point of time in the interval $[t_i, t_j]$ with probability between p and p_0"), and provide algebraic relational operators for their data model. Anselma et al. [Anselma et al, 13] have proposed a general semantic model for temporal indeterminacy in

TDBs. They identified different forms of temporal indeterminacy, and proposed a family of achievable representational models and algebrae for such forms. However, such an approach is semantic-oriented, abstract and not in 1NF (thus inefficient and not suitable for a direct implementation). A 1NF approach for a form of temporal indeterminacy has been proposed in [Anselma et al., 16], but no semantics for the model has been presented.

3 Towards an AI-style semantic-based methodology to cope with time in relational DBs

A premise is important, when starting a discussion about the *semantics* of temporal relational DBs. Indeed, seen from an AI perspective, a "traditional" non-temporal relational database is just an elicitation of all and only (given the *closed-world assumption* [Minker, 82]) the facts that are true in the modeled mini-world. In such a sense, the semantics of a non-temporal DB is "trivial", since the DB do not contain any implicit data\information. Since all the data are explicit, no "AI-style" reasoning mechanism is required, and (algebraic) query operators are enough to *extract* the relevant data from a DB.
However, such an "easy" scenario drastically changes when *time* is introduced into DBs, by associating with each fact its valid time, i.e., the time when it holds\occurs. Roughly speaking, in such a case, eliciting explicitly all true facts would correspond ***to elicit, for each possible point in time, all the facts that hold at that point***. Despite the extreme variety of TDB approaches in the literature, almost the totality of them is based, explicitly or (in many cases) implicitly, on the above idea, commonly termed "***snapshot semantics***": a temporal database is a set of "standard" (non-temporal) databases, each one considering a snapshot of time, and eliciting all facts (tuples) true in the modelled mini-world at that time. As an example, the "consensus" TSQL2 presents the BCDM semantic model, which supports the "snapshot semantics" mentioned above, and proves that such a semantics underlies many different approaches in the literature [Snodgrass et al., 95]. For the sake of completeness, the BCDM semantics is briefly reported in Appendix 1.

Of course, for space and time efficiency reasons, no TDB approach in the literature directly implements temporal databases making all data explicit: *representational models* are used to encode facts in a more compact and efficient form. Notably, this is a major departure from "traditional" DB concepts: a TDBs is not just an elicitation of all facts that holds in the modelled mini-world, but a ***compact implicit representation*** of them. This consideration becomes even more important while considering temporal indeterminacy. In TDBs, temporal indeterminate facts are facts for which the time of occurrence can only be approximated. Therefore, they may involve many different alternative possibilities and, for the sake of space and computational efficiency, no TDB approach aims at *explicitly* storing and managing all of them. As a consequence, a high degree of ***implicit information*** is present in all TDB data models for temporal indeterminacy. The adoption of representation model that are not fully "explicit", but involve a degree of implicit information, makes a temporal

indeterminate TDB closer to the AI notion of ***knowledge base*** (in the sense that implicit information is involved). As a consequence, in this paper, we suggest that a new, "semantic-based" and "AI-based" methodology should be used, when approaching indeterminate time in relational DBs. First of all,

(M1) *a semantics for making explicit the intended meaning of the representational models must be devised.*

In such a context, the algebraic query operators cannot simply select and extract data (since some data are implicit). Making all data explicit before\while answering queries is certainly not a good option (for the sake of space and time efficiency of the approach). As a consequence,

(M2) *algebraic operators must operate on the (implicit) representation*

(M3) *algebraic operators must provide an output expressed in the given representation (i.e., the representation formalism must be closed with respect to the algebraic operators)*

(M4) *algebraic operators must be correct with respect to the semantics of the representation*

In the following, we make the above discussion more concrete and formal. We start from the definition of a semantics for determinate time, and then we extend it to consider temporal indeterminacy.

4 Snapshot semantics for Determinate Time DBs: a "functional" perspective

In BCDM [Snodgrass et al., 95], a semantics for ***determinate time*** TDBs is provided (see also Appendix 1). It encodes the "*snapshot semantics*" discussed above, and has been proven to encompass the semantics of many different TDB approaches [Snodgrass et al., 95]. However, such a semantics is deliberately "*operational*", aiming at being not far from possible implementations. In this section we propose a "functional" specification of the snapshot semantics, which is more abstract and, in the meanwhile, more suitable for

(i) being extended to cope with temporal indeterminacy and

(ii) specifying the semantics of temporal algebraic operators in terms of their non-temporal counterparts.

4.1 Data Semantics

We first introduce the notion of tuple, relation, and database. We then move to the definition of time, and define the notion of (semantics of) a temporal database.

Definition 1. (non-temporal) Database, Relation, Tuple. A (non-temporal) relational database *DB* is a set of relations over the relational schema $\sigma = (R_1{:}s_i\ , ..., R_k{:}s_j)$ where $s_i, ..., s_j \in S$ are the sorts of $R_1, ..., R_k$, respectively. A relation $R(x_1, ..., x_k){:}s$

of sort $s \in S$ is a sequence of attributes x_1, ..., x_k each with values in a proper domain $D_1,\ldots D_k$. An instance $r(R{:}s)$ of a relation $R(x_1, ..., x_k)$ of sort $s \in S$ is a set $\{a_1, \ldots, a_n\}$ tuples, where each tuple a_i is a set $<v_1,\ldots,v_k>$ of values in $D_1 \times \ldots \times D_k$. ■

Notation. In the following, we denote by $\mathbb{DB}_\sigma$ the domain of all possible database instances over a schema σ. ■

As in many TDB approaches, including TSQL2 [Snodgrass et al., 95], and in the BCDM "consensus" semantics [Snodgrass et al., 95], we assume that time is discrete and bounded.

Definition 2. Temporal domain D_T. We assume a limited precision for time, and call *chronon* (as in TSQL2 [Snodgrass et al., 95]) the basic time unit. The domain of chronons is totally ordered and isomorphic to a subset of the domain of natural numbers. The domain of valid times D_T is given as a set $D_T=\{c_1,\ldots,c_k\}$ of chronons. ■

In the (commonly agreed) snapshot semantics, a temporal database is a set of conventional (non-temporal) databases, one for each chronon of time. In this paper, we propose to specify such a concept formally through the introduction of functions, relating each time to the facts holding\occurring at that time.

Definition 3. Temporal database (semantic notion). Given a relational schema $\sigma = (R_1{:}s_i , ..., R_k{:}s_j)$ a temporal database DB^T is a function f_{σ,D_T}: $D_T \rightarrow \mathbb{DB}_\sigma$ ■

Analogously a ***temporal relation*** is a function from D_T to the tuples that hold at each chronon in D_T.

Definition 4. Time slice. Given a temporal database DB^T and a temporal relation $r^T \in$ in DB^T, and given a chronon $c \in D_T$, we define the time slice of DB^T (denoted by $DB^T(c)$) and of r^T (denoted by $r^T(c)$) the result of the application of the functions DB^T and r^T to the chronon c. ■

Example. 1. As a simple running example, let us consider a simple database DB^T_1 modeling patient symptoms. The database contains a unique relation SYM of schema <Patient,Symptom,Value> and models two facts:

(f1) John had high fever from 10 to 12 of 1/1/2018

(f2) Mary had moderate fever from 11 to 13 of 1/1/2018

(in the example, we assume that chronons are at the granularity of hours, and hour 1 represent the first hour of 1/1/2018).

The temporal database (semantic notion) modeling such a state of affairs is the following (for the sake of clarity and simplicity, we omit the chronons in D_T for which no tuple hold, and we omit the name of the relation(s)).

10 → {<John, fever, high>}

11 → {<John, fever, high>,<Mary, fever, moderate>}

12 → {<John, fever, high>,<Mary, fever, moderate>}

13 → {<Mary, fever, moderate>}

In this example $DB^T_1(10) = SYM^T(10) = \{<John, fever, high>\}$ ■

Notably, definition 3 above is a purely "semantic" definition. Other definitions of the snapshot semantics for TDBs, such as the one in BCDM, are more "operational" and are closer to actual representations\implementations (see Appendix 1). Indeed, our "functional" definition is similar to the "relation-stamped representation" of abstract temporal databases in the work by Chomicki and Toman [Chomicki & Toman, 98].

4.2 Query semantics

The semantic of queries is commonly expressed by specifying in terms of relational algebraic operators. Codd designated as complete any query language that was as expressive as his set of five relational algebraic operators: relational union ($\cup$), relational difference ($-$), selection (σ_P), projection (π_X), and Cartesian product ($\times$) [Codd, 72]. Different approaches have generalized such operators to cope also with temporal databases. Though quite different approaches have been proposed (depending on the chosen representation for a temporal database), there is a common agreement that temporal algebraic operators (i) *should behave exactly as Codd's non-temporal ones, at each point (chronon) of time*. Roughly speaking, such a requirement is usually formally expressed in TDBs by the *reducibility* property, stating that temporal algebraic operators should reduce to standard Codd's operators in case time is removed (see [McKenzie & Snodgrass, 91; Snodgrass et al., 95], and also Appendix 1). Notably, property (i) above (as well as *reducibility*) has also very important practical implications, since it grants the possible *interoperability* of temporal databases with standard non-temporal DBs [Snodgrass et al., 95]. Given our "functional" definition of temporal databases above, in our approach we can formally define such a property (see below).

Definition 5. **Relational algebraic operators on temporal databases ("semantic" notion).** Denoting by Op^C a Codd's operator, and by Op^T its corresponding temporal operator, Op^T must be defined in such a way that the following holds

$\forall c \in D_T \ \ (Op^T(r^T,s^T)\ (c)) \ = Op^C(r^T(c),s^T(c)))$ ■

(In Definition 5 above, we assume that r^T and s^T are temporal relations in a temporal database DB^T, and that Op is a binary operator. $r^T(c)$ represents the *time slice* of r^T at the chronon *c*. The definition of unary operators is analogous).

Of course, the above "purely semantic" definition of temporal relational algebraic operators is highly inefficient, since snapshot(s) of the underlying relation(s) at every single chronon (e.g., day, millisecond) are computed. As a consequence, more "operational" definitions of algebraic operators have been proposed in the literature. Notably, however, the "commonly agreed" BCDM definition of the semantics of algebraic operators (see Appendix 1) is consistent with definition 5 above.

5 Snapshot semantics for temporal indeterminacy in TDB

In TDBs, the notion of temporal indeterminacy is usually paraphrased as "don't know exactly when" indeterminacy (consider, e.g., the Encyclopedia survey in [Dyreson, 09]): facts hold at times that are not exactly known. An example is reported in the following:

Example. 2. As a simple running example, let us consider a simple database DB^{T}_{1} modeling patient symptoms. The database contains a unique relation SYM^{I} of schema <Patient,Symptom,Value> and models two facts:

(f1) John had high fever at 10 and 11, and possibly at 12, or 13, or both.

(f2) Mary had moderate fever at 12 and 13, and possibly at 11

(in the example, we assume that chronons are at the granularity of hours, and hour 1 represent the first hour of 1/1/2018).

5.1 Data Semantics of Indeterminate Time DBs

Of course, we can still retain the definition of the temporal domain D_T provided in section 2. However, the definition of an indeterminate temporal database is different: informally speaking, an indeterminate TDB is simply a set of alternative (determinate) TDBs, each one encoding one of the different possibilities. Technically speaking, it requires the introduction of a set of functions.

Definition 6. Indeterminate temporal database (semantic notion). Given a relational schema $\sigma = (R_1:s_i , ..., R_k:s_j)$ an indeterminate temporal database DB^{IT} is a set $DB^{IT}=\{f_1, \ldots, f_k\}$ of functions $f_{\sigma,D_T}: D_T \rightarrow DB_\sigma$ ■

Analogously, a temporally indeterminate relation r^{IT} is a set $S(r^{IT})$ of functions from D_T to the set of tuples of r^T that hold at each chronon in D_T. As an example, eight functions are necessary to cover all the alternative possibilities (henceforth called ***scenarios***) for Example 2.

Example. 2 (cont).

The indeterminate temporal database DB^{IT} (semantic notion) modeling Example 2 is the following (for the sake of brevity, we denote with "J" the tuple <John, fever, high> and with "M" the tuple <Mary, fever,moderate>).

f_1	f_2	f_3	f_4
10 → {J}	10 → {J}	10 → {J}	10 → {J}
11 → {J}	11 → {J}	11 → {J}	11 → {J}
12 → {M}	12 → {J,M}	12 → {M}	12 → {J,M}
13 → {M}	13 → {M}	13 → {J,M}	13 → {J,M}

f_5	f_6	f_7	f_8
10 → {J}	10 → {J}	10 → {J}	10 → {J}
11 → {J,M}	11 → {J,M}	11 → {J,M}	11 → {J,M}
12 → {M}	12 → {J,M}	12 → {M}	12 → {J,M}
13 → {M}	13 → {M}	13 → {J,M}	13 → {J,M} ■

For the technical treatment that follows, it is useful to introduce the notion of alternative slice.

Definition 7. Scenario slice. Given an indeterminate temporal database DB^{IT} = $\{f_1,\ldots,f_k\}$ and a temporal relation $r^{IT}\in$ in DB^{IT}, and given any $f \in\{f_1,\ldots,f_k\}$, we define the scenario slice f of DB^{IT} (denoted by DB_f^{IT}) and of r^{IT} (denoted by r_f^T) the determinate temporal database and the determinate temporal relation obtained by considering only the scenario f for DB^{IT} ■

For example, considering Example 2 above, $DB_{f_1}^{IT}=SYM_{f_1}^{IT}$= {10→{J}, 11→{J}, 12→{M}, 13→{M}}.

5.2 Query semantics

Of course, for the algebraic query operators, we can still retain all the general requirements discussed so far for determinate time. However, we have to generalize the above approach, to consider the fact that a set of *alternative* (determinate) temporal databases (*scenarios*) are involved. Therefore, given two temporally indeterminate relations r^{IT} and s^{IT}, binary temporal algebraic operators must consider, at each chronon, all the possible combinations of the scenarios $f_r\in S(r^{IT})$ of r^{IT} and $f_s\in S(s^{IT})$ of s^{IT}. At any temporal chronon c, the result of the temporal operator should be the result obtained through the application of the corresponding Codd's operator in each pair of scenarios, considered at time c.

Definition 8. **Relational algebraic operators on indeterminate temporal databases ("semantic" notion).** Denoting by Op^C a Codd's operator, and by Op^{IT} its corresponding temporal operator for indeterminate time, Op^{IT} must be defined in such a way that the following holds

$\forall c\in D_T\ (Op^{IT}(r^{IT},s^{IT})\ (c)) = \{\{Op^C(f_r(c), f_s(c)))\} \setminus f_r\in S(r^{IT}) \wedge f_s\in S(s^{IT})\}$ ■

(In Definition 8, r^{IT} and s^{IT} are temporal relations in a temporally indeterminate database DB^{IT}, and Op is a binary operator. $f_r(c)$ represents the *time slice* at the chronon c of the scenario f_r of r^{IT}. The definition of unary operators is simpler).

Indeed, indeterminacy intrinsically involves alternative possibilities, so that the approach usually followed in the TDB area (as well as, of course, in AI) is the introduction of *modalities*, to ask for *possible* (i.e., valid in at least one of the possible scenarios) or *necessary* (i.e., valid in all the possible scenarios) answers (see, e.g., [Anselma et al, 13]).

With the introduction of the modalities (POSS – for possible) and NEC, the semantics of algebraic query operators in the indeterminate (temporal) context can be still be easily expressed in terms of their Codd's non-temporal counterparts, as shown in definition 9 below.

Definition 9. **Relational algebraic operators on indeterminate temporal databases ("semantic" notion).** Denoting by Op^C a Codd's operator, and by Op^{IT} its corresponding temporal operator for indeterminate time Op^{IT} must be defined in such a way that the following holds

$$\forall c \in D_T \; POSS\,(Op^{IT}(r^{IT},s^{IT})\,(c)) = \bigcup_{f_r \in S(r^{IT}) \wedge f_s \in S(s^{IT})} Op^C(r_{f_r}{}^{IT}(c),\, s_{f_s}{}^{IT}(c))$$

$$\forall c \in D_T \; NEC\,(Op^{IT}(r^{IT},s^{IT})(c)) = \bigcap_{f_r \in S(r^{IT}) \wedge f_s \in S(s^{IT})} Op^C(r_{f_r}{}^{IT}(c),\, s_{f_s}{}^{IT}(c)) \;\blacksquare$$

(In definition 9 above, r^{IT} and s^{IT} are temporal relations in a temporally indeterminate database DB^{IT}, and Op is a binary operator. $f_r(c)$ represents the *time slice* at the chronon *c* of the scenario f_r of r^{IT}. The definition of unary operators is simpler).

Roughly speaking, the above definition dictates that, to preserve the snapshot semantics in the indeterminate context, the evaluation of a binary temporal operator Op^{IT} (r^{IT},s^{IT}) must provide the result that would be obtained by

(i) Applying the corresponding Codd's operator at each chronon, and considering each combinations $<f_r,f_s>$ of the scenarios $S(r^{IT})$ of r^{IT} and $S(s^{IT})$ of s^{IT} (please notice that $r_{f_r}{}^{IT}(c)$ denotes the *time slice* at the chronon *c* of the relation r^{IT}, considering only the scenario f_r).

(ii) In the NEC modality, (at each chronon) the intersection among the results of such an application in the different scenarios is performed, to grant that the facts hold in all the scenarios. In the POSS modality, (at each chronon) union is used, since we want to report as output (at the given chronon) all the facts that hold in at least one scenario.

Notably, we regard definition 9 above as one of the major results of this paper: *until now, no approach in the TDB community has been able to clarify the semantics of temporal algebraic operators on indeterminate time in terms of their Codd's counterparts*.

But, obviously, this is query data and query ***semantics***: a direct implementation of the data model and algebraic operators defined so far would be highly inefficient, as regards both space and time. As a consequence, "compact" implementations should be performed. We address this issue in the next section, showing that, in any case, an in-depth semantic analysis in necessary to provide correct implementations.

6 Possible "compact" approaches to temporal indeterminacy

Very different realizations of temporal relational databases for *determinate time* have been proposed in the literature. Actually, all of them (except few "pioneering"

approaches) respect the "snapshot" semantics, and provide an efficient implementation for it. The large majority of such approaches enforce at least two key requirements to achieve efficiency:

(i) The First Normal Form (1NF) is used to represent data[1]

(ii) Temporal algebraic operators directly manipulate the representation

In this section we adopt the same requirements to face temporal indeterminacy.

The most frequently adopted representational model to cope with (valid) time in a compact and 1NF way is the ***interval-based*** representation (consider, e.g., the TSQL2 "consensus" representational model). A time interval (compactly modelled by a starting and an ending time) is associated with each temporal tuple, to denote that the (fact represented by the) tuple holds in each time points in the interval. In the indeterminate time context, such an interval-based representation has also been used, e.g., in [Snodgrass & Dyreson, 98; Anselma et al., 16]: four temporal attributes (say T1, T2, T3, and T4) are associated with each temporal tuple (see Definition 10).

Definition 10. Temporally indeterminate Database, Relation, Tuple (representation level). At the representation level, a temporally indeterminate relational database DB^{IT} can be modeled by a set of (temporally indeterminate) relations over the relational schema $\sigma = (R_1{:}s_i\ , ..., R_k{:}s_j)$ where s_i, ..., $s_j \in S$ are the sorts of R_1, ..., R_k, respectively. A relation $R(x_1, ..., x_k|T1,T2,T3,T4){:}s$ of sort $s \in S$ is a sequence of non-temporal attributes x_1, ..., x_k each with values in a proper domain $D_1,\ldots D_k$, and temporal attributes T1,T2,T3,T4 with domain D_T. An instance $r(R{:}s)$ of a relation relation $R(x_1, ..., x_k|T1,T2,T3,T4){:}s$ is a set $\{t_1, \ldots, t_n\}$ tuples, where each tuple t_i is a set $<v_1,\ldots,v_k|t_1,t_2,t_3,t_4>$ of values in $D_1\times\ldots\times D_k\times D_T \times D_T \times D_T \times D_T$. ■

Example 3. In the temporally indeterminate context, the relation SYM (called SYM^{IT}) may be represented with the schema <Patient,Symptom,Value| T_1,T_2,T_3,T_4> ■

Intuitively and ***roughly*** speaking, the semantics of such a compact 1NF "interval-based" representation of temporal indeterminacy is the following:

(sem1) the fact represented by the tuple $\mathbf{<v_1,\ldots,v_k|t_1,t_2,t_3,t_4>}$ occurs possibly in $[t_1,t_2)$ and in $[t_3, t_4)$, and certainly in $[t_2,t_3)$.

In the following, we show that a "rough" semantics like (sem1) above is not enough: it must be fully formalized (e.g., in terms of the sematic model proposed in Section 5 above) as a starting point for devising a "proper" representational model and algebra, following the methodological requirements M1—M4 above.

6.1 "Single occurrence" semantics

A first way of interpreting the "ambiguous" semantics (sem1) above is the following:

[1] A relation is in first normal form if and only if the domain of each attribute contains only atomic (indivisible) values, and the value of each attribute contains only a single value from that domain [Elmasri et al, 03]

(sem1') the fact represented by the tuple $<v_1,\ldots,v_k|t_1,t_2,t_3,t_4>$ certainly occurs in the time interval $[t_2,t_3)$, but it may start before, in a chronon in the interval $[t_1,t_2)$ and end after, in a chronon in the interval $[t_3,t_4)$.

In such a semantics, all the scenarios includes the chronons $[t_2,t_3)$, as well as (possibly) a set of chronons in $[t_1,t_2) \cup [t_3,t_4)$, provided that such chronons extend the interval $[t_2,t_3)$ on the left, or on the right, or in both directions, without creating any gap. This notion can be formalized in terms of scenarios as follows.

Definition 11. Semantics sem1'. The semantics of a tuple $<v_1,\ldots,v_k|t_1,t_2,t_3,t_4>$ is the set of functions $F = \{f': [c_s,c_e] \rightarrow \{ <v_1,\ldots,v_k >\} \setminus t_1 \leq c_s \leq t_2 \wedge t_3 < c_e < t_4\}$

This is, probably, the most intuitive notion of temporal indeterminacy in TDBs: each tuple represents ***a single occurrence*** of a fact, and temporal indeterminacy concerns the starting and ending chronons of it. In such a context, it looks natural to impose $t1 \leq t2 < t3 \leq t4$, thus granting that there is at least one chronon in which the fact certainly occurs (see, e.g., [Dyreson & Snodgrass, 98]).

Example 4. Given the temporally indeterminate relation SYM^{IT}, with the semantics (sem1') above, the fact

(f2) Mary had moderate fever at 12 and 13, and possibly at 11

can be represented by the tuple <Mary,fever,moderate|11,12,14,14>
The semantics of such a tuple is (where "M" stands for <Mary,fever,moderate>):

	11 → {M}	
12 → {M}	12 → {M}	
13 → {M}	13 → {M}	■

Notably, if we assume the semantics (sem1'), the fact (f1)

(f1) John had high fever at 10 and 11, and possibly at 12, or 13, or both

cannot be represented in the representational model: as a matter of fact, in the semantics (sem1') the tuple

<John,fever,high|10,10,12,14>

would be interpreted as the compact representation of the semantics below:

10 → {J}	10 → {J}	10 → {J}
11 → {J}	11 → {J}	11 → {J}
	12 → {J}	12 → {J}
		13 → {J}

while the scenario <10 → {J}, 11 → {J}, 13 → {J}> would not be part of the semantics of the representation. Indeed, if we assume (sem1'), each tuple represents a single occurrence of a fact, while the scenario <10 → {J}, 11 → {J}, 13 → {J}> above represents two separate occurrences, one at [10,12), and one at [13,14).

Of course, the specification of the semantics if fundamental also for the definition of the algebraic operators. In particular, we must grant that such operators

(i) Are *correct* with respect to the semantics, and
(ii) Are *closed* with respect to the representational model

Notably, if we assume the semantics (sem1') for the representational model in Definition 10, there is no way to satisfy both requirements (i) and (ii)[2]. A trivial counter-example is discussed in the following, considering algebraic difference.

Example 5. Consider the difference between two relations r^{IT} and s^{IT} having the same schema $(A_1,\ldots,A_k|T_1,T_2,T_3,T_4)$. Let $r^{IT}=\{<a_1,\ldots,a_k|1,3,5,7>\}$ and $s^{IT}=\{<a_1,\ldots,a_k|3,3,8,8>\}$ (i.e., the two tuples are value-equivalent, and the tuple in s^{IT} is determinate, starts at 3 and ends at 7). In such a case the result of the difference r^{IT} -^{IT} s^{IT} should be a fact $a_1,\ldots,a_k$ which may not occur, or occur in {2}, or in {1,2}. A tuple with such a semantics cannot be represented in the given representation. Therefore, this example suffices to show that (correct) difference is not closed with respect with the given formalism (with semantics sem1' above). ■

6.2 "Independent chronons" semantics

A different way of interpreting the "rough" semantics (sem1) above is the following:

(sem1") the fact represented by the tuple $<v_1,\ldots,v_k|t_1,t_2,t_3,t_4>$ certainly holds in each chronon in $[t_2,t_3)$ (if any), and may hold in each one of the chronons between t_1 and t_2, and between t_3 and t_4

In such a semantics, each scenario includes the chronons $[t_2,t_3)$, as well as (possibly) a set of chronons in $[t_1,t_2) \cup [t_3,t_4)$. There are as many different scenarios as the cardinality of the power set of the chronons in $[t_1,t_2) \cup [t_3,t_4)$.

Definition 12. Semantics sem1". The semantics of a tuple $<v_1,\ldots,v_k|t_1,t_2,t_3,t_4>$ is the set of functions $F = \{f': C_S \rightarrow \{ <v_1,\ldots,v_k >\} \setminus C_S = \{t_2, t_2+1, t_2+2, \ldots,t_3-1\} \cup S'$, where $S' \in PS(\{t_1, t_1+1, t_1+2, \ldots,t_2-1\} \cup \{t_3, t_3+1, t_3+2, \ldots,t_4-1\})$, and PS(S) denotes the power set of a set S.

In such a semantics, there is no notion of single occurrence at all. One simply regards chronons independently of each other. In such a context, it looks natural to impose $t_1 \le t_2 \le t_3 \le t_4$ and $t_1 < t_4$ so that the fact must be possible in at least a chronon, but may also not be certain in any chronon (in case $t_2=t_3$).

Example 6. Given the temporally indeterminate relation SYM^{IT}, with the semantics (sem1") above, the fact (f1)

(f1) John had high fever at 10 and 11, and possibly at 12, or 13, or both

is represented in the representational model by the tuple

<John,fever,high|10,10,12,14>

which has the semantics discussed above, i.e.,

---

[2] Notably, it is possible to show that it is not possible to define correct algebraic operators closed with respect to the representational model also in case one admits the possibility that facts in the TDBs do not necessarily occur, i.e., imposing $t_1 \le t_2 \le t_3 \le t_4$ in the representational model.

10 → {J}	10 → {J}	10 → {J}	10 → {J}
11 → {J}	11 → {J}	11 → {J}	11 → {J}
	12 → {J}	13 → {J}	12 → {J}
			13 → {J}

■

With such a semantics for the representational model, it is possible to define correct and closed algebraic operators as follows:

Definition 13. **Algebraic operators for indeterminate time (independent chronons semantics)**. Let r and s denote relations of the proper sort and <v|t_1,t_2,t_3,t_4> a tuple with non-temporal part v and temporal part t_1,t_2,t_3,t_4.

$r \cup^{IT} s$ = { < v| t_1,t_2,t_3,t_4> | < v| t_1,t_2,t_3,t_4 > ∈ r ∨ < v| t_1,t_2,t_3,t_4 > ∈ s}

$r \times^{IT} s$ = { < v_r ∘ v_s| t_1,t_2,t_3,t_4 > | ∃ t'_1,t'_2,t'_3,t'_4 ∃ t''_1,t''_2,t''_3,t''_4 (< v_r| t'_1,t'_2,t'_3,t'_4 > ∈ r ∧
< v_s| t''_1,t''_2,t''_3,t''_4 > ∈ s ∧ t_1=max(t'_1, t''_1) ∧ t_4=min(t'_4, t''_4) ∧ t_1< t_4 ∧
let t_s=max(t'_2, t''_2) ∧ t_e=min(t'_3, t''_3)
if t_s≤ t_e then t_2= t_s ∧ t_3= t_e else t_2=t_3=t where t is any value in [t_1,t_4)}

$\pi^{IT}_X(r)$ = { < v| t_1,t_2,t_3,t_4 > | ∃v_r,t_1,t_2,t_3,t_4 (< v_r| t_1,t_2,t_3,t_4 > ∈ r ∧ v = $\pi_X(v_r)$ }

$\sigma^{IT}_P(r)$ = { < v|t > | < v|t > ∈ r ∧ $P(v)$ }

$r -^{IT} s$ = { < v| t_1,t_2,t_3,t_4> | (∃v, t_1,t_2,t_3,t_4 (< v| t_1,t_2,t_3,t_4 > ∈ r ∧
¬∃ t'_1,t'_2,t'_3,t'_4 (< v| t'_1,t'_2,t'_3,t'_4 > ∈ s))) ∨
(∃ t'_1,t'_2,t'_3,t'_4 ∃ t''_1,t''_2,t''_3,t''_4 (<v| t'_1,t'_2,t'_3,t'_4> ∈ r ∧ <v| t''_1,t''_2,t''_3,t''_4>∈s
∧ t_1,t_2,t_3,t_4 = difference([t'_1,t'_4), [t'_2,t'_3), [t''_1,t''_4),[t''_2,t''_3))}

where *difference* can be defined by the following function (where *s* a function that returns the starting point of an interval and *e* returns the ending point):

difference(p1,n1,p2,n2)

(1) if (p1 ⊆ n2) then return ∅
(2) else if (p1 ∩ n2 = ∅) then
if ((n1-p2) ≠∅) then return <s(p1),s(n1-p2),e(n1-p2),e(p1)>
else return <s(p1),t,t,e(p1)> where t is any value s(p1)≤t≤e(p1)
(3) else if (p1 ⊃ n2) then
if (s(n1)<s(p2)) let r1= <s(p1),s(n1-p2),e(n1-p2),s(n2)>
else let r1= <s(p1),t,t,s(n2)> where t is any value s(p1) ≤t≤s(n2)
if (e(n1)>e(p2)) let r2=<e(n2),s(n1-p2),e(n1-p2),e(p1)>
else let r2=<e(n2),t,t,e(p1)> where t is any value e(n2)≤t≤e(p1)
return {r1, r2}
(4) else
if ((n1-p2) ≠∅) return <s(p1-n2),s(n1-p2),e(n1-p2),e(p1-n2)>
else return <s(p1-n2),t,t,e(p1-n2)> where t is any value s(p1-n2)≤t≤e(p1-n2)

■

The difference function accepts as parameters two time intervals for the minuend (p1 and n1) and two time intervals for the subtrahend (p2 and n2). p1 and p2 are the

possible intervals, i.e., they contain the chronons that are in at least one scenario, and n1 and n2 are the necessary – certain – intervals, i.e., they contain the chronons that are in every scenario (thus n1⊆p1 and n2⊆p2). The function operates along the following idea: if a chronon is both in the minuend and in the subtrahend, and in the subtrahend such a chronon is (i) necessary (i.e., it belongs to n2), it will not be in the result, (ii) only possible (i.e., it belongs to p2 but not to n2), it will be possible in the result. From (i) and the fact that n1⊆p1, descends part (1) of the difference function, from (ii) descends part (2), from (i) and (ii) and the fact that n2⊆p1 descends part (3) and, in particular, since n2⊆p1 the minuend "breaks" into two (pairs of) intervals, from (i) and (ii) and the fact that n2⊈p1 descends part (4).

In such a context, the NEC and POSS operators can be defined as follows:

Definition 14. **NEC and POSS operators for indeterminate time (independent chronons semantics)**. Let r denote an indeterminate time relation and $<v|t_1,t_2,t_3,t_4>$ a tuple with non-temporal part v and temporal part t_1,t_2,t_3,t_4.
NEC(r) $= \{ < v| t_2,t_2,t_3,t_3> | < v| t_1,t_2,t_3,t_4 > \in r \wedge t_2<t_3\}$
POSS(r) $= \{ < v| t_1,t_1,t_1,t_4> | < v| t_1,t_2,t_3,t_4 > \in r \}$ ■

Roughly speaking, since the semantics of the representation is (sem1") above, the chronons in which a fact necessarily occurs\holds are the ones in $[t_2,t_3)$. A fact with a determinate time $[t_2,t_3)$ can be represented, in our formalism, by a tuple $<v| t_2,t_2,t_3,t_3>$, in which the intervals for possible chronons (i.e., $[t_2,t_2)$ and $[t_3,t_3)$) are empty, and the interval for necessary chronons is $[t_2,t_3)$. Analogously, the chronons in which the fact may occur (i.e., in which the fact possibly or necessarily occur) are the ones in [t1,t2) $\cup$ $[t_2,t_3) \cup [t_3,t_4)$, i.e., in $[t_1,t_4)$. A fact with determinate time $[t_1,t_4)$ can be represented, in our formalism, by a tuple $< v| t_1,t_1,t_1,t_4>$, in which the first interval for possible chronons (i.e., $[t_1,t_1)$) and the interval for necessary chronons (i.e., $[t_1,t_1)$) are empty, while the second interval for possible chronons is $[t_1,t_4)$. Notably, other semantically equivalent representations are possible.

Property 1. The algebraic operators in Definitions 13 and 14 are correct (with respect to the semantics defined so far) and are closed with respect to the representational model.
Proof (sketch). Let us consider, for example, the case of Cartesian Product. The closure of such an operation directly follows from its definition: the output is a set of tuples having as non-temporal values the concatenation of the non-temporal values being paired, and as temporal part a quadruple of values t_1,t_2,t_3,t_4 such that $t_1 \le t_2 \le t_3 \le t_4$ and $t_1<t_4$ (notably, in case $\max(t'_1, t''_1) \ge \min(t'_4, t''_4)$, the tuple is not part of the output). As regards correctness with respect to the semantics, we have to prove that, given any chronon $c \in D_T$,

(i) a tuple $t=<v \mid t_1,t_2,t_3,t_4 >$ belongs to $NEC(r \times^{IT} s)(c)$ if and only if it belongs to $\bigcap_{f_r \in S(r^{IT}) \wedge f_s \in S(s^{IT})} (r_{f_r}{}^{IT}(c) \times^{C} s_{f_s}{}^{IT}(c))$

(ii) a tuple $t=< v \mid t_1,t_2,t_3,t_4 >$ belongs to $POSS(r \times^{IT} s)(c)$ if and only if it belongs to $\bigcup_{f_r \in S(r^{IT}) \wedge f_s \in S(s^{IT})} (r_{f_r}{}^{IT}(c) \times^{C} s_{f_s}{}^{IT}(c))$

(where $\times^C$ denotes Codd's Cartesian Product). Let us consider, for short, the a part of the proof: if a tuple $t=< v \mid t_1,t_2,t_3,t_4 >$ belongs to NEC($r \times^{IT} s$)(c) , then it must belong to $\bigcap_{f_r \in S(r^{IT}) \wedge f_s \in S(s^{IT})} (r_{f_r}{}^{IT}(c) \times^C s_{f_s}{}^{IT}(c))$. By definition of NEC, all tuples in NEC(r) are of the form $< v \mid t_2,t_2,t_3,t_3>$. If $t=< v \mid t_2,t_2,t_3,t_3>$ and $t \in$ NEC($r \times^{IT} s$)(c), we have that (i) c must belong to $[t_2,t_3)$ and (ii) there must be a tuple $t' \in (r \times^{IT} s)$ such that $t'=< v \mid t_1,t_2,t_3,t_4>$. Given (i), the semantics (sem 1") implies that $c \rightarrow \{v\}$ belongs to all the scenarios of $(r \times^{IT} s)$; If such the tuple t' belongs to $(r \times^{IT} s)$, given the definition of $\times^{IT}$, there must be a tuple $t_r=<v_r \mid t^r_1,t^r_2,t^r_3,t^r_4> \in r$ and a tuple $t_s=<v_s \mid t^s_1,t^s_2,t^s_3,t^s_4> \in s$ such that $v=v_r \cdot v_s$ and $t_2=\max(t^r_2, t^s_2) \wedge t_3=\min(t^r_3, t^s_3)$ (i.e., $[t_2,t_3)= [t^r_2,t^r_3) \cap [t^s_2,t^s_3)$). Since $c \in [t_2,t_3)$ and $[t_2,t_3)= [t^r_2,t^r_3) \cap [t^s_2,t^s_3))$, we thus have that $c \in [t^r_2,t^r_3)$ and $c \in [t^s_2,t^s_3)$. Since $c \in [t^r_2,t^r_3)$, given (sem1"), we have that $c \rightarrow \{v_r\}$ belongs to all the scenarios $f_r \in S(r^{IT})$ of r^{IT}; Since $c \in [t^s_2,t^s_3)$, given (sem1"), we have that $c \rightarrow \{v_s\}$ belongs all the scenarios $f_s \in S(s^{IT})$ of s^{IT}. As a consequence, given the definition of Codd's Cartesian Product, in each combination of scenarios for r^{IT} and s^{IT}, $v=v_r \cdot v_s$ belongs to the output, so that $v=v_r \cdot v_s$ belongs to intersection of all the outputs.
The proof in the other direction is similar, and is not reported for the sake of brevity.
■

7 Experimental evaluations

As discussed in Section 3, the treatment of temporal indeterminacy involves the management of implicit data in the treatment of query answering (algebraic operators), so that AI symbolic manipulation techniques can be used. Specifically, in Section 6.2, we have defined new algebraic operators for indeterminate time, and we have proven their correctness with respect to the underlying semantics. In this Section, we discuss some experimental evaluations, showing that, though significantly more expressive, our approach only adds a negligible overhead with respect to TSQL2, in which only determinate (i.e., exact) valid time is considered (notably, in TSQL2 book, Chapter 18, also temporal indeterminacy is considered; however, no extension of algebraic operators to cope with indeterminacy is proposed).

We have implemented the data model and algebra discussed in Section 6.2 and the TSQL2 model and algebra to cope with determinate time, using PL/pgSQL stored procedures and PostgreSQL. We have experimentally evaluated the performance of our temporal algebra. Experiments have been performed on an Intel Core i7-6700HQ CPU, 2.60 GHZ, 8 GB RAM, OS Windows 10, using PostgreSQL 10 DBMS with default settings (effective cache size 500 MB, 8 KB page size and 500 MB shared buffers).

We have generated datasets of different dimension, to test and compare the scalability of our approach, with respect to TSQL2. In particular, we have focused our experiments on the algebraic operators that manipulate the temporal attributes, i.e., Cartesian Product and difference.

As regard Cartesian Product in our approach, we have generated two tables T2 and T2 (of increasing dimension: 500, 1000, 2500, 5000, 10000, 25000, and 50000 tuples

each) such that 10% of the tuples of T1 intersects as regards the possible chronons and, among them, 50% intersects also as regards certain chronons. In TSQL2, the two tables T1' (corresponding to T1) and T2' (corresponding to T2) have been implemented considering only two temporal attributes (to model the start and the end of the valid time). They contain the same tuples as T1 and T2 as regard the non-temporal attributes, and have as valid time the *possible* valid time of the corresponding tuples in T1 and T2.

Left part of Fig. 1 shows the execution times of Cartesian Product in our approach (blue and solid line) and in TSQL2 (red and dotted line), expressed in milliseconds. Right part of Fig. 1 shows the physical I/O (number of blocks). The answer set is not shown, since, obviously, it is the same for the two approaches. Notably, the I/O is slightly greater in our approach, due to the fact that our data model has two additional temporal attributes (to model both possible and certain time) with respect to TSQL2 (in which only determinate time is modeled). Such a fact also explains the slight overhead in execution time, which is also caused by the fact that the computation of intersection requires more operations in our approach (with respect to TSQL2).

As regard temporal difference, we have generated two tables T1 and T2 of increasing dimension (10000, 25000, 50000, 100000, 250000, 500000, 1000000 tuples) in such a way that 10% of the tuples in T1 have a value-equivalent counterpart (i.e., tuples with the same values for the non-temporal attributes) in T2. Among them, 20% temporally intersects as regards possible times, and 10% intersects also as regards certain time. As above, the tuples in the corresponding TSQL2 tables contain the same tuples as T1 and T2 as regard the non-temporal attributes, and have as valid time the *possible* valid time of the corresponding tuples in T1 and T2. Notably, the answer set of our approach is significantly greater than the one in TSQL2. This is a natural consequence of the different semantics of data. In TSQL2, only certain time can be considered, so that, e.g., the difference between a time interval t and a time interval t' containing t is obviously empty. On the other hand, in case t and t' are

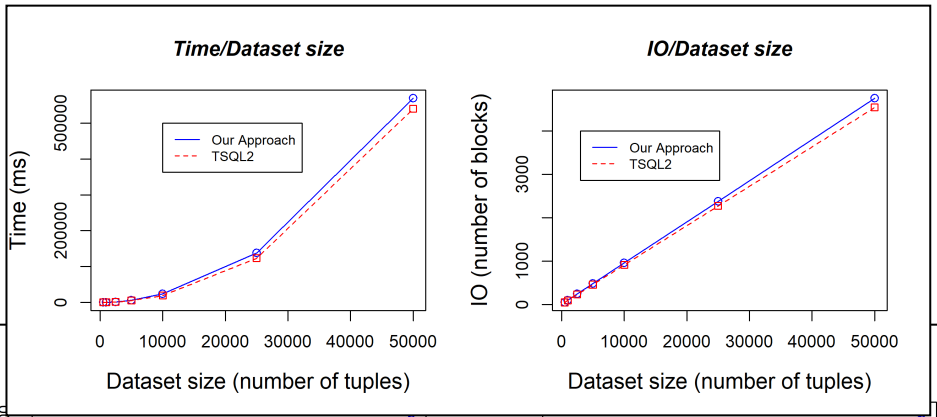

Figure 1. Cartesian Product. Comparisons between our approach and TSQL2 as regards execution time (on the left) and I/O (on the right), considering datasets of different dimensions.

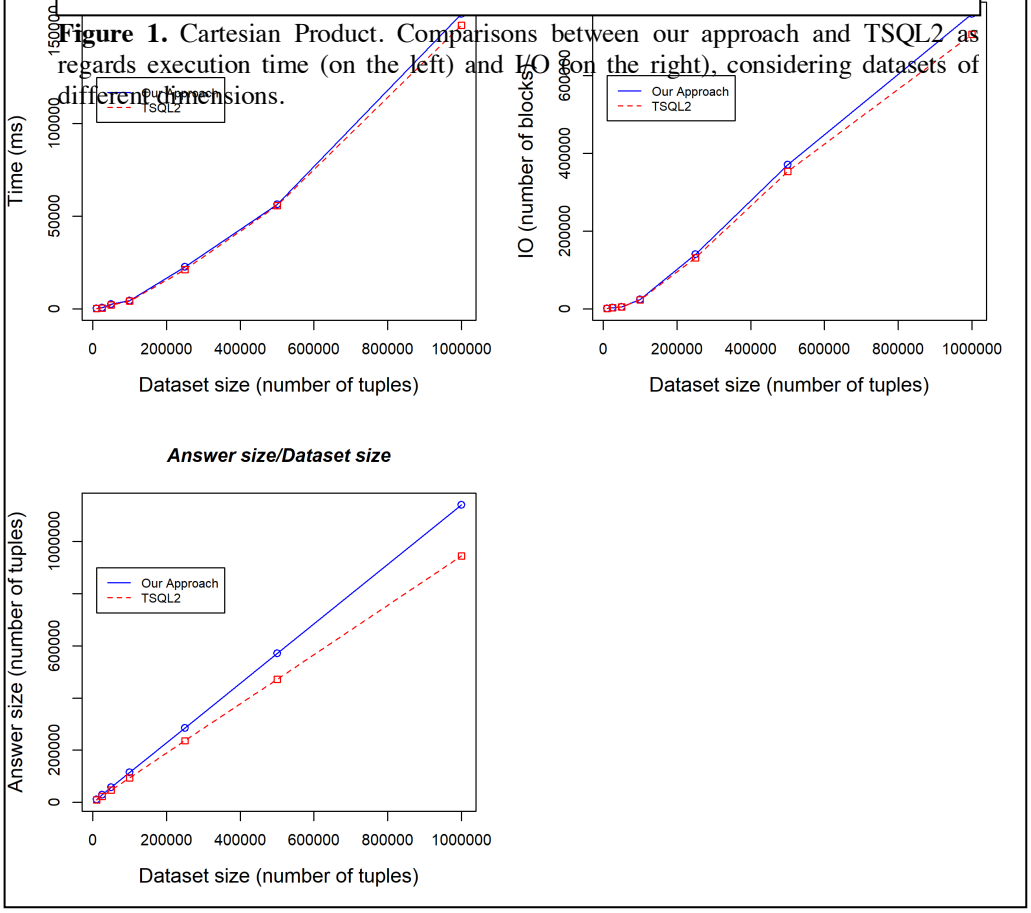

Figure 2. Difference. Comparisons between our approach and TSQL2 as regards execution time (top left figure), I/O (top right figure), and answer set (bottom left figure), considering datasets of different dimensions.

possible times, their difference is t. As an obvious consequence, the output in our approach can be larger. Given the larger answer set, and the additional memory needed to store four temporal attributes (instead than two, as in TSQL2), also the I/O (and the execution time) of our approach is higher than in TSQL2. Notably however, Fig. 2 clearly shows that the I/O and execution-time overhead of our approach is negligible.

To summarize, our experimental evaluations show that, while adding the possibility of modeling and querying indeterminate time, our AI-based approach only adds a negligible overhead to the "consensus" TSQL2 approach, in which only exact time is considered.

8 Conclusions and future work

In this paper, we propose an innovative AI approach in which a semantic-based AI-style methodology is applied to the context of TDBs in order to cope with temporal indeterminacy. Specifically:

(1) We propose a new "functional" semantic definition for indeterminate time in TDBs, which allows us to express the semantics of temporal algebraic operators in terms of their Codd's counterparts (thus formally providing a "*snapshot semantics*" for *indeterminate time* TDBs).
(2) We propose a new AI-style methodology to the treatment of TDBs, using it to develop a semantically-grounded 1NF approach (data model plus algebra) to cope with "interval-based" temporal indeterminacy.

Indeed, in this paper we have widely discussed the fact that, when introducing the temporal dimension, TDBs have to cope with ***implicit*** information, which has to be ***symbolically manipulated*** by algebraic operators to answer queries. As a consequence, we have proposed an innovative AI-based methodology to cope with time in relational DBs. We are confident that our methodology can be fruitfully applied to other types of temporal information in TDBs (e.g., "now-relative" data [Anselma et al., 16b], implicit representation of periodically repeated data [Terenziani, 16a, 16b]), and possibly of other forms of indeterminacy, thus leading to a new AI stream of research to cope with *indeterminate\implicit* data in relational DBs.

Appendix 1. BCDM Semantics

BCDM (Bitemporal Conceptual Data Model) [Snodgrass et al., 95] is a unifying data model, isolating the "core" semantics underlying many temporal relational approaches, including TSQL2. In BCDM, tuples are associated with *valid time* and *transaction time* [Ahn & Snodgrass, 86]. For both domains, a limited precision is

assumed (the *chronon* is the basic time unit). Both time domains are totally ordered and isomorphic to the subsets of the domain of natural numbers. The domain of valid times D_{VT} is given as a set $D_{VT}=\{c_1,\ldots,c_k\}$ of chronons, and the domain of transaction times D_{TT} is given as $D_{TT}=\{c'_1,\ldots,c'_j\}\cup\{UC\}$ (where *UC* –Until Changed– is a distinguished value). In general, the schema of a BCDM relation $R=(A_1,\ldots,A_n|T)$ consists of an arbitrary number of non-timestamp (*explicit* henceforth) attributes A_1, …, A_n, encoding some fact, and of a timestamp attribute T, with domain $D_{TT}\times D_{VT}$; the explicit attributes and the timestamp attribute are separated by the symbol "|". Thus, a tuple $x=(v_1,\ldots,v_n|t_b)$ in a BCDM relation $r(R)$ on the schema R consists of a number of attribute values associated with a set of bitemporal chronons $c_{bi}=(c'_h, c_i)$, with $c'_h\in D_{TT}$ and $c_i\in D_{VT}$, to denote that the fact $v_1,\ldots,v_n$ is current (present in the database) at time c'_h and valid at time c_i. An empty timestamp and *value-equivalent* tuples (i.e., tuples which are equal as regards the values of the non-temporal attributes [Snodgrass et al., 95]) are not admitted.

Valid-time, transaction-time and atemporal tuples are special cases, in which either the transaction time, or the valid time, or both of them are absent. In the following, we restrict our attention to valid time (in fact, temporal indeterminacy cannot affect transaction time). In BCDM, in which each tuple is paired with all the chonons when it holds. In BCDM, temporal databases directly associate valid times with tuples, so that the semantics of Example 1 above would be modeled as follows:

{<John, fever, high| {10,11,12)>, <Mary, fever, moderate| {11,12,13}>.

Codd designated as complete any query language that was as expressive as his set of five relational algebraic operators: relational union ($\cup$), relational difference (–), selection (σ_P), projection (π_X), and Cartesian product ($\times$) [Codd, 72]. BCDM generalizes these operators to cover bitemporal relations. BCDM dictates that, to support the "snapshot semantics", temporal operators must behave as standard non-temporal operators on the non-temporal attributes, and apply set operators on the temporal component of tuples. Cartesian product involves the intersection of the temporal components, projection and union involve their union, and difference the difference of temporal components.

Definition 14. BCDM temporal algebraic operators. Let r and s denote relations of the same sort and <v|t> a tuple with non-temporal part v and temporal part t.

$r \cup^T s = \{ < v|t > | < v|t > \in r \vee < v|t > \in s\}$

$r -^T s = \{ < v|t > | (< v|t > \in r \wedge \neg\exists t s\ (< v|t s > \in s\ ))\ \vee$
$\exists t_r\ \exists t_s\ (< v|t_r > \in r\ \wedge < v|t_s > \in s \wedge t = t_r - t_s\ \wedge t \neq \varnothing\ )\ \}$

$r \times^T s = \{ < v_r \cdot v_s|t > | \exists t_r\ \exists t_s\ (< v_r|t_r > \in r \wedge < v_s|t_s > \in s \wedge t = t_r \cap t_s \wedge t \neq \varnothing\ )\ \}$

$\pi^T_X\ (r) = \{ < v|t > | \exists v_r\ t_r\ (< v_r|\ t_r > \in r \wedge v = \pi_X\ (v_r))\ \}$

$\sigma^T_P\ (r) = \{ < v|t > | < v|t > \in r \wedge P(v)\ \}$ ■

This definition can be motivated by a sequenced snapshot semantics [Dunn et al., 02]: results should be valid independently at each point of time. Such a property is formally proved by proving the reducibility property.

To prove reducibility, they first introduce the timeslice operators. For the sake of simplicity, here we consider valid time relations only, so that valid timeslice operator is introduced.

Definition 15. Valid timeslice operator. Let r^T be valid time relation defined over the schema $(A_1, \ldots, A_n \mid T)$ and t a valid time. The result of the valid timeslice operator $\rho_t(r^T)$ is a standard nontemporal relation over the schema $(A_1, \ldots, A_n)$, defined as follows:
$\rho_t(r^T) = \{x \setminus \exists\, x' \in r^T \ (x[A_1, \ldots, A_n] = x'[A_1, \ldots, A_n] \wedge t \in x[T]\}$
(in the definition, $x[A_1, \ldots, A_n]$ represents the values of the attributes $A_1, \ldots, A_n$ in the tuple x, and x[T] its valid times). ■

The valid timeslice operator, given a valid time BCDM relation and a time t, removes the temporal part of the tuple and retains only the tuples whose valid time contained t.

Property 2. Reducibility of BCDM algebra to Codd's one. BCDM algebraic operators are reducible to Codd operators, i.e., for BCDM operator Op^T that extends a Codd's operator Op to cope with time, and for each time t, the following holds (the analogous holds for unary operators):
$\rho_t{}^T(r^T \ Op^T \ s^T) = \rho_t{}^T(r^T) \ Op \ \rho_t{}^T(s^T)$.
(see the proof in [Snodgrass et al., 95]) ■

References

[Allen, 91] J. F. Allen, Time and time again: The many ways to represent time. Int. J. Intell. Syst., 6, pp. 341–355, (1991).

[Anselma et al., 13] Anselma, L., Terenziani, P., Snodgrass, R.T.: Valid-Time Indeterminacy in Temporal Relational Databases: Semantics and Representations. IEEE Transactions on Knowledge and Data Engineering. 25, 2880–2894 (2013).

[Anselma et al., 16] Anselma, L., Piovesan, L., Terenziani, P.: A 1NF temporal relational model and algebra coping with valid-time temporal indeterminacy. Journal of Intelligent Information Systems. 47, 345–374 (2016).

[Anselma et al., 16b] Anselma, L., Piovesan, L., Sattar, A, Stantic B., Terenziani, P.: A Comprehensive Approach to 'Now' in Temporal Relational Databases: Semantics and Representation. IEEE Trans. Knowl. Data Eng. 28(10): 2538-2551 (2016)

[Chomicki & Toman, 98] J. Chomicki and D. Toman: Temporal Logic in Information Systems, in Logics for Databases and Information Systems, Chomicki and Saake (eds.), Kluwer, 31-70, (1998).

[Codd, 72] E. F. Codd, Relational Completeness of Data Base Sublanguages. In: R. Rustin (ed.), Database Systems 65-98, Prentice Hall and IBM Research Report RJ 987, San Jose, California, (1972).

[Dekhtyar et al., 01] Dekhtyar, A., Ross, R., Subrahmanian, V.: Probabilistic temporal databases, I: algebra. ACM Transactions on Database Systems (TODS). 26, 41–95 (2001).

[Dyreson & Snodgrass, 98] Dyreson, C.E., Snodgrass, R.T.: Supporting valid-time indeterminacy. ACM Transactions on Database Systems (TODS). 23, 1–57 (1998).

[Dyreson, 09] Dyreson, C.: Temporal Indeterminacy. In: Liu, L. and Ozsu, M.T. (eds.)

Encyclopedia of Database Systems. pp. 2973–2976. Springer US, Boston, MA (2009).

[Dunn et al., 02] J. Dunn, S. Davey, A. Descour, and R.T. Snodgrass, Sequenced Subset Operators: Definition and Implementation, proc. ICDE'02, (2002).

[Dutta, 89] Dutta S: Generalized events in temporal databases. In: Data Engineering, Proceedings. ICDE 1989, pp 118-125,(1989).

[Elmasri et al., 03] Elmasri, Ramez; Navathe, Shamkant B. (July 2003). Fundamentals of Database Systems, Fourth Edition. Pearson.

[Emerson, 90] Emerson AE. Temporal and Modal Logic, In: Van Leeuwen (ed) Handbook of Theoretical Computer Science, Vol. B, pp. 997-1072, Elsevier Science Publishers, (1990).

[Gadia et al., 92] Gadia SK, Nair SS, Poon YC: Incomplete information in relational temporal databases. In: Yuan LY (ed) VLDB, Morgan Kaufmann, pp 395-406, (1992).

[Jensen & Snodgrass, 96] Jensen, C.S., Snodgrass, R.T.: Semantics of Time-Varying Information. INFORMATION SYSTEMS. 21, 311–352 (1996).

[Liu & Özsu, 2009] Liu, L., Özsu, M.T. eds: Encyclopedia of database systems. Springer (2009).

[Mc Kenzie & Snodgrass, 91] McKenzie, L.E., Jr., Snodgrass, R.T.: Evaluation of Relational Algebras Incorporating the Time Dimension in Databases. ACM Comput. Surv. 23, 501–543 (1991).

[Minker, 82] Minker, Jack*:* On indefinite databases and the closed world assumption*, Lecture Notes in Computer Science, 138,* Springer Berlin Heidelberg*, pp. 292–308, (1982).*

[Snodgrass, 82] Snodgrass RT: Monitoring distributed systems: A relational approach. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh,PA, (1982).

[Snodgrass, 00] Snodgrass, R.: Developing Time-oriented Database Applications in SQL. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2000).

[Snodgrass & Ahn 86] R. T. Snodgrass and I. Ahn, "Temporal Databases," IEEE Computer 19(9): 35–42, September, 1986.

[Snodgrass et al., 95] Snodgrass RT (ed.), Ahn I, Ariav G, Batory D, Clifford J, Dyreson C, Elmasri R, Grandi F, Jensen C, Kafer W, Kline N, Kulkarni K, Leung C, Lorentzos N, Roddick J, Segev A, Soo M, Sripada S.M (1995) The temporal query language TSQL2, Kluwer Academic Publishers, Boston (1995).

[Terenziani, 16a] Terenziani P.: Irregular Indeterminate Repeated Facts in Temporal Relational Databases. IEEE Trans. Knowl. Data Eng. 28(4): 1075-1079 (2016)

[Terenziani, 16b] Terenziani P.: Nearly Periodic Facts in Temporal Relational Databases. IEEE Trans. Knowl. Data Eng. 28(10): 2822-2826 (2016).

[Vila, 94] Vila L. A survey on temporal reasoning in artificial intelligence, AI Communications 7(1): 4-28 (1994).

[Wu et al., 98] Wu, Y., Jajodia, S., Wang, X.S.: Temporal database bibliography update. In: Etzion, O., Jajodia, S., and Sripada, S. (eds.) Temporal Databases: Research and Practice. pp. 338–366. Springer Berlin Heidelberg, Berlin, Heidelberg (1998).