Representation and Querying in Temporal Databases: the Power of Temporal Constraints*

Manolis Koubarakis
Computer Science Division
Dept. of Electrical and Computer Engineering
National Technical University of Athens
Zographou 157 73
Athens GREECE
koubarak@theseas.ntua.gr

Abstract

We propose a temporal database model capable of representing absolute, relative, imprecise and infinite temporal data. Our model is based on temporal tables i.e., relation-like representations which can contain variables constrained by the formulas of a temporal theory. We define an algebraic query language for temporal tables and discuss some problems related to query answering.

1 Introduction

Research in temporal databases (see [16] for a survey) has mostly concentrated on temporal models with absolute temporal information (e.g., "the salary of Jones was $20,000 from January 1989 to April 1989"). There are very few exceptions to this general rule. [2] presents a model which allows relative temporal information (e.g., "event A happened before event B and after January 23, 1992"). [5] allows two kinds of indefinite information: (i) the existence of a tuple in a relation can be uncertain over some interval of time, and (ii) attribute values can be unknown during the period of existence of a tuple in a relation. There are also some other proposals (notably [8]) which deal successfully with (finite or infinite) periodic data (e.g., "the commitee meets every Monday").

On the other hand, the artificial intelligence (AI) community has investigated many temporal logics and systems which can deal successfully with absolute, relative, imprecise or default temporal information [13, 12, 4, 14]. In this paper, we make a first attempt to integrate these two research programs. We propose a temporal database model which offers some of the representational capabilities found in AI formalisms. This model is based on *temporal tables* which are relation-like representations capable of representing absolute, relative, imprecise or infinite temporal data. The model of temporal tables is a hybrid of the *generalized relations* of [9, 8], and the *conditional tables* of [7, 6]. To the best of our knowledge, this is the first proposal for a temporal database model with this expressive power. We also define an algebraic query language and discuss some problems related to query answering.

The organization of this paper is as follows. In the following section, we introduce the model of temporal tables through a simple but detailed example. In sections 3 and 4, we define temporal tables and their query language formally. In section 5, we discuss some problems related to query answering in our model. Finally, we discuss related work and future research.

2 Representing Temporal Information in the Model of Temporal Tables

In this section, our model is introduced. We start with an example which is intended to motivate the subsequent theoretical development. Let us assume that we want to design a database for a big transportation company. This database will be used by the scheduling personnel to keep track of orders, vehicles, loads, trips and deliveries. Two of the temporal tables which can possibly be part of this database are ON_SERVICE and TRIP:

*Appears in: Proceedings of the 9th International Conference on Data Engineering, Vienna, April 1993.

ON_SERVICE

VEH	CITY	TIME	CON
V1	Athens	t1	$\omega 1_{\rm L} \leq {\rm t1_L},\ {\rm t1_R} \leq \omega 1_{\rm R}$
V2	Piraeus	t2	$\omega 2_{\rm L} \leq {\rm t2_L},\ {\rm t2_R} \leq \omega 2_{\rm R}$

TRIP

VEH	FROM	TO	TIME	CON
V3	Athens	Patra	$\langle 10.5, 14\rangle$	true

The constraint ${\rm G}:\ \omega 1_{\rm L} = 9,\ 14 \leq \omega 1_{\rm R} \leq 16,\ 10 \leq \omega 2_{\rm L} \leq 10.5,\ 2 \leq \omega 2_{\rm R} - \omega 2_{\rm L} \leq 4$ holds for variables $\omega 1$ and $\omega 2$ which denote time intervals whose duration is not known precisely.

The model of *temporal tables* is a hybrid of the conditional tables of [7, 6], the generalized relations of [9] and the generalized temporal relations of [8].[1] Temporal tables are beasts similar to conditional tables but they can also have *e-variables* or *u-variables* as attribute values.[2] E-variables (e.g., $\omega 1$ and $\omega 2$ in the above tables) are semantically equivalent to the variables of conditional tables. They represent values which are not completely known but for which a constraint G (called the *global temporal constraint*) holds. U-variables are like the variables of generalized relations: they can be thought of as being universally quantified. They are local to some tuple and can be constrained by the tuple's *local temporal constraint*.

Temporal constraints are conjunctions of special kinds of formulas in a constraint language $\mathcal{L}$ which is defined in section 3.1. They can be used to express qualitative or quantitative properties of points and intervals of time. For example, in the first tuple of table ON_SERVICE the variable t1 represents the interval during which vehicle V1 is in Athens for service. The local constraint of this tuple constrains interval $\rm t_1$ to be during interval ω_1. This containment relation is expressed by order constraints on the endpoints of the intervals $\rm t_1$ and ω_1 (for any interval i, the left and right endpoints of i are denoted by the terms i_L and i_R respectively). In a similar manner, the global constraint G states that (i) ω_1 starts at time 9 and ends at some time between 14 and 16, and (ii) ω_2 starts at some time between 10 and 10.5 and its duration is 2 to 4 time units.

Temporal tables are then syntactic devices for the representation of our possibly imprecise (finite or infinite) temporal knowledge. Each table is semantically equivalent to a set of possible worlds which capture all the possible ways the domain of application could have been according to our imprecise knowledge. Each possible world can be represented by a *u-table* i.e., a table without e-variables (a generalized relation in the terminology of [8]). One can find all the possible worlds represented by a table as follows:

- Find an assignment to the e-variables such that the global constraint becomes true.
- Substitute these values for the e-variables in the table.

For example, the u-table

POSS_ON_SERVICE

VEH	CITY	TIME	CON
V1	Athens	t1	$9 \leq {\rm t1_L},\ {\rm t1_R} \leq 15$
V2	Piraeus	t2	$10 \leq {\rm t2_L},\ {\rm t2_R} \leq 14$

represents one of the possible worlds corresponding to table ON_SERVICE. A tuple of a u-table represents a possibly infinite set of ground tuples. For example, the meaning of the first tuple in the above u-table is that vehicle V1 is in Athens for service for interval $\langle 9, 15\rangle$ and all of its subintervals. This is a simple kind of infinite temporal information.[3] U-variables like t1 are essentially universally quantified: for each value of t1 which satisfies the local constraint of the tuple, a ground tuple is included in the meaning of the u-table. This is an example of how *properties* of objects (in the terminology of [1]) can be modeled in our framework. The distinguishing feature of a property (e.g., "being on service") is that it is true over an interval iff it true over every subinterval of this interval.

Let us now look at the simpler table TRIP. This table does not contain variables thus it represents our *complete* and *finite* knowledge about some part of the world. The first tuple of table TRIP is much like a tuple in a standard relation. It states that the trip of vehicle V3 from Athens to Patra occured over the interval $\langle 10.5, 14\rangle$ (without saying anything about the subintervals of that interval). This is an example of an *event* in the terminology of [1]. What distinguishes events from properties is that the former cannot occur over two intervals one of which contains another.[4]

We now consider the query "Find the vehicles that are on service throughout the first hour of the afternoon". Using the table algebra defined in section 4, this query can be answered by evaluating the expression $\pi_{\rm VEH}(\sigma_{{\rm TIME_L} \leq 12\ \wedge\ 13 \leq {\rm TIME_R}}({\rm ON\_SERVICE}))$.

[1] However we do not deal with periodic data in this paper. See [11].

[2] In the examples of this paper, e-variables will be denoted by ω (possibly with a subscript) while u-variables will be denoted by t (possibly with a subscript).

[3] Infinitely periodic data (possibly impecise) is another important case of infinite temporal information arising in some applications. See [11] for examples.

[4] For a richer categorization of temporal propositions see [15].

Let us examine this expression carefully. The selection operation selects all tuples of table ON_SERVICE whose attribute TIME contains interval $\langle 12, 13\rangle$.[5] The following table is the result of this selection operation.

VEH	STATION	TIME	CON
V1	Athens	t1	$\omega 1_{\mathrm{L}} \leq \mathrm{t1}_{\mathrm{L}},\ \mathrm{t1}_{\mathrm{R}} \leq \omega 1_{\mathrm{R}},$ $\mathrm{t1}_{\mathrm{L}} \leq 12,\ 13 \leq \mathrm{t1}_{\mathrm{R}}$
V2	Piraeus	t2	$\omega 2_{\mathrm{L}} \leq \mathrm{t2}_{\mathrm{L}},\ \mathrm{t2}_{\mathrm{R}} \leq \omega 2_{\mathrm{R}},$ $\mathrm{t2}_{\mathrm{L}} \leq 12,\ 13 \leq \mathrm{t2}_{\mathrm{R}}$

The difference with the selection operation of relational algebra is easy to understand. Since temporal attribute values can be variables, we no longer know for certain whether the selection condition is true or false. Thus we simply append the selection condition to the local constraint of each tuple of the input table. The correctness of this procedure is easily verified: a tuple is in the answer table iff its new local constraint is true.

To execute the projection operation, we need to eliminate the variables t1 and t2 from the local constraints. If we treat terms $\mathrm{t1}_{\mathrm{L}}$ and $\mathrm{t1}_{\mathrm{R}}$ as they were rational variables and we keep in mind the implicit relation $\mathrm{t1}_{\mathrm{L}} < \mathrm{t1}_{\mathrm{R}}$ then we can eliminate $\mathrm{t1}_{\mathrm{L}}$ and $\mathrm{t1}_{\mathrm{R}}$ and obtain $\omega 1_{\mathrm{L}} \leq 12,\ 13 \leq \omega 1_{\mathrm{R}}$. Similarly, $\omega 2_{\mathrm{L}} \leq 12,\ 13 \leq \omega 2_{\mathrm{R}}$ is obtained from the second tuple. Therefore, the answer to the query is the following table ANS1 together with the global constraint G.

ANS1

VEH	CON
V1	$\omega 1_{\mathrm{L}} \leq 12,\ 13 \leq \omega 1_{\mathrm{R}}$
V2	$\omega 2_{\mathrm{L}} \leq 12,\ 13 \leq \omega 2_{\mathrm{R}}$

Since the global constraint G captures the relationships between the e-variables of all tables, it should not be modified by the evaluation of a query.

The set of possible worlds represented by ANS1 is given by the set of relations $\left\{ \left| \mathrm{V1} \right|, \left| \begin{array}{c} \mathrm{V1} \\ \mathrm{V2} \end{array} \right| \right\}$. Each one of these relations represents one set of vehicles that are possibly on service throughout the first hour of the afternoon.

Let us now note that vehicle V1 is always in the answer set while this is not the case for V2. Therefore for the case of temporal tables answering a query should involve more than providing an answer table and a global constraint on its e-variables. If this table is quite complex, it might not be easy for a user to detect whether a ground tuple is *possibly* or *certainly* in the answer. Thus the system must be ready to inform the user whether a ground tuple (or a set of ground tuples) is possibly or certainly in the answer. Moreover, it must be capable of providing the user with a finite description of the (possibly infinite) set of ground tuples that are possibly or certainly in the answer. For example, the set of tuples possibly in ANS1 is $possible(\mathrm{ANS1}) = \{(\mathrm{V1}), (\mathrm{V2})\}$ while the set of tuples certainly in ANS1 is $certain(\mathrm{ANS1}) = \{(\mathrm{V1})\}$.

Let us also consider query $\pi_{\mathrm{VEH,TIME}}(\sigma_{\mathrm{VEH=V1}}(\mathrm{ON\_SERVICE}))$ (i.e., "Find all times during which vehicle V1 is on service; output only vehicle identification and time"). The answer to this query is given by table ANS2:

ANS2

VEH	TIME	CON
V1	t1	$\omega 1_{\mathrm{L}} \leq \mathrm{t1}_{\mathrm{L}},\ \mathrm{t1}_{\mathrm{R}} \leq \omega 1_{\mathrm{R}}$

In this case, the set of tuples possibly in ANS2 is *infinite*. The following u-table *possible*(ANS2) is a *finite representation* of this infinite set. The set of tuples which are certainly in ANS2 is empty.

possible(ANS2)

VEH	TIME	CON
V1	t1	$9 \leq \mathrm{t1}_{\mathrm{L}},\ 14 \leq \mathrm{t1}_{\mathrm{R}} \leq 16$

Section 5 discusses algorithms for discovering whether a tuple is possibly or certainly in the answer; the longer version of this paper [11] discusses how to compute finite representations of the (possibly infinite) sets of these tuples.

3 The Model of Temporal Tables

The previous discussion must have given the reader the flavour of our model. Now we turn to its formal definition. At first, we present a language for expressing temporal constraints. Then we give the formal definition of the model of temporal tables.

3.1 Temporal Constraints

In our model, time is assumed to be linear, dense and unbounded. Time entities include points and intervals. Points are identified with the rationals while intervals are considered as pairs of points.

The language $\mathcal{L}$ for expressing temporal constraints is a many-sorted first order calculus with equality. It has two sorts: *Rational* and *Interval*. The logical

[5] In other words, the left endpoint of TIME is less than or equal to 12 and the right endpoint of TIME is greater than or equal to 13.

symbols of $\mathcal{L}$ include parentheses, a countably infinite set of variables V_s for each sort s, an equality symbol $=_s$ for each sort s and the standard sentential connectives. The non-logical symbols of $\mathcal{L}$ include a countably infinite set of function symbols of arity 0 and sort *Rational* (the rational numerals), function symbols L and R of arity 1 with domain *Interval* and range *Rational* (which denotes the left and right endpoint of an interval), function symbol $-$ of arity 2 with domain $Rational \times Rational$ and range *Rational* (which denotes subtraction over the rationals), function symbol $\langle\cdot,\cdot\rangle$ of arity 2 with domain $Rational \times Rational$ and range *Interval* (which denotes an interval built from two endpoints) and predicate symbols $<$ of arity 2 with sort $Rational \times Rational$ (which denotes the relation "less than" over the rationals.)

The symbols of $\mathcal{L}$ are interpreted with respect to the many-sorted structure $\mathcal{T}$ which encompasses our assumptions for time. $\mathcal{T}$ is defined formally in [11]. Variable assignments, satisfaction and truth in $\mathcal{T}$ are defined in the usual way.

The *simple terms* of $\mathcal{L}$ are variables of sort *Rational* and terms t_L and t_R where t is a variable of sort *Interval*. Simple terms are used in definining temporal constraints.[6] *Temporal constraints* are formulas of any of the following forms:

- *Inequalities*: $t_1 - t_2 \leq r$ or $t_1 - t_2 < r$ where t_1, t_2 are simple terms and r is a constant of sort *Rational* (t_1 or t_2 can be omitted).[7]

- *Disjunctions of inequations*: $t_1 - t'_1 \neq r_1 \ \vee\ t_2 - t'_2 \neq r_2 \ \vee\ \cdots\ \vee\ t_n - t'_n \neq r_n$ where $t_1, t_2, \cdots, t_n, t'_1, t'_2, \cdots, t'_n$ are simple terms and $r_1, r_2, \cdots, r_n$ are constants of sort *Rational* (t_i or t'_i can be ommited).[8]

- the special formulas *true* and *false*.

The set of temporal constraints in $\mathcal{L}$ will be denoted by $TC_{\mathcal{L}}$. For a set (i.e., a conjunction) C of temporal constraints in variables $x_1, \ldots, x_n$ of sorts $S_1, S_2, \ldots, S_n$ the *solution set* of C is defined as

$$Sol(C) = \{(\tau_1, \ldots, \tau_n) : \ \tau_i \in \mathcal{I} \text{ if } S_i = Interval, \\ \tau_i \in \mathsf{Q} \text{ if } S_i = Rational \text{ and for each constraint} \\ c \in C,\ \mathcal{T} \models c[x_1 \leftarrow \tau_1, \ldots, x_n \leftarrow \tau_n]\}.$$

[6] They will be used to dissalow unecessarily complex terms of $\mathcal{L}$ e.g., $\langle p_1, \langle p_2, p_3\rangle_R\rangle_R \equiv p_3$.

[7] For conjunctions of inequalities, we will usually write $\alpha \leq \beta \leq \gamma$ instead of $\alpha \leq \beta \ \wedge\ \beta \leq \gamma$.

[8] [10] explains why this type of constraints are interesting and discusses some of their properties.

Each member of $Sol(C)$ is called a *solution* of C. A set of temporal constraints is called *consistent* or *satisfiable* iff $Sol(C) \neq \emptyset$. Two sets of constraints are called *equivalent* iff they have the same solution set. A set of constraints is the algebraic counterpart of the logical conjunction of its members. Thus, in the following sections, we will frequently mix the terms "set of constraints" and "conjunction of constraints".

Having defined temporal constraints, we now turn to the definition of temporal tables.

3.2 Temporal Tables

Let us assume the existence of three countably infinite sets: the set of *interval attributes* U_I, the set of *point attributes* U_P and the set of *data attributes* U_D. Elements of $U_I \cup U_P$ are also called *temporal attributes.* The set of attributes U is defined as $U = U_I \cup U_P \cup U_D$. For each $A \in U_D$, we define a countably infinite set $dom(A)$, the *domain* of A. Similarly, for each $A \in U_I$, we define $dom(A) = \mathcal{I}$ and for each $A \in U_P$ we define $dom(A) = \mathsf{Q}$. For a set of attributes $S = \{A_1, ..., A_n\}$, we define $dom(S) = dom(A_1) \cup \cdots \cup dom(A_n)$.

We also assume the existence of four disjoint countably infinite sets of *variables* whose intersection with the domains of attributes is empty. These are the sets of *rational e-variables* V_e^p, *interval e-variables* V_e^i, *rational u-variables* V_u^p and *interval u-variables* V_u^i. We assume that V_e^p and V_u^p are subsets of $V_{Rational}$ and V_e^i and V_u^i are subsets of $V_{Interval}$. Let us define $V_e = V_e^p \cup V_e^i$ and $V_u = V_u^p \cup V_u^i$.

Temporal schemes, tuples and tables. A *temporal scheme* (or simply *scheme*) is a set of data and temporal attributes. Let $S = \{A_1, ..., A_n, T_1, ..., T_m\}$ be a scheme ($A_i \in U_D,\ i = 1, ..., n$ and $T_j \in U_I \cup U_P,\ j = 1, ..., m$) and *Con* be the set of all conjunctions of temporal constraints in the language $\mathcal{L}$ with variables from $V^e \cup V^u$. A *temporal tuple* (or simply *tuple*) t on scheme S is a mapping from $S \cup \{CON\}$ to $dom(S) \cup V_u \cup Con$ such that (i) $t(A_i) \in dom(A_i)$ for $i = 1, \ldots, n$, (ii) $t(T_j) \in V_u^p$ if $T_j \in U_P$ and $t(T_j) \in V_u^i$ if $T_j \in U_I$ (for $j = 1, \ldots, m$), and (iii) $t(CON) \in Con$. A tuple which does not contain variables will be called *ground.* The value $t(CON)$ is the *local constraint* of the tuple t while $t(S)$ is the *proper part* of t. The proper part of a tuple t is further divided into the *data part* $t(A_1 \cdots A_n)$ and the temporal part $t(T_1 \cdots T_m)$. A *temporal table* (or simply *table*) T on scheme S is a finite set of temporal tuples on S together with a constraint $G(T) \in Con$ involving only variables from V_e (this is called the *global constraint*). A *u-table* is a temporal table which does not contain variables from

V_e.[9] The *type* of a tuple t (or table T) is the scheme over which the tuple (or table) has been defined; it is denoted by $\alpha(t)$ (or $\alpha(T)$). The set of all tables on scheme S is denoted by $Tab(S)$. The set of all tables (on any scheme) is denoted by Tab.

A *table name* is a symbol A (possibly subscripted) with an associated scheme or type $\alpha(A) \subset U$. An *instance* of A is a table T such that $\alpha(T) = \alpha(A)$.

Valuations. We now define the concept of a valuation (a special kind of variable assignment). An *e-valuation* (valuation of e-variables) is a function $v : V_e \cup V_u \rightarrow \mathsf{Q} \cup \mathcal{I}$ such that $v(a) = a$ if $a \in V_u$, $v(a) \in \mathsf{Q}$ if $a \in V_e^p$ and $v(a) \in \mathcal{I}$ if $a \in V_e^i$. Similarly, we define an u-valuation (valuation of u-variables) as a function $v : V_u \rightarrow \mathsf{Q} \cup \mathcal{I}$ such that $v(a) \in \mathsf{Q}$ if $a \in V_u^p$ and $v(a) \in \mathcal{I}$ if $a \in V_u^i$. We will denote by Val^e and Val^u the set of e-valuations and u-valuations respectively. The *concatenation* of a u-valuation u and a e-valuation v is the function $u; v : V_e \cup V_u \rightarrow \mathsf{Q} \cup \mathcal{I}$ such that $(u; v)(a) = u(a)$ if $a \in V_u$ and $(u; v)(a) = v(a)$ if $a \in V_e$.

The result of applying a u-valuation to the proper part of a tuple can be defined as follows. If t is a tuple on scheme S and u is a u-valuation then $u(t)$ is a ground tuple such that for each $A \in S$, $u(t)(A) = u(t(A))$ if $t(A) \in V_u$ and $u(t)(A) = t(A)$ otherwise.

Semantics. The semantics of a table T on scheme S is captured by the set of relations

$$sem(T) = \{r_v(T) : \ v \in Val^e \text{ and } \mathcal{T} \models G(T)[v]\}$$

where $r_v(T) = \{u(t(S)) \ : \ t \text{ is a tuple of } T, \ u \in Val^u \text{ and } \mathcal{T} \models t(CON)[u; v]\}$.

Note that $sem(T) = \emptyset$ iff $G(T)$ is inconsistent. If $G(T)$ is consistent but for each valuation $v \in Val^e$ such that $\mathcal{T} \models G(T)[v]$ there is no valuation $u \in Val^u$ such that $\mathcal{T} \models t(CON)[u; v]$ then $sem(T) = \{\emptyset\}$.

In a similar way, we can define multishemes (i.e., sequences of schemes), multitables (i.e., sequences of tables) and their semantics. We will freely use these notions in what follows; the reader can consult [11] for detailed definitions.

[9] In the previous definitions, observe that constants and e-variables are not allowed as values of temporal attributes (as in table TRIP of the previous example). This is not a real restriction of our model; it only makes some of the following definitions simpler. Whenever a temporal table which does not obey this rule is encountered, we can easily transform it to an equivalent table which respects this rule in the following way. Any tuple of a table T such that $t(T_i) = c$ and $t_i(CON) = \phi$ (where T_i is a temporal attribute of T, c a constant or an e-variable of sort *Rational* and $\phi \in Con$) is equivalent to a new tuple t' which is the same as t except that $t'(T_i) = x$ and $t'(CON) = \phi \ \wedge \ x = c$ (where x is a brand new variable from the set V_u^p). Similar comments apply to the case of an attribute value of sort *Interval*.

4 An Algebraic Query Language for Temporal Tables

In this section, a *temporal table algebra* is defined. We only give definitions for selection, projection, join and difference. [11] gives definitions for cartesian product, union, rename, PointsToIntervals and IntervalsToPoints. Most of the operations are extensions of similar relational algebra operations; the last two operations are used for going from points to intervals and vice versa. Every operation takes as input one (or two) tables with a common global constraint and returns a table and the same global constraint.

Selection. Without loss of generality, the *selection* operation σ_E for tables is defined only for conjunctive selection conditions E. Selection conditions are defined as follows:

1. A temporal constraint whose variables are attributes is a temporal selection condition.
2. A formula of the form $A = a, A \neq a, A = B$ or $A \neq B$ is a data selection condition.
3. If E_1 and E_2 are selection conditions (data or temporal) then $E_1 \ \wedge \ E_2$ is a selection condition.

If $E = E_t \ \wedge \ E_d$ where E_t is a conjunction of temporal selection conditions and E_d is a conjunction of data selection conditions then $\sigma_E(T)$ is defined as $\sigma_{E_t}(\sigma_{E_d}(T))$. For a data selection condition E, σ_E is defined as in the relational case [17]. On the other hand if T is a table on scheme S and E is a temporal selection condition then $\sigma_E(T)$ is defined as follows:

$$\sigma_E(T) = \{\sigma_E(t) : \ t \in T\}$$

where $\sigma_E(t)(S) = t(S)$ and $\sigma_E(t)(CON) = t(CON) \wedge E(t)$. The constraint $E(t)$ is obtained from E by substituting $t(A)$ for each temporal attribute A that appears in E.

Projection. If T is a table on scheme S and $X \subseteq S$ then $\pi_X(T)$ is defined as follows:

$$\pi_X(T) = \{\pi_X(t) : \ t \in T\}$$

where $\pi_X(t)$ is a tuple with scheme X and $\pi_X(t)(X) = t(X)$ and $\pi_X(t)(CON)$ is obtained from $t(CON)$ after all the variables $u \in V^u$ such that there exists $A \in R - X$ and $t(A) = u$ are *eliminated*. If $X = \emptyset$ then for any tuple t, $t(X)$ is the empty tuple ().[10]

[10] The *elimination of variables* can be achieved in the following way. Given a set of temporal constraints C and a set of variables V to be eliminated, C is first translated into an "equivalent" set of temporal constraints C' involving only variables and

Join. Let us assume that S_1 and S_2 are the schemes of tables T_1 and T_2, X_t is the set of temporal attributes common to S_1 and S_2 and X_d is the set of data attributes common to S_1 and S_2. The join of T_1 and T_2 can now be defined as follows:

$$T_1 \bowtie T_2 = \{t_1 \bowtie t_2 : \; t_1 \in T_1, \; t_2 \in T_2 \text{ and } t_1(X_d) = t_2(X_d)\}$$

The join of two tuples $t_1 \bowtie t_2$ is defined by the following:

1. $(t_1 \bowtie t_2)(A) = t_1(A)$ if $A \in S_1$
2. $(t_1 \bowtie t_2)(A) = t_2(A)$ if $A \in S_2 - (X_t \cup X_d)$
3. $(t_1 \bowtie t_2)(CON) = t_1(CON) \;\wedge\; t_2'(CON)$ where $t_2'(CON)$ is the same as $t_2(CON)$ but with each variable $t_2(A)$ substituted by $t_1(A)$ for each $A \in X_t$.

Difference. Let us take tables T and W with scheme $S = S_t \cup S_d$. The difference of T and W can now be defined as $T - W = \bigcup_{t \in T} (t_W)^+$ where $W = \{t_1, \cdots, t_n\}$ and $t_W = (((t - t_1) - t_2) - \cdots - t_n)$. The difference of two temporal tuples $t_1 - t_2$ is a temporal *table* which is defined as follows:

1. $t_1 - t_2 = \{t_1\}$ if $t_2(S_d) \neq t_1(S_d)$.
2. $t_1 - t_2 = \{s : s(S_d) = t_1(S_d)$ and $s(S_t \cup \{CON\}) = \overline{t_2(S_t \cup \{CON\})} \cap t_1(S_t \cup \{CON\}) \}$ otherwise.

The last formula needs some explanation. The tuple $\overline{t}$ is the complement of t. It is computed by negating the constraints and leaving the rest of the tuple the same. We will abuse our terminology and call $\overline{t}$ a tuple although its local constraint is a conjunction of temporal constraints and negated conjunctions of temporal constraints. The intersection of two tuples $t_1 \cap t_2$ is defined by $(t_1 \cap t_2)(S_t) = t_1(S_t)$ and $(t_1 \cap t_2)(CON) = t_1(CON) \;\wedge\; t_2'(CON)$, where $t_2'(CON)$ is the same as $t_2(CON)$ but with each variable $t_2(A)$ substituted by $t_1(A)$ for each $A \in S_t$. Finally, the operation T^+ has the effect of *normalizing* the local constraints of table T in order to obtain a true temporal table. This is done by the following three steps:

- Application of De Morgan's laws to transform the negated part of each local constraint of T (which was produced by the complementation operation) into a disjunction whose disjuncts are inequality constraints or conjunctions of equality constraints.
- Application of the law of associativity of conjunction with respect to disjunction to transform each local constraint of T into a disjunction of temporal constraints.
- Splitting of disjunctive constraints into different tuples with the same temporal part.

Now we present a query which uses some of the operations defined above. More examples can be found in [11].

Example 4.1 *Find all times during which both vehicles* V1 *and* V2 *are on service.*

This query can be answered by joining $\pi_{\mathrm{TIME}}(\sigma_{\mathrm{VEH=V1}}(\text{ON\_SERVICE}))$ and $\pi_{\mathrm{TIME}}(\sigma_{\mathrm{VEH=V2}}(\text{ON\_SERVICE}))$. The answer is table ANS3.

ANS3

TIME	CON
t1	$\omega 1_{\mathrm{L}} \leq \mathrm{t1}_{\mathrm{L}},\; \mathrm{t1}_{\mathrm{R}} \leq \omega 1_{\mathrm{R}},$ $\omega 2_{\mathrm{L}} \leq \mathrm{t1}_{\mathrm{L}},\; \mathrm{t1}_{\mathrm{R}} \leq \omega 2_{\mathrm{R}}$

From ANS3 and the global constraint, we can see that the tuples $(\langle 10, 12\rangle)$, $(\langle 13, 14\rangle)$ etc. are in *possible*(ANS3) and $(\langle 10.5, 12\rangle)$, $(\langle 11, 12\rangle)$ etc. are in *certain*(ANS3).

5 The Possibility and Certainty Problems

As we have already discussed in section 2, it is not enough to simply return a table and a global constraint as an answer to a user query. Since the structure of a table can be quite complex, a user might need to know whether it is *possible* or *certain* that a ground tuple (or a set of ground tuples) is in the answer set. In addition, he or she might want to obtain a finite representation of all tuples that are possibly or certainly in the answer set. This motivates the following definitions [6].

constants of sort *Rational* (let us call these constraints *rational constraints*). This translation can be achieved by (i) substituting terms t_L and t_R by brand new rational variables (e.g., X_{t_L} and X_{t_R}) and (ii) adding the constraint $X_{t_L} < X_{t_R}$ to C', for each interval variable t of C. Then the set of variables

$$V' = \{v : v \text{ is a variable of sort } Rational \text{ and } v \in V, \text{ or there exists } t \in V \text{ of sort } Interval \text{ and } v = X_{t_L} \text{ or } v = X_{t_R}\}$$

are eliminated from C' using the algorithm of [10] and the result is translated back to temporal constraints.

Let f be an algebra expression and $\widetilde{T}$ a multitable. The *possible answer* to query f on $\widetilde{T}$ is the set of ground tuples $possible(f, \widetilde{T}) = \{t :$ there exists $r \in sem(f(\widetilde{T}))$ and $t \in r\}$. The *certain answer* to query f on $\widetilde{T}$ is the set of ground tuples $certain(f, \widetilde{T}) = \cap sem(f(\widetilde{T}))$.[11]

Let f be an algebra expression, r a relation and $\widetilde{T}$ a multitable. Deciding whether there is a relation $r' \in sem(f(\widetilde{T}))$ such that $r \subseteq r'$ is called the *possibility problem* for r, f and $\widetilde{T}$. Deciding whether $r \subseteq certain(f, \widetilde{T})$ is called the *certainty problem* for r, f and $\widetilde{T}$. The algorithms in figure 1 solve these two problems.[12] More details on these algorithms and their complexity are given in [11].

6 Related Work and Future Research

In this paper we showed how the ideas of [6, 9, 8] can be integrated to provide a unified treatment of absolute, relative, imprecise or infinite temporal data. We will now compare our proposal with some of the existing temporal database models.

At first, observe that we do not offer any clear analogue to the notions of lifespan of a tuple or attribute as they appear in more traditional models [3]. In our model, temporal information is represented by u- and e-variables which are constrained appropriately. In our opinion, this results in a more flexible and powerful representation scheme.

Up to now, only [2] has studied databases with relative temporal information. The novel feature of [2] is that temporal relationships (e.g., before, after) can appear as attribute values in tuples (e.g., $\langle eventA, before, eventB\rangle$, $\langle eventA, after, 23Jan1992\rangle$ are tuples). This allows one to ask queries of the form "What is the temporal relationship between event A and event B ?". The model of [2] is in some sense weaker than ours since it can capture a less expressive class of temporal constraints (e.g., it cannot represent bounds on the distance of two time points; it cannot represent disjunctive constraints). In addition, its query language is not directly comparable with ours. We will have more on this point in the full version of this paper [11].

[11] If $\mathcal{S}$ is a set of sets then $\cap\mathcal{S} = \cap_{X \in \mathcal{S}} X$.

[12] The algorithms are easy to understand. We simply note that the function call $Eval(\phi, u)$ returns a formula which is obtained by modifying ϕ as follows: (i) all u-variables of ϕ are substituted by constants according to valuation u, and (ii) simplification has been performed so that the resulting expressions conforms with the definition of temporal constraints (e.g., $x - \langle 2,3\rangle_L \leq 5$ becomes $x \leq 7$).

Algorithm $Possibility(r, f, \widetilde{T})$

```
Compute f(T~) and assign the result to ANS
For every subset {t_1, ..., t_|r|} of ANS do
    For every permutation {r_1, ..., r_|r|} of
        the tuples of r do
        For i = 1 to |r| do
            Initialize CON_i to true
        EndFor
        OK := true
        For i = 1 to |r| do
            If r_i unifies with t_i with u-variable
                substitution u_i then
                CON_i := Eval(t_i(CON), u_i)
            Else
                OK := false; Break
            EndIf
        EndFor
        If OK then
            If G(T~) ∧_{i=1}^{|r|} CON_i is consistent then
                Return YES
            EndIf
        EndIf
    EndFor
EndFor
Return NO
```

Algorithm $Certainty(r, f, \widetilde{T})$

```
Compute f(T~) and assign the result to ANS
For every tuple t ∈ r do
    For every tuple t_i ∈ ANS do
        If t_i unifies with t with u-variable
            substitution u_i then
            CON_i := Eval(t_i(CON), u_i)
        Else
            CON_i := false
        EndIf
    EndFor
    If 𝒯 ⊭ G(T~) ⊃ ∨_{i=1}^{|ANS|} CON_i then
        Return NO
    EndIf
EndFor
Return YES
```

Figure 1: Algorithms for possibility and certainty

The recent model of [5] deals with indefinite temporal information. However, only two kinds of indefite information are allowed. The first is a limited case of the kind of indefiniteness which can be expressed in our model: we can say that a tuple is in a relation at least over the interval $[a, b]$ and at most over the interval $[c, d]$ where $[a, b] \subseteq [c, d]$ (i.e., we are uncertain about the existence of the tuple during interval $[c, d] - [a, b]$). The second kind of indefinite information allows attribute values to be unknown during the period of (possible or certain) existence of the tuple they belong. This kind of indefinite information is not expressible in our model.

We have discussed a number of interesting questions concerning the model of temporal tables but many important research problems still remain open. We are currently working on (i) defining a calculus-like query language for temporal tables, and (ii) providing upper and lower bounds for the possibility and certainty problem for various classes of tables and queries. The next step will be to enhance our model with deductive rules. Among other things this will allow us to express causality, persistence [4, 12], recursive queries and so on. Last, but not least, implementation issues must be addressed.

The importance of these results will not be limited to temporal databases. The theory of temporal tables can be reconstructed in a proof-theoretic framework following the methodology of [9]. Then temporal tables can be seen to correspond to simple first-order temporal theories similar in many respects to the ones studied in AI [4, 12]. It will be an interesting exercise to translate our results for temporal tables to results for subsets of these temporal logics.

Acknowledgements

This research was initiated when the author was at the University of Toronto, and was continued at the National Technical University of Athens and the Institute of Computer Science, FORTH. I would like to thank John Mylopoulos, Alberto Mendelzon, Timos Sellis, Yannis Vassiliou, Costas Courcoubetis, Dimitris Plexousakis and Thodoros Topaloglou for their encouragement and support. Gosta Grahne deserves special thanks for sending me a copy of his thesis. Financial support was gratefully received from the Dept. of Computer Science, University of Toronto and the Institute of Computer Science, FORTH.

References

[1] J.F. Allen. Towards a General Model of Action and Time. *Artificial Intelligence*, 23(2):123–154, July 1984.

[2] S. Chaudhuri. Temporal Relationships in Databases. In *Proceedings of the 14th VLDB Conference*, pages 160–170, 1988.

[3] James Clifford and Albert Croker. The Historical Relational Data Model (HRDM) and Algebra Based on Lifespans. In *Proceedings of the International Conference on Data Engineering*, pages 528–537, 1987.

[4] T. Dean and D.V. McDermott. Temporal Data Base Management. *Artificial Intelligence*, 32:1–55, 1987.

[5] S.K. Gadia, S. S. Nair, and Y.-C. Poon. Incomplete information in relation temporal databases. In *Proceedings of VLDB-92*, pages 395–406, 1992.

[6] Gosta Grahne. The Problem of Incomplete Information in Relational Databases. Technical Report Report A-1989-1, Department of Computer Science, University of Helsinki, Finland, 1989. Also published as Lecture Notes in Computer Science 554, Springer Verlag, 1991.

[7] T. Imielinski and W. Lipski. Incomplete Information in Relational Databases. *Journal of ACM*, 31(4):761–791, 1984.

[8] F. Kabanza, J.-M. Stevenne, and P. Wolper. Handling Infinite Temporal Data. In *Proceedings of ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 392–403, 1990.

[9] Paris C. Kanellakis, Gabriel M. Kuper, and Peter Z. Revesz. Constraint Query Languages. In *Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 299–313, 1990.

[10] Manolis Koubarakis. Dense Time and Temporal Constraints with $\neq$. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92)*, pages 24–35. Morgan Kaufmann, San Mateo, CA, October 1992.

[11] Manolis Koubarakis. Representation and Querying in Temporal Databases: the Power of Temporal Constraints. Long version of this paper. Forthcoming., 1993.

[12] R.A Kowalski and M.J. Sergot. A Logic-based Calculus of Events. *New Generation Computing*, 1(4):67–95, 1986.

[13] I. Meiri. Combining Qualitative and Quantitative Constraints in Temporal Reasoning. In *Proceedings of AAAI-91*, pages 260–267, 1991.

[14] John Mylopoulos, Alex Borgida, Matthias Jarke, and Manolis Koubarakis. Telos: A Language for Representing Knowledge About Information Systems. *ACM Transactions on Information Systems*, 8(4):325–362, October 1990.

[15] Yoav Shoham. *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence.* MIT Press, 1987.

[16] R. Snodgrass. Temporal Databases: Status and Research Directions. *Sigmod Record*, 19(4):83–89, 1990.

[17] J. Ullman. *Principles of Data Base and Knowledge Base Systems*, volume 2. Addison Wesley, 1989.