Automatic deduction of temporal information

F. Barbic, R. Maiocchi, and B. Pernici

Dipartimento di Elettronica
Politecnico di Milano
piazza Leonardo da Vinci 32
I-20133 Milano MI, Italy
e-mail: relett07@imipoli.bitnet

May 24, 1990

Contents

1 Introduction **6**

2 TSOS Time Model **11**

2.1 Temporal aspects in information systems 11

2.2 TSOS time modeling concepts 12

2.2.1 Temporal abstractions 13

2.3 Events and propositions 15

2.3.1 Events 15

2.3.2 Propositions 16

2.4 TSOS time calculus 18

2.4.1 Predicates 18

2.4.2 Deduction rules and derived predicates 21

2.4.3 Computation of admissibility intervals 25

3 Temporal Queries **28**

3.1 Introduction 28

3.2 Queries on events 29

3.2.1 Queries based on admissibility interval 29

3.2.2 Necessity and possibility queries 29

3.3 Proposition validity queries 34

4 Further Modeling Features **37**

4.1 Rollback specification 37

4.2 Metalevel temporal assertions 40

5 Architectures for TSOS Time Systems **46**

5.1 Temporal front-end to conventional databases 48

5.2 Specialized temporal systems 48

5.3 Prototype realizations 49

6 Comparison with other Work **51**

7 Concluding Remarks and Future Work 54

8 References 55

List of Figures

1 Calendar time model 13
2 Admissibility interval for TP1 26
3 Example of use of admissibility intervals 28
4 Computation of modal belonging queries 30
5 Computation of a "necessary belonging" query 31
6 Computation of a "necessary distance" query 33
7 *holds(P,M,TQ)?* 36
8 Example of use of modalities in temporal queries 38
9 Example of rollback support 41
10 A simple data model 42
11 Example of instantiation process 45
12 Temporal precedence of the procedure "Personnel management" . . 46
13 Time systems architectures 47
14 Architecture of the system 50

Abstract

In many computer based applications, temporal information has to be stored, retrieved, and related to other temporal information. Several time models have been proposed to manage temporal knowledge in the fields of conceptual modeling, database systems, and artificial intelligence.

In this paper we present TSOS, a system for reasoning about time that can be integrated as a time expert in environments designed for broader problem-solving domains. The main intended goal of TSOS is to allow a user to infer further information on the temporal data stored in the database through a set of deduction rules handling various aspects of time. For this purpose, TSOS provides the capability of answering queries about temporal specifications it has in its temporal database.

Distinctive time modeling features of TSOS are the introduction of *temporal modalities*, i.e., the possibility of specifying if a piece of information is always true within a time interval or if it is only sometimes true, and the capability of answering about the possibility and the necessity of the validity of some information at a given time, the association of temporal knowledge both to *instances of data* and to *types of data*, and the development of a *time calculus* for reasoning on temporal data. Another relevant feature of TSOS is the capability to reason about temporal data specified at different time granularities [1].

[1]This work was partially supported by CEE under ESPRIT Project N. 2409, EQUATOR

1 Introduction

Temporal information is associated to data and processes in many applications: information systems, databases, planning and scheduling systems, real time systems.

Several time models have been proposed to manage temporal knowledge in the last few years. Literature on time modeling is increasing at a steady rate, as demonstrated by comparing two surveys on the subject presented at a distance of four years [Bol 82, McK 86].

The various approaches to time information can be classified in three research areas: conceptual data modeling, whose focus is time representation, database systems, that are concerned with management of temporal information, and artificial intelligence, to address time reasoning.

Research in *conceptual data modeling* is concerned with the conceptual description of time, the representation of the occurrence time of events and of temporal relationships between events.

Although identified as one of the primary components of human knowledge, the role of time in the initial infological approaches was limited to mere indexing of recorded facts [Lan 75]. In fact, the precise characterization of the properties of temporal data and operators over them requires the adoption of ad-hoc temporal models.

A broader perspective to information modeling and time as a modeling construct was first introduced in [Bub 80]. In this approach, a time model is specified independently of any other concept or data modeling construct, with the purpose of defining event occurrence conditions, existence criteria for entities, function and predicate value variations, and various dynamic constraints. The most relevant observation introduced by this approach is that the time aspects of data management and their treatment should be generalized across applications and application areas.

In [Bol 83], further considerations about modeling temporal aspects in information systems are discussed using the notion of abstraction and the machinery of first order logic. The most relevant abstraction discussed in the paper is the *abstraction of time*, which is needed to manage the dichotomy between absolute and relative timing statements. *Absolute timing statements* identify the time of an event with a point or an interval in a linear sequence of time points; on the other hand, *relative timing statements* express time information using timing relationships such as "before" and "during". In particular, absolute times are managed as elements of an abstract domain based solely on relative times.

More recently, several approaches to the conceptual modeling of conventional data have been augmented to support temporal data: for example, extensions to the

entity-relationship model aiming at the semantic specification of the properties of temporal data have been proposed in [Klo 83, Che 86].

The major drawback of the conceptual data modeling approach is that little attention is paid to the problem of managing temporal information, i.e., supporting retrieval of data. Temporal information is substantially supposed to be extracted in the way it was stored; moreover, the issue of dealing with vague or imprecise information is only addressed in few approaches. A more global perspective is thus needed to provide reasoning capabilities to time models, what seems to be the direction of research in the area of database systems.

Database systems must deal with the problem of outdated data. Conventional databases store the values of data that are presently true; although the contents of the database continue to change as new information is added, these changes are viewed as modifications to the present state, with the old data being deleted from the database. In fact, in many applications, such as keeping employee records, temporal information becomes a critical part of the data.

To handle time varying data various models for temporal databases analogous to the relational model for conventional databases and the associated temporal query languages have been designed.

According to the classification given in [Sno 86], databases handling time information can be classified in three classes. *Rollback databases* just add an aspect of time to conventional database management systems by storing all past states, indexed by time, of the database as it changes. Such an approach requires a representation of *transaction time*, i.e., the time the information was stored in the database [Woo 83].

One limitation of supporting only transaction time is that the history of database activities is recorded, rather than the history of the real world; errors in past tuples cannot be corrected. On the other hand, *historical databases* represent only *valid time*, i.e., the time in which the stored information is known to correspond to reality: the entire history of a piece of information is stored and historical queries about the past are supported [Cli 83, Lum 84, Tan 86]. However, in historical databases, previous knowledge, once corrected if erroneous, is not retained any more. Thus, it is not possible to perform rollback operations.

The most complete approach is given by *temporal databases*, that include the aspects of time found in both rollback and historical databases by representing both valid and transaction time [Ari 84, Sno 86]. Time is incorporated into conventional DBMS by storing the entire history, as it is best known at a given point of time, of each relationship being modeled, and storing all past historical states, indexed by

time, of the database as it changes.

Time can be attached to whole tuples [Sno 86, Nav 87] or to single attributes [Cli 84, Tan 86]: in the latter case, it is allowed to express different time information for different attributes.

Several *query languages* incorporating time have also been designed over the last decade. The most relevant approaches in this direction are extensions of conventional query languages, such as Quel [Gad 85, Sno 87] and SQL [Ari 86, Nav 87].

A new direction of research in the area of database systems is the development of *temporal inferencing capabilities* in databases. For instance, the Event Calculus [Kow 85, Sri 88] is an approach for reasoning about events and time within a logic programming framework. The notion of event is taken to be more primitive than that of time and both are explicitly represented by means of Horn clauses augmented with negation by failure.

The main intended applications of the event calculus are the updating of databases and narrative understanding. In contrast with conventional databases which assume that updates are made in the same order as the corresponding events occur in the real world, the explicit treatment of events allows for updates which provide new information about the past. Default reasoning on the basis of incomplete information is obtained as a consequence of using negation by failure. Default conclusions are automatically withdrawn if the addition of new information renders them inconsistent. Since events are differentiated from times, it is possible to represent events with unknown times, as well as events which are partially ordered and concurrent.

Temporal inferencing systems seem to provide more complete solutions to the problems related to time modeling and managing, e.g., dealing with vague temporal specifications and imprecision of information. In fact, the consideration and formalization of such database systems have not been given yet, so many issues concerning time representation remain to be solved (e.g., the representation of periodic and non-connected time intervals, or the dichotomy between absolute and relative time information).

In *Artificial Intelligence* (AI), the problem of representing time is relevant to planning and problem solving in general. In fact, since in this context researchers are interested in providing a useful representation for action reasoning (e.g., planning the actions of a robot), all formalisms must be able to characterize the different types of events, processes, action and properties of the domain at hand.

One of the first contribution acknowledging the importance that an understanding of time plays in many problem-solving situations is given in [Kah 77]. In the paper, the authors propose the construction of a program knowledgeable about time which

is independent of the particular application environment and which can be used by a higher level program to deal with the temporal aspects of its problem solving. Having given the time specialist statements involving temporal references, the general problem-solving program can ask it to make a variety of deductions and to answer a variety of questions concerning such statements. The principal issues addressed in the paper are how the time specialist organizes statements involving temporal references, checks them for consistency, and uses them in answering questions.

Perhaps the most influential theory of time in AI has been introduced by Allen assuming to represent events by intervals rather than by time instants [All 83, All 84]. Time intervals may be related to each other by a set of temporal relations (e.g., overlapping, during) and their inverses; each of these relations is represented by a predicate in the corresponding temporal logic. A set of axioms needed to deduce relationships between any two arbitrary intervals has been developed, based on the transitivity behavior of the relations. Indeed, a difficulty with the proposed computational method is that it requires that all time relations be recomputed whenever a new time relationship is entered.

Properties, processes, and events may be distinguished by considering the characteristics of the set of temporal intervals that they hold or occur over. Properties are said to hold at intervals and all the possible subintervals included in them. In contrast, an event is said to occur over the smallest time interval for it to occur. Processes fall between events and properties: unlike events, processes may occur over subintervals of the general interval in which they occur; unlike properties, however, it is not the case that the process must be occurring on all subintervals of the general interval.

More recently, Dean and McDermott have proposed a temporal system called Time Map Manager (TMM) [Dea 87] designed to support a problem solver that, despite the lack of complete information, is forced to make predictions in order to pursue hypothesis and plans for the future and that has to recompute such predictions when contravened by subsequent evidence. The framework chosen by the authors to deal with this issue is an extended classical predicate calculus database. In order to accomplish such an extension, the notion of an assertion in classical databases has been replaced by that of time token, which refers to an interval of time during which a given fact is alleged to be true.

Temporal information in TMM is allowed to be incomplete, metric, and defeasible, i.e., temporal reasoning proceeds on the basis of assumptions concerning the persistence of certain facts that may turn out to be unwarranted as new information is acquired. TMM's approach to temporal reasoning is implemented by means of a temporal database called time-map. In addition, since the system is supposed to

work in a dynamic environment, the time map machinery extends the functionalities of reason maintenance systems like Doyle's [Doy 79] to handle temporal assertions, keeping track of data dependencies.

The major problem shared by the AI approaches and temporal database systems is maintaining temporal knowledge consistency. The techniques developed for time constraint propagation in AI are similar to those for inferencing in databases, as shown in [Sad 86] comparing the Event Calculus to Allen's temporal logic. As for database systems, in AI many modeling issues (e.g., different time levels, time independence, date reasoning) are rather addressed than developed.

In this paper we present TSOS (Temporal Semantic Office Systems), a system for reasoning about time that can be integrated as a time expert in environments designed for broader problem-solving domains. Although TSOS was first conceived and developed for office applications [Bar 85, Mai 86], it has been significantly extended to provide global functionalities applicable wherever time modeling and reasoning are required. With respect to the literature, it indeed deals with issues related to time representation and management of all the three research areas whose main approaches to time have been discussed above. The main intended goal of TSOS is to allow a user to infer further information on the temporal data stored in the database through a set of deduction rules handling various aspects of time. For this purpose, TSOS provides the capability of answering queries about temporal specifications it has in its temporal database.

Distinctive time modeling features of TSOS which are not covered in literature are the introduction of *temporal modalities*, i.e., the possibility of specifying if a piece of information is always true within a time interval or if it is only sometimes true, and the capability of answering about the possibility and the necessity of the validity of some information at a given time, the association of temporal knowledge both to *instances of data* and to *types of data*, and the development of a *time calculus* for reasoning on temporal data. Another relevant feature of TSOS is the capability to reason about temporal data specified at different time granularities.

The organization of the paper is as follows. Section 2 is dedicated to the presentation of the time modeling aspects of TSOS and of the time calculus to make inferences on temporal data. The queries that can be asked to the system are presented and discussed in Section 3. In Section 4, we discuss various features supported by TSOS: how to associate transaction time to temporal knowledge, how to associate temporal knowledge to types of entities vs. to instances of entities, and how to control the active set of temporal assertions.

Section 5 describes possible architectures for systems supporting time reasoning that include the TSOS time reasoner. In Section 6, we discuss TSOS in comparison

to other approaches to temporal knowledge management, pointing out the relevant modeling characteristics and goals of each of the systems considered with respect to our proposal. Finally, considerations about the implementation of the system are given in Section 7, and further development are envisioned.

2 TSOS Time Model

2.1 Temporal aspects in information systems

In information systems (IS), temporal information is used both in their static part (data) and in their dynamic part (transactions).

In TSOS, we define the concepts of *instantaneous event* and of *proposition* as the basic elements to which temporal information is associated.

Temporal information associated to events and propositions is based on the primitive concept of *time point*, from which other temporal concepts, such as time intervals and durations are derived. Instantaneous events are used to model data to which a single time point is associated, and therefore they are considered instantaneous in the temporal framework of reference for the system. Propositions model data valid over a time span. For instance, a transaction in a database system is usually modeled as an instantaneous event, while a given salary of an employee is valid for a given time interval.

In IS there is also the need to deal with imprecise temporal information. For instance, let us consider the following examples:

"John was hired during 1987",

where a precise date is not specified and

"John was first hired as an employee and then he became assistant manager",

where the precise time of "becoming assistant manager" is not specified, but is related to the time of "being hired".

To handle imprecise time, TSOS supports the concepts of *relative time*, *time granularity*, and *modalities* for propositions.

In Section 2.2, we present TSOS time modeling concepts and in Section 2.3 we specify the concepts of event and proposition. In Section 2.4, the TSOS time calculus is presented.

2.2 TSOS time modeling concepts

The temporal domains on which temporal data may be specified in the model are: time points, time intervals, and time extensions.

Time Points (TPset)

Time is defined in TSOS on a *temporal axis* and is *discrete.*

The **quantum of time** is defined as the most specific non-divisible unit of time [And 82]. The quantum of time to be considered depends on the application domain. In present TSOS implementations, we assumed the minute as the quantum of time considering the calendar time model for information systems.

The set TPset of time points is defined on the most specific unit of time. In the following, we define time points in terms of the set of integers Z.

Time is *infinite* in the past and in the future, so, for each point on the temporal axis, it is possible to find a past and a future point of time.

$$\forall t \in TPset(\exists t' \in TPset \mid t' = t + 1) \wedge (\exists t'' \in T \mid t'' = t - 1) \qquad \text{(D1)}$$

The point of time at the infinite in the past is $-\infty$ and in the future is $+\infty$.

A special TP belonging to TPset is the **current time** (*tpnow*), that represents the present moment (now) moving in time. At each moment, the correspondence between *tpnow* and one of the elements of *TPset* is defined.

Time Intervals (TIset)

Time intervals are defined as a pair (t_s, t_e), such that $t_s, t_e \in TPset$, where t_e and t_s are the **starting** and **ending points** of the interval respectively.

In the following, we adopt the convention that intervals ti_j are **closed** in their **lower end** and **open** in their **upper end**, as in [All 83], and that each time interval is connected.

Let us denote as $start(ti_j)$ and $end(ti_j)$ the lower and upper bounds of an interval, respectively. An interval is defined as a set of time points:

$$\begin{aligned} ti_j \equiv \{t \in TPset \mid\ & t \geq start(ti_j) \wedge \\ & t < end(ti_j) \wedge \\ & \neg \exists t' \in TPset \mid t' \geq start(ti_j) \wedge \\ & \quad t' < end(ti_j) \wedge t' \notin ti_j \end{aligned} \qquad \text{(D2)}$$

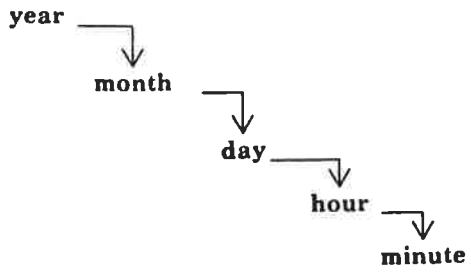


Figure 1: Calendar time model

Following the definition, time intervals are connected.

The set of time intervals TIset is defined as the set of all possible time intervals.

Following from definitions D1 and D2 above, time intervals can be infinite in the past and in the future. Special time intervals are the interval ($tinow^+$) starting at the current time and ending at infinite in the future, and $tinow^-$, the interval spanning from $-\infty$ to *tpnow*.

Time Extensions (TEset)

A time extension denotes a number of time points (quanta of time). Time extensions are used to specify time concepts such as the **distance** of a time point from another time point and the **duration** of a time interval. For example, in the calendar time model "one day" and "two weeks" are time extensions.

Time extensions are mapped on the set of natural numbers N. The set *TEset* of time extensions is defined.

2.2.1 Temporal abstractions

As stated in the previous paragraph, time in TSOS is discrete and it is based on the notion of quantum of time. However, when specifying temporal data, it is often useful to be able to reason at different time levels.

In TSOS, it is possible to express times at different **levels of abstractions**, using the common hierarchical calendar schema for abstractions (Fig. 1), where, for instance, a year is considered to be at a higher level than a month.

The TSOS calendar time model is based on the time model at the quantum of time level, which is the *lowest level of abstraction* (*minlev*) at which it is possible to specify a time element.

We specify *calendar times* in the following form:

$$YY/MO/DD/HH/MM$$

where year YY $\in N$, month MO $\in N$, day DD $\in N$, hour HH $\in N$, and minute MM $\in N$.

The **granularity** of a given time element is defined as the unit of the level of abstraction at which the element is defined. A calendar time CT specified at level G of granularity, where G is not at the lowest level of abstraction, denotes the set of calendar times at the lowest level of abstraction, considering all time points defined at *minlev*. For instance, the calendar time 90/06/11, defined at the "day" level of granularity, denotes the set of times:

$$\{90/06/11/0/0, 90/06/11/0/1, \ldots, 90/06/11/23/58, 90/06/11/23/59\}$$

that is, the underlying model is the time axis at the lowest level of granularity, i.e., the minute level in the IS application domain. We denote with *lower(CT)* the function that returns the smallest time belonging to *TPset* corresponding to a given calendar time CT, and *upper(CT)* the largest time in *TPset* belonging to CT.

In the next section, when introducing the time calculus, we define sum and subtraction operations on times. To avoid errors due to variable length of months (e.g. March is 31 days long, while February can be 28 or 29 days long) and years (leap versus non-leap years), conversions from these levels to the minute level are applied only when actual dates of time points have to be computed. When comparison between times is performed, the month and year levels should be avoided if a greater precision is needed, due to possible conversion errors due to the variable durations of months and years in the calendar.

Times on the calendar model are assigned to **time points** with a date predicate:

date (TP,D)

where $TP \in TPset$ and D is a calendar time. The above predicate indicates that TP belongs to the interval defined as follows:

$$TP \in ti \mid (start(ti) = lower(D) \wedge end(ti) = upper(D) + 1)$$

For instance, if we consider the following date:

d = 90/06/26/-/-

The time point TP may be any time from the first minute (0) of the first hour (0) of June 26th, 1990 to the last minute (59) of the last hour (23) of June 26th, 1990, i.e. in the interval:

[90/06/26/0/0, 90/06/27/0/0)

2.3 Events and propositions

Events and propositions allow the user of the temporal database to assign times to something happening instantaneously and to data that span over a time interval.

2.3.1 Events

We assume that a unique event identifier for events stored in the system exists. We define *EVI* as the set of valid event identifiers.

Instantaneous events happen at a precise time point, and we formally specify this as follows:

happens (EV,TP)

where $EV \in EVI$, and $TP \in TPset$.

Absolute and relative times

The time associated to an instantaneous event, may be absolutely or relatively defined [Bol 83]. *Absolute times* can be located on the time axis, and a single point on the time axis corresponds to the happening time of the event. *Relative times* are specified only indirectly, through references to other times, stating, for instance, precedences between time points.

In fact, in many cases, due to limited available temporal knowledge, it is possible to give only approximate information about the happening time of events. For instance, let us consider the following sentences:

"John became assistant manager in August 1987" (EX1)

"John's salary increased to 40K after he became assistant manager"

Starting from these sentences, it is possible to refer to the time at which the event of "salary increase" took place only indirectly, relating it to another event, the event of "becoming assistant manager". If no additional information is available, it is not possible to state precisely the happening time of both these events.

Admissibility interval

To deal with imprecise information, an *admissibility interval* is associated to the happening time of events.

happens (EV,TP).
adm(TP,TIADM)

where $EV \in EVI$ is an event identifier, TP is its happening time, $TIADM \in TIset$, and TP belongs to the interval $TIADM$.

The **admissibility interval** is the most restricted period of time during which the event may be proven to have happened in the past or is expected to happen in the future according to the temporal knowledge available about that event.

In TSOS, it is assumed that admissibility intervals belong to TIset. In fact, in TSOS disjunctive assertions about possible occurrence times for events are not allowed, and, as a consequence, admissibility intervals are computed as restrictions of intervals.

A goal of the TSOS time calculus is computing the admissibility intervals for events from the given temporal knowledge. Admissibility intervals are the basis for answering queries about happening times of events.

For instance, in the previous example, we can define a set of time points, subset of *TPset*, in which the two events may have happened. Given the information available for example EX1, the admissibility interval for John becoming assistant manager is the set.

{*1987/8/1/0/0/, 1987/8/1/0/1, ..., 1987/8/31/23/58, 1987/8/31/23/59*}

In example EX2, the admissibility interval for the event "John's salary increased to 40K" is the interval starting immediately after "John became assistant manager" (assuming now that no other information is associated to this event, such as it being in the past and so on):

{*1987/8/1/0/1, 1987/8/1/0/2, ...*}

2.3.2 Propositions

Information expressed in *propositions* has a duration in time. For instance, a proposition in used to associate a value to an attribute of an entity for a certain time. Another example of information expressed in propositions are processes, e.g., learning to play chess. Propositions are needed both for the static and for the dynamic part of an information system.

Like for events, we assume we have a unique identifier for facts stated in propositions. We define *PROP* as the set of valid identifiers for facts. To assert their validity, we associate in each proposition a fact to one *observation interval* $OBS \in TIset$. Within the observation interval we can state:

- either that the fact was true for *all* the time points in the observation interval ("*always*")
- or that the fact was true in *at least one* of the time points in the observation interval ("*sometimes*").

The knowledge about an interval does not exclude that the fact could also be true outside the mentioned interval, i.e. it states only known information about its validity within the given observation interval.

Formally, we specify this type of assertions with the following time predicate:

holds (P,M,OBS).

stating that the proposition identified by $P \in PROP$ holds in the observation interval $OBS \in TIset$ with modality M, where $M \in \{always, sometimes\}$.

We call observation interval the reference time interval in the *holds* predicate since no assumption is made about the fact outside this interval.

Let us consider the two following sentences:

"The salary of John from April 1985 to June 1986 was 30K"

"John wrote the final report of the project on May 2, 1990"

In the first case, we need to express the fact that it is known that during the time interval from April 1985 to June 1986, the fact P: *"The salary of John is 30K"* was always true. In the second example, the fact "John writes the final report" is stated to be true at some times during the given interval, but not necessarily at all times; in fact, the action of writing the report may have taken some time, maybe a few hours, but presumably not the whole day. We express formally the above illustrated facts as follows:

holds (salary-of-john-is-30K, always, obs1)

holds (john-writes-the-report, sometimes, obs2)

where

$$\text{obs1} = \{1985/4/1/0/0, \ldots, 1987/6/30/23/59\}$$

and

$$\text{obs2} = \{1990/5/2/0/0, \ldots, 1990/5/2/23/59\}$$

Observation intervals can be defined also in terms of their temporal relationship to other intervals and events happening times. Therefore also observation intervals can be associated with absolute times (like the ones shown above) and relative times.

The deduction rules of the TSOS time calculus can be used to determine observation times as precisely as possible, in order to be able to answer queries about propositions. The time calculus is the same for admissibility and for observation intervals and its goal is to relate the temporal knowledge about the application, in order to answer temporal queries.

2.4 TSOS time calculus

In this section, the TSOS time calculus for time points, time intervals and time extensions is presented. The basic idea in the time calculus is that of relating available information on different time points and intervals to get the *admissibility interval* for time points whose location on the temporal axis in not precisely known.

Predicates, axioms, and deduction rules at the basis of the reasoning mechanism are presented in this section. They constitute the formal specification of the time reasoner, or in logical terms, its axiomatization. Using the set of deduction rules and the assertions stating the initial configuration of the system, the time reasoner is able to find out the admissibility interval for time points.

To enhance readability, rules are given in the format:

$$conclusions \Longleftarrow premises$$

where premises and conclusions are written in a Prolog-like syntax. The translation into Prolog rules is thus immediate.

For ease of presentation, only the main aspects of the TSOS time calculus are given in this section; the complete set of axioms and predicates defined for the TSOS time calculus may be found in [Mar90].

2.4.1 Predicates

In this paragraph, we introduce the predicates defined on the different time categories.

Time Extensions

The following predicates are defined on Time Extensions: [2]:

te-sum (TE,TE1,TE2)

[2] As in Edinburgh Prolog [Clo 81], we denote variables by identifiers starting with an upper-case letter, constants and predicates by identifiers starting with a lower-case letter.

where $TE \in TEset$ is the sum of $TE1, TE2 \in TEset$, as computed for integers.

te-sub (TE,TE1,TE2)

where TE is the subtraction of TE1 from TE2, where TE2 must be greater or equal to TE1.

geq (TE1,TE2)

compares TE1 and TE2 as for positive integers.

Axioms

Axioms for the predicates defined above are the following [3]:

te-sum (TE, 0_{minlev}*, TE)* (AX1)

the sum of a TE and a time extension whose duration is 0 is a TE with the same duration.

te-sub (TE, 0_{minlev}*, TE)* (AX2)

the subtraction of TE and a time extension whose value is 0 is TE.

geq (TE, 0_{minlev}*)* (AX3)

stating the assumption to allow only absolute values.

Time extensions and granularity

The definitions above deal with time extensions defined at the level of minimum granularity, *minlev*, on the axis of quanta of times. As shown in the previous subsection, in TSOS it is possible to define times at different levels of granularity. Accordingly, the above predicates have been extended to the case of several levels of granularity. To be able to sum time extensions specified at different levels of detail, it is necessary to convert them to the same metric. To minimize problems due to conversions of variable duration times such as months and years, we assume that no conversion is performed at these levels when adding and subtracting times. Moreover, to be able to handle times at different granularities, we use pairs of time extensions for denoting the lower and upper bounds of calendar times, i.e. $te = (te_{min}, te_{max})$. We denote with $min(te)$ the lower bound of a time extension with multiple granularity, and $max(te)$ its upper bound. This allows performing operations at the common lowest level, as proposed in [Cli 87].

As an example, let us consider a time extension TE1:

TE1 = 0/0/3/-/- (3 days at the day level of abstraction).

We can assign an internal representation at the minimum level as follows:

[3] 0_{minlev} denotes a time extension of 0 time units at the minimum level, i.e., quanta of time.

TE1 = (0/0/3/0/0, 0/0/3/23/59)

specifying all possible durations for the given time extension at the minute level.

The predicate *te-sum* (and analogously *te-sub*) is defined applying the definition given for *te-sum* above, considering different levels separately. Given, for instance:

TE2 = 0/0/2/3/-

which is handled in TSOS as:

TE2 = (0/0/2/3/0/, 0/0/2/3/59),

the sum of time extensions TE1 and TE2

te-sum (TE3,TE1,TE2)

is the following:

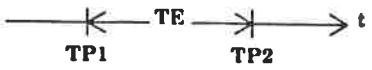
$$\begin{aligned} \text{TE3} &= (0/0/3/0/0 + 0/0/2/3/0,\ 0/0/3/23/59 + 0/0/2/3/59) = \\ &\quad (0/0/5/3/0,\ 0/0/5/26/118) \end{aligned}$$

Time Points

The precedence concept between time points is expressed with the "is-prior-to" and the (derived) "is-beyond" predicates. In TSOS we assume that, for every pair of TP's, it is possible to determine a TE so that the distance between the TP's is TE (when using calendar times with multiple granularity in the calculus, TE is defined as in the previous paragraph).

We introduce the following predicate to relate time points:

is-prior-to (TP1,TE,TP2)



where $TP1, TP2 \in TPset$ and $TE \in TEset$, and $geq(TE, 0_{minlev})$. The predicate states the position of TP1 on the temporal axis in terms of the distance TE from TP2.

Axiom

is-prior-to (TP,0_{minlev},TP)

If the duration of TE is 0 (at the level of quanta of time), the two time points TP1 and TP2 have in fact the same location on the temporal axis.

Time Intervals

The following predicate is used to specify time intervals:

span (TI,TI^-,TI^+)

Time interval TI is defined as starting from time point $TI^- \in TPset$ and ending at time point $TI^+ \in TPset$, where $TI^- = start(TI)$ and $TI^+ = end(TI)$.

The following predicate expresses that a $TP \in TPset$ belongs to the set of points of interval $TI \in TIset$:

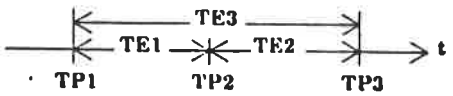
tp-belongs-to (TP,TI)

2.4.2 Deduction rules and derived predicates

- *Transitivity of is-prior-to relationship*

  The transitivity of the *is-prior-to* relationship between time points is expressed by the following rule:

$$\begin{aligned} &\textit{is-prior-to (TP1,TE3,TP3)} \Longleftarrow \\ &\qquad \textit{is-prior-to (TP1,TE1,TP2)}, \\ &\qquad TP2 \neq TP1 \\ &\qquad \textit{is-prior-to (TP2,TE2,TP3)}, \\ &\qquad TP2 \neq TP3 \\ &\qquad \textit{te-sum (TE3,TE1,TE2)} \end{aligned} \tag{R1}$$



Example

For instance, if the following assertions are known:

"Mary became assistant manager 2 months after she was hired" (EX2)

and

"Mary's salary increased 1 month after she became assistant manager"

it is possible to infer that:

"Mary's salary increased 3 months after she was hired".

- *Derived predicates on time points*

  The following predicates are defined to specify temporal precedences between time points:

  is-beyond (TP2,TE,TP1)

  such that:

  is-beyond (TP2,TE,TP1) $\Longleftarrow$ *is-prior-to (TP1,TE,TP2)* (R2)
  is-prior-to(TP2,TE,TP1) $\Longleftarrow$ *is-beyond (TP1,TE,TP2)* (R3)

  precedes(TP1,TP2) and *follows(TP2,TP1)* are defined in a similar way.

  The following theorem holds:

  follows (TP2,TP1) $\Longleftrightarrow$ *precedes (TP1,TP2)*

  The proof can be simply given by using the definition of the two derived predicates in terms of the primitive predicate *is-prior-to*.

  As TE can be 0, according to the definition of *is-prior-to*, we have the following axioms:

  follows(TP,TP)

  precedes(TP,TP)

- *Derived predicates on time intervals*

  Since the TP's of principal interest of a TI are its end points, we allow to directly address these time points with the following predicates:

  begins (TI^-,TI), where $TI \in TIset$ and $TI^- \in TPset$, corresponding to the *start* function.)

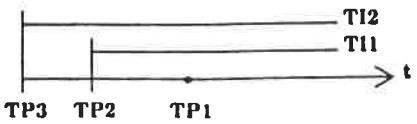
  ends (TI^+,TI), where $TI \in TIset$ and $TI^+ \in TPset$, corresponding to the *end* function.

- *Rules relating time points belonging to different related intervals*

  A set of rules is defined to make deductions about time points belonging to related intervals.

  For instance, the following axiom holds in case *TP1* belongs to an interval *TI1* started by time point *TP2* belonging to *TI2* [4]:

$$\begin{aligned} &\textit{tp-belongs-to(TP1,TI2)} \Leftarrow \\ &\quad \textit{span (TI2,TP3,+}\infty\textit{), span (TI1,TP2,+}\infty\textit{),} \\ &\quad \textit{TI1} \neq \textit{TI2, TP1} \neq \textit{TP2} \\ &\quad \textit{tp-belongs-to (TP1,TI1), tp-belongs-to (TP2,TI2)} \end{aligned} \qquad \text{(R4)}$$



  Example

  As an example, let us consider the following assertion:

  "Mary started working some time after her graduation". (EX3)

  If we assert in addition that Mary got her degree in 1987, the system uses this axiom and the given assertions to prove that:

  "Mary started working after 1987".

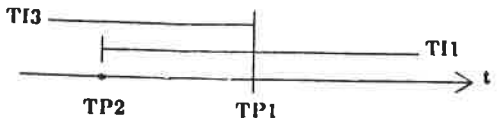
- *Inversion of a belonging predicate.*

  Given two time points *TP1* and *TP2*, if it is known that a time point *TP1* belongs to an interval *TI1* starting from *TP2* and ending at $+\infty$, then the following (inverse) assertion is true: *TP2* belongs to the interval starting from $-\infty$ and ending at *TP1* (under the condition that *TP1* is not the starting time of *TI1*). An analogous axiom may be stated if the interval starts at $-\infty$ and lasts until *TP2*.

  The following axiom belongs to this class:

[4] the equality operator = is defined as in Prolog

tp-belongs-to (TP2,TI3) $\Longleftarrow$ (R5)
tp-belongs-to (TP1,TI1), span (TI3,−∞,TP1),
span (TI1,TP2,+∞), not (begins (TP1,TI1))



Example

For instance, given the following assertion:

"Mary left the job after she was hired" (EX4)

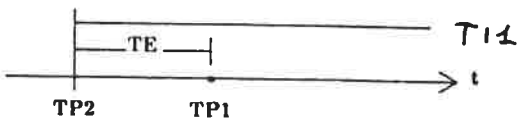
is equivalent to the assertion:

"Mary was hired before leaving the job".

- *A belonging relationship implies an ordering relationship between time points*

  If it is known that a time point *TP1* belongs to interval $TI1$, then it is possible to infer that *TP1* is after the starting time of $TI1$ (TE may be also 0_{minlev}).

precedes(TP2, TP1) $\Longleftarrow$ (R6)
not (TP2 = TP1),
tp-belongs-to(TP1,TI1),
span(TI1,TP2,+∞)



Example

The assertion:

"Mary left the job after she was hired" (EX5)

implies that the hiring event occurred before the leaving one.

- *Derived predicates expressing relationships between intervals*

  Additional (derived) predicates allow to express in a more compact way relationships between intervals. For instance, with the following predicate:

  belongs-to(TI1,TI2)

  where $TI1, TI2 \in TIset$, we introduce the belonging relationship between time intervals [5]:

$$\begin{aligned} &\textit{belongs-to}(TI1,TI2) \Longleftarrow \\ &\quad \textit{span}(TI1,\ TI1^-,\ TI1^+), \\ &\quad \textit{span}(TI2,\ TI2^-,\ TI2^+), \\ &\quad \textit{precedes}(TI2^-,TI1^-), \\ &\quad \textit{precedes}(TI1^-,TI2^-) \end{aligned} \qquad \text{(R7)}$$

  where $TI1^-,\ TI1^+,\ TI2^-,\ TI2^+ \in TPset$.

  Analogously, we may define other predicates on intervals such as: *meets*, *overlaps*, *starts*, *finishes*, *equals-ti*; we can also derive other predicates such as: *intersection*, *union*.

2.4.3 Computation of admissibility intervals

The admissibility interval TI of a time point TP, computed with the time calculus applied on a set of temporal assertions is defined as follows: no contradiction can be shown if, through the insertion of a new temporal assertion, TP is forced to belong to TI, while a contradiction can be shown if TP is forced to be outside TI.

The admissibility interval is computed as the smallest interval in which it is possible to infer that a given time point may be situated. Since we are considering connected time intervals in this paper, the problem is to find the most stringent right and left bounds on the time axis:

[5] In Allen's terminology [All 83], this is the "during" predicate.

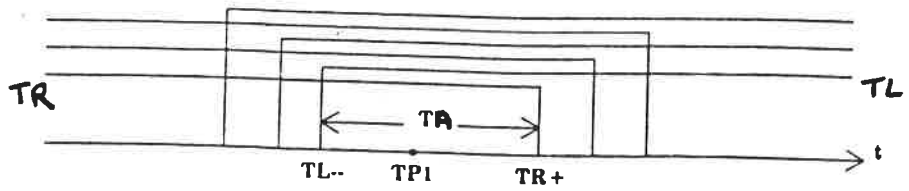


Figure 2: Admissibility interval for TP1

$$\begin{aligned} &\textit{adm (TP1,TADM)} \Longleftarrow \\ &\qquad \textit{span(TADM, TL-, TR+)}, \\ &\qquad \textit{right-adm (TP, TR)}, \\ &\qquad \textit{left-adm (TP, TL)}, \\ &\qquad \textit{span (TR, } -\infty\textit{, TR+)}, \\ &\qquad \textit{span (TL, TL-, } +\infty\textit{)}. \end{aligned} \tag{R8}$$

where *TP1, TP, TL*$^-$ $\in$ *TPset* and *TI, TR, TL* $\in$ *TIset*; *right-adm(TP,TR)* defines the smallest open right interval (i.e., bound to the right) *TR* $\in$ *TIset* such that *TP* $\in$ *TPset* can be proved to belong to the interval. Symmetrically, *left-adm(TP,TL)* defines *TL* as the smallest left interval (i.e. bound to the left) for *TP* (see Fig. 2).

The following rule defines *left-adm* (*right-adm* is defined similarly), stating that no interval can be found among the intervals containing *TP1* that it is contained in *left-adm*:

$$\begin{aligned} &\textit{left-adm(TP, TL)} \Longleftarrow \\ &\qquad \textit{span (TL, TL-, } +\infty\textit{)}, \\ &\qquad \textit{tp-belongs-to (TP, TL)}, \\ &\qquad \textit{not} \quad \textit{(span (TI, TI-, } +\infty\textit{)} \\ &\qquad\qquad \textit{TI} \neq \textit{TL}, \\ &\qquad\qquad \textit{precedes(TL-, TI-)}, \\ &\qquad\qquad \textit{tp-belongs-to (TP, TI)}. \end{aligned} \tag{R9}$$

where *TP, TL*$^-$, *TI*$^-$ $\in$ *TPset* and *TI, TL* $\in$ *TIset*.

Example

Let us consider the following assertions [6]:

"Mary is hired during or after 1980". (EX6)

Formally:

happens(mary-hired, tp1).
tp-belongs-to(tp1,ti1).
span(ti1,ti1-,+∞).
date(ti1-, 1980).

"Mary was hired before John left".

Formally:

happens(john-left,ti2+).
tp-belongs-to(tp1, ti2).
span(ti2, −∞, ti2+).

"John left before 1985".

Formally:

tp-belongs-to(ti2+, ti3).
span(ti3, −∞, ti3+).
date(ti3+, 1985).

"Mary became manager 1 year after her hiring date" [7].

Formally:

happens(mary-mgr,tp2).
is-beyond(tp2,1, tp1).

Let us consider the query:

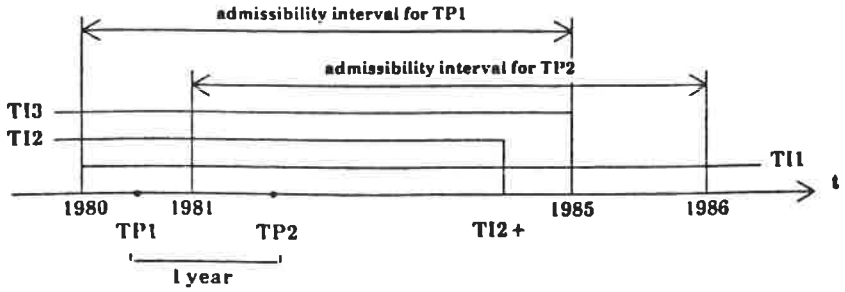


Figure 3: Example of use of admissibility intervals

"When was Mary hired?"

As shown in Fig. 3, the admissibility interval of *tp1* time of the event "mary-hired" is obtained as the intersection of the intervals after 1980 hiring date of Mary, and before 1985, since John's leaving date is not known precisely.

As it is shown in next subsections, the determination of the admissibility interval is critical for answering most types of temporal queries.

3 Temporal Queries

3.1 Introduction

In this section, we consider the temporal queries supported in the TSOS time reasoner.

In general, temporal queries are submitted when a set of **temporal assertions** have been specified. Temporal assertions are primitive or derivate predicates that specify facts valid in a specific domain.

For instance, we showed in the previous section that the fact:

"Mary is hired during or after 1980"

is captured by four TSOS assertions:

Usually, time points can not be univocally associated with dates, as their knowledge

[6] For simplicity, we assume in this example to reason only at the year level. Considering granularity, 1980 should be written 1980/0/0/0/0

[7] Again for simplicity we express 1 year as TE=1; at the minute level, if we consider exactly one year, we should write (1/0/0/0/0, 1/0/0/0/0)

is incomplete. In the above example, we can not tell precisely Mary's hiring date. TSOS provides facilities to query about happening times of events, using the time calculus to infer as much temporal knowledge as possible from the specified temporal assertions.

In the following, we distinguish between *queries on instantaneous events* and *proposition validity queries.*

Temporal *queries on events* have the goal of locating the time point associated to the event on the time axis with the maximum possible precision, given the available set of temporal assertions, based on the time calculus presented in Section 2.

Temporal *queries on propositions* concern the validity of a given proposition during a given temporal interval. Such information is derived from the assertions about observation intervals for the proposition and the time calculus.

3.2 Queries on events

3.2.1 Queries based on admissibility interval

When a time point associated to an event appears within a temporal assertion, the assertion itself restricts the set of admissible values for the time point. It is therefore of interest to identify all the restrictions on the admissible values, enforced directly or indirectly by all the specified assertions, i.e. derive of the **admissibility interval** for the time point, that is the set of possible times when the corresponding event may happen. For instance, given the above mentioned example, we can ask:

adm(tp1, TQ)?

where $TQ \in TIset$, obtaining interval ti1 as an answer if there is no additional information about Mary's hiring date.

3.2.2 Necessity and possibility queries

A type of information that is often in queries is related to the *modality* of the validity of a temporal relationship.

We define a temporal relation to be *necessary* when no further temporal assertion can have as a consequence that the relation does not hold anymore. We define a temporal relation to be *possible* when it can not be proved that the temporal relation does not hold. Necessity and possibility queries can apply to the temporal relations defined in the previous section. The computation of the query is based on an appropriate rule defined to derive the possibility or necessity of the happening

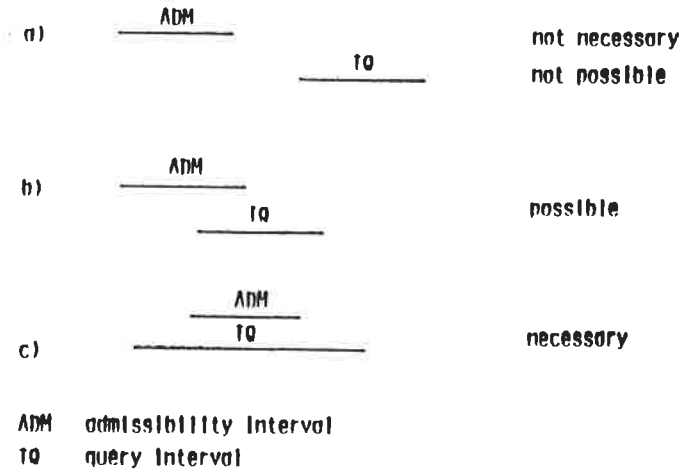


Figure 4: Computation of modal belonging queries

of an event from its admissibility interval.

We examine now modalities for a relation between a time point and an interval and for a relation between time points.

Modal belonging queries

Belonging queries concern the relationship between a given time point and a given time interval. A time point *necessarily* belongs to a time interval if it is prohibited that it occurs outside the interval. A time point *possibly* belongs to an interval if it cannot be proved that it does not belong to the interval.

The computation is based on the relative position of the admissibility interval for the time point and the query interval (see Fig. 4).

Necessary belonging

The rule to derive necessary belonging is the following:

$$\begin{aligned} &\textit{nec-belongs-to(TP1, TQ)} \Longleftarrow \\ &\qquad \textit{span(TQ, TQ-, TQ+)}, \\ &\qquad \textit{adm(TP1,TA)}, \\ &\qquad \textit{belongs-to(TA, TQ)}. \end{aligned} \qquad \text{(R10)}$$

where $TP1,\ TA^-,\ TA^+,\ TQ^-,\ TQ^+ \in TPset$ and $TQ,\ TA \in TIset$.

Example

For instance, we can ask:

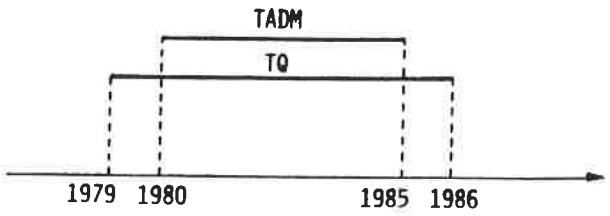


Figure 5: Computation of a "necessary belonging" query

"Was Mary necessarily hired between 1979 and 1986?" (EX7)

Formally, we have:

span(tq, tq-, tq+).
date(tq-,1979).
date(tq+,1986).

If the event "mary-hired" is associated to time point *tp1*, the query is:

nec-belongs-to(tp1, tq)?

Since the admissibility interval for Mary's hiring date is between 1980 and 1985, the query interval fully contains it (Fig. 5). Then rule R10 applies and a positive answer is returned.

Possible belonging

Similarly, if, given a time point *tp1* and a query interval *tq*, we want to know whether or not it is possible, according to the specified temporal assertions and the time calculus, that *tp1* belongs to *tq* we have the following query:

pos-belongs-to(tp1, tq)?

In order to answer to such a query the following axiom is applied:

pos-belongs-to(TP1, TQ) $\Longleftarrow$ (R11)
span(TQ, TQ-, TQ+),
adm(TP1,TA),
overlaps(TA, TQ).

where $TP1$, TA^-, TA^+, TQ^-, $TQ^+ \in TPset$ and TQ, $TA \in TIset$.
The predicate *overlaps(TA, TQ)* is true if the intervals TA and TQ overlap.

Example

For instance, we can ask:

"Is it possible that Mary was hired before 1979?" (EX8)

Since the query interval and the admissibility interval do not overlap, a negative answer is returned.

Modal distance queries

Another type of information of interest is the **precedence relationships** between a pair of time points and the **temporal distance** between them, i.e., the temporal extension beyond the happening of the first event and prior to the happening of the second event. As above, we distinguish between the necessity of a given precedence relationship and the possibility of a given precedence relationship.

Necessary distance

The necessary distance is defined as the minimum possible distance between two time points. Given a pair of time points *tp1* and *tp2*, we want to know the minimum temporal extension, or minimum duration of the temporal interval, that exists between *tp2* and *tp1*, according to the specified temporal assertions and the time calculus. The query is expressed as follows:

nec-is-prior-to(tp2, TE, tp1)?

In order to answer such a query, the following rules are applied [8]:

nec-is-prior-to(TP2, TE, TP1) $\Longleftarrow$ (R13)
is-prior-to(TP2, TE, TP1).

nec-is-prior-to(TP2, TE, TP1) $\Longleftarrow$ (R14)
span(TA1, TA1-, TA1+),
span(TA2, TA2-, TA2+),
adm(TP1, TA1),
adm(TP2, TA2),
is-prior-to(TA2+,TE,TA1-).

[8] With multiple granularity, the rule is modified considering *min(te* instead of *te* in the right hand part of the rules

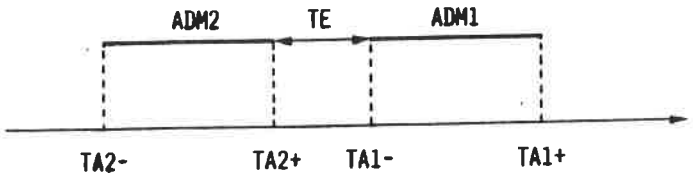


Figure 6: Computation of a "necessary distance" query

where *TP1, TP2,* $TA1^-$, $TA1^+$, $TA2^-$, $TA2^+ \in$ *TPset, TA1, TA2* $\in$ *TIset*, and *TE* $\in$ *TEset.*

The first axiom simply states that if a temporal extension exists between two time points, it is also the minimum distance between them.
The second axiom states that the minimum temporal extension between two time points is the minimum temporal extension between the two respective admissibility intervals (see Fig. 6).

For instance, considering the events Mary being hired (*ev1*, with time point *tp1*) and Mary becoming a manager (*ev2*, with *tp2*), we can ask:

"Is it necessary that Mary becomes a manager after being hired?" (EX9)

i.e.

nec-is-prior-to(tp1,X,tp2)?

In this case, rule (R13) applies, since it is known from (EX6) that

is-prior-to(tp1,1/0/0/0/0,tp2).

If the admissibility intervals overlap, then the answer is negative, otherwise, the answer to the query is positive and the necessary distance returned is the minimum distance between the two events.

Possible distance

Given a pair of time points *tp1* and *tp2*, we want to know the *maximum* temporal extension that may exist beyond *tp2* and prior to *tp1*, according to the specified temporal assertions and the time calculus.
Formally, the query is expressed as follows:

pos-is-prior-to(tp2, TE, tp1)?

In order to answer such a query, the following axiom is applied [9]:

pos-is-prior-to(TP2, TE, TP1) $\Longleftarrow$ (R15)
is-prior-to(TP2, TE, TP1)

pos-is-prior-to(TP2, TE, TP1) $\Longleftarrow$ (R16)
span(TA1, TA1-, TA1+),
span(TA2, TA2-, TA2+),
adm(TP1, TA1),
adm(TP2, TA2),
is-prior-to(TA2-, TE, TA1+).

where $TP1, TP2, TA1^-, TA1^+, TA2^-, TA2^+ \in TPset$, $TA1, TA2 \in TIset$, and $TE \in TEset$.

3.3 Proposition validity queries

Propositions assertions specify that a given fact is true (always or sometimes) during an observation interval. For instance, the following predicate:

holds(john-writes-the-report, sometimes, obs2) (EX10)

where *obs2* represents May 2, 1990, means that there has been at least a time point on that day in which John was writing the report.

It is of interest to answer queries about the validity of a given fact during an observation interval. For instance, the query "Was John writing the report sometimes on May 2, 1990?" will produce a positive answer, while the query "Was John always writing the report on May 2, 1990?" will produce a negative answer.

The predicate *holds* is used to express the validity of a fact during an observation interval, with the temporal modalities *always* or *sometimes*.
proposition validity queries are of the following types:

- *holds(P, always, TQ)?*
- *holds(P, sometimes, TQ)?*

In order to answer such queries, several cases have to be examined to answer such queries, relating the query interval $TQ \in TIset$ to observation intervals for fact $P \in$

[9] Here again we do not consider granularity levels. In that case the rules should be modified considering *max(TE) in the right hand part*

PROP. In Fig. 7 the basic cases are illustrated (other cases can be derived by the repeated application

of the rules for computing the *holds* predicate).

Corresponding to the cases presented in Fig. 7, a number of rules in TSOS allow the computation of the modality for a proposition validity query. For instance, rule (R17) corresponds to case a) of Fig. 7:

holds(P, always, TI-SMALL) $\Leftarrow$ (R17)
belongs-to(TI-SMALL, TI-BIG),
not equals-ti(TI-SMALL, TI-BIG),
holds(P, always, TI-BIG).

This rule states that the fact $P \in PROP$ is always true during an interval *TI-SMALL* $\in$ *TIset*, if it is true during an interval *TI-BIG* $\in$ *TIset* containing *TI-SMALL*.

In tense logic [Res 71], the four operators F, P, G, H are used to qualify the validity of sentence A in the following way:

FA - A is true at some future time;

PA - A was true at some past time;

GA - A will be true at all future times;

HA - A has always been true in the past.

The above axiom corresponds to the tense logic theorem [Res 71]:

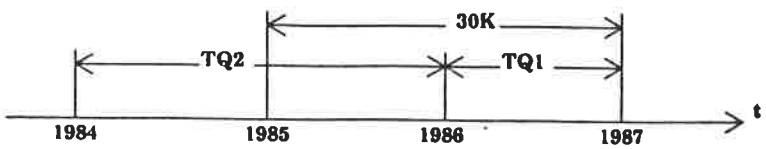
$GP \rightarrow FGP$

stating that if P is always true in the future, then there exists a future time, in whose future, P is always true. The main difference between tense logic and the present approach is that we do not use the current time to express temporal assertions. Hence, we homogeneously deal with past and future.

Example

To show the use of the concepts of validity of temporal assertion, let us consider the following assertion depicted below:

"John's salary has been 30K from 1985 to 1987" (EX11)



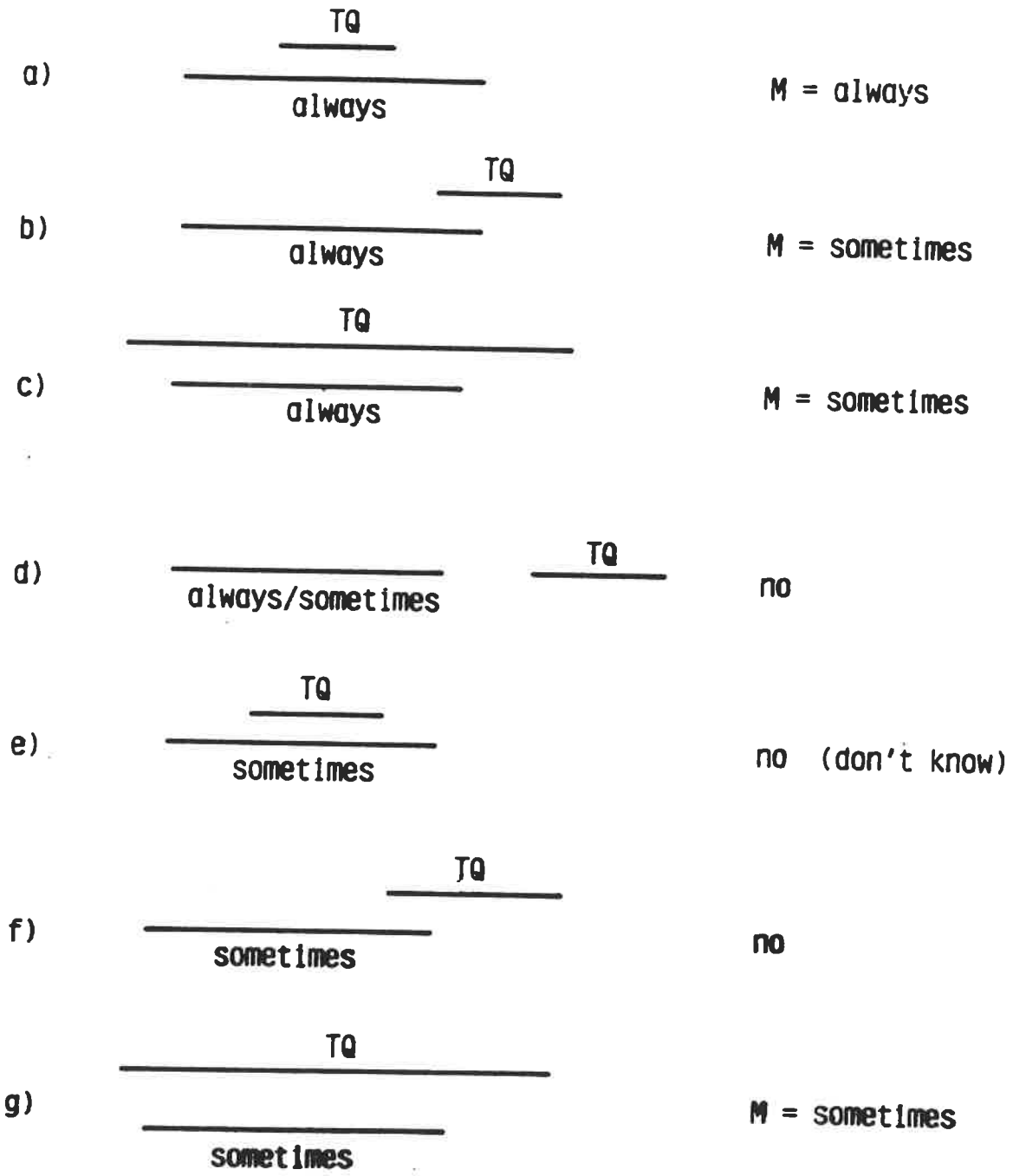


Figure 7: *holds(P,M,TQ)?*

We consider two query intervals:

*tq*1 from 1986 to 1987
*tq*2 from 1984 to 1986.

It is possible to derive temporal knowledge from the above temporal assertions. For instance, we may ask:

"What proposition is valid and with which modality (always, sometimes) during tq1 and tq2?"

In Fig. 8, a session showing the use of modalities in temporal queries is shown. In Fig. 8, the rules used during the query session and the temporal assertions on which time reasoning is performed are presented . The answers presented in the session examples are that John's salary has always (and sometimes) been 30K during *tq*1, while it has sometimes been 30k during *tq*2.

4 Further Modeling Features

4.1 Rollback specification

A reasoning system is often used to perform deductions according to the knowledge available at the present time. Hence, if in the past some mistake was done and afterward corrected, the latest version of the reality is usually used to deduce other knowledge. However, this is not always the case. It is sometimes useful to reason exactly as if the system was in some time point in the past or in the future ("rollback" databases). Reasoning as if in the past allows the user to make queries about information previously known in the system.

Systems allowing to "rollback" to a different state of knowledge must maintain the history and the evolution of the knowledge itself and provide mechanisms to simulate future states. In TSOS this is achieved by maintaining all the temporal assertions entered by the user, even when they are invalidated, and by enforcing the concept of persistence to simulate future states of knowledge. No temporal assertion is ever deleted from the system to preserve the entire history of the specified knowledge.

In TSOS, rollback is applied to propositions, since validity of facts is expressed in propositions, and to *happens* predicates, since time points are assigned to events

```
/****************** EXAMPLE EX11 **************/

/*************** temporal assertions ***************/

/* John's salary was 30K  from 1985 to 1987 */

holds(salary_of_john_is_30K,always,ti1).
span(ti1,ti1m,ti1p).
date(ti1m,1985).
date(ti1p,1987).


/* R17 */

holds(P,always,TI_SMALL) :-
  belongs_to(TI_SMALL,TI_BIG),
  not equals_ti(TI_SMALL,TI_BIG),
  holds(P,always,TI_BIG).


/*************** query **************/

/****** session 1 *******/

/* With what modality is the salary of John 30K in the
interval from 1986 to 1987? */

span(tq1,tq1m,tq1p).
date(tq1m,1986).
date(tq1p,1987).

?- holds(salary_of_john_is_30K,always,ti1).

/***** answer ******/

M = always.


/******* session 2 *****/

/* With what modality is the salary of John 30K in the
interval from 1984 to 1986? */

span(tq2,tq2m,tq2p).
date(tq2m,1984).
date(tq2p,1986).

holds(salary_of_john_is_30K, M, tq2).

/***** answer ******/

M = sometimes.
```

Figure 8: Example of use of modalities in temporal queries

through their happening times. We do not allow invalidating assertions about relationships between time points. Should this be necessary, facts and events are associated to a different time point.

Times associated to facts and events are asserted with the predicate *valid* and invalidated with the predicate *invalid*.

- **Facts**

  valid(P, M, TI, TPV)
  invalid(P, M, TI, TPV)

  where $P \in PROP$, M $\in$ {*sometimes*, *always*}, $TI \in TIset$ express a fact identifier, a modality and an observation interval respectively, as in the *holds* predicate. $TPV \in TPset$ is the time point in which the assertion is made (*transaction time*): with the *valid* predicate the assertion is inserted in the temporal database, with the *invalid* predicate it is retracted.

- **Events**

  valid(EV, TP, TPV)
  invalid(EV, TP, TPV)

  where $EV \in EVI$, $TP \in TPset$ express an event identifier and its happening time respectively, as in the *happens* predicate. *TPV* denotes the transaction time.

The predicate "reasoning-as-of" is used to express the desired reasoning time:

reasoning-as-of(TP)

where $TP \in TPset$.

A set of rules is defined to derive TSOS temporal assertions from *valid* and *invalid* assertions. For instance, the following rule:

holds(P,M,TI) $\Longleftarrow$ (R18)
reasoning-as-of(TP),
valid(P,M,TI,TP1),
precedes(TP1,TP),
not (invalid(P,M,TI,TP2),
precedes(TP2,TP),
precedes(TP1,TP2)).

states that a given proposition P is to be considered valid with a given modality M (*always* or *sometimes*) during an observation interval TI, if P has been asserted at a time point *TP1* prior to the time point *TP* to which the knowledge must be rolled back and if such piece of knowledge has not been invalidated at a time point *TP2* between *TP1* and *TP*.

Rollback specification is normally intended to "roll in the past" the time of reasoning. In general, it is possible to "roll in the future" the time of reasoning as well. In this case, the concept of **persistence** is enforced, which guarantees that every temporal assertion that is holding now, will be holding forever in the future unless invalidated. If no rollback specification is given, by default, the system performs the reasoning "as of now".

Example

Let us consider the following example (EX12):

> *In 1980, the yearly salary of John was asserted to be 30K from 1980 to 1985. In 1982, John's salary was raised to 40K effective from 1983 to 1985, thus modifying the validity of the previous assertion.*

In Fig. 9, the situation is depicted, the corresponding temporal assertions are given and two example sessions are shown. In Session 1, the reasoning is performed "as of 1981" and the salary of John is found to be always 30K during the query interval *tq* (i.e., in 1984). In Session 2, the same query about the salary of John in 1984 is answered differently, since the reasoning is performed "as of 1984". In Session 3, it is shown how in 1984 we can derive that the salary of John was 30K sometimes between 1980 and 1985.

4.2 Metalevel temporal assertions

To this point, we have discussed properties of **instances** of an information system model, e.g., the happening time of a certain event or the value of an entity attribute in a certain period of time. However, in several cases it is interesting to express temporal constraints also at a metalevel: the type level.
In this section, we present a mechanism based on the TSOS time calculus for dealing both with **type** and **instance level temporal assertions**. We assume a model for data elements based on the concept of entity and entity-properties. In Fig. 10, a simple data model is shown, in which, in a company, the entity type "person" has (among others) the properties "noticing-ev" and "leaving-ev". A type level temporal

```
/* Time interval and date definitions */

span(ti1,ti1m,ti1p).
span(ti2,ti2m,ti2p).
span(ti3,ti3m,ti3p).
span(tq,tqm,tqp).

date(tp1,1980).
date(ti1m,1980).
date(ti1p,1985).

date(tp2,1982).
date(ti2m,1980).
date(ti2p,1983).
date(ti3m,1983).
date(ti3p,1985).
date(tqm,1984).
date(tqp,1985).

date(tp4,1981).
date(tp5,1984).

/* validity assertions */

valid(salary-of-john-is-30K,always,ti1,tp1).
invalid(salary-of-john-is-30K,always,ti1,tp2).
valid(salary-of-john-is-30K,always,ti2,tp2).
valid(salary-of-john-is-40K,always,ti3,tp2).


 /* Begin Session 1 */
 reasoning-as-of(tp4).
 ?- holds(salary-of-john-is-30K,X,tq).
    X=always;

 /* Begin Session 2 */
 reasoning-as-of(tp5).
 ?- holds(salary-of-john-is-30K,X,tq).
    no;

 /* Begin Session 3 */
 reasoning-as-of(ti4).
 span(ti4,tp1,ti3p).
 ?- holds(salary-of-john-is-30K,M,ti4).
    sometimes
 /* End Session */
```

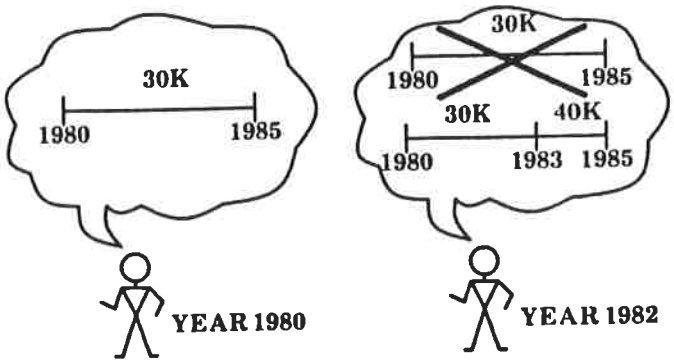


Figure 9: Example of rollback support

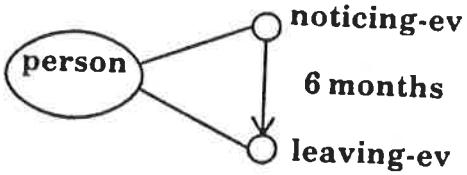


Figure 10: A simple data model

relationship states that a person must give notice of the intention of leaving the company six months before the actual leaving event.

Properties are a convenient way to associate events to entity types.

We define the following sets:

- *ENT*: the set of entity types;
- *PT*: the set of property types, defined as $\{EVT \cup PROPT\}$, union of the set of event types and the set of property types;
- *ENI*: the set of entity identifiers;
- *PI*: the set of property identifiers, where $PI = \{EVI \cup PROP\}$.

We introduce the following predicates for specifying the properties of entity types:

is-entity(Entity-type)
is-property(Property-type, Entity-type)

where *Entity-type* $\in$ *ENT* and *Property-type* $\in$ *PT*.

We introduce the following predicates to define instances of entities and properties:

is-instance(Instance, Type)
is-property-instance(Property-instance, Entity-instance)

where *Instance* $\in$ *ENI* and *Type* $\in$ *ENT* or *Instance* $\in$ *PI* and *Type* $\in$ *PT* and *property-instance* $\in$ *PI* and *Entity-instance* $\in$ *ENI*

The first predicate is used to associate property and entity instances to their types; the second predicate is used to associate property instances to the corresponding entity instances. We assume that properties are single-valued for a given entity instance, i.e. only a value is valid at a given time.

Temporal relationships can be specified among the properties of entity types. Temporal precedences at the type level are asserted with *meta-temporal* predicates. Meta-temporal predicates are the same as temporal predicates defined for time points, time intervals and time extensions, prefixed by *meta*; for instance

meta-is-prior-to(entype, property-type-1, te, property-type-2)

where *entype* $\in$ *ENT*, *property-type-1*, *property-type-2* $\in$ *EVT*, *te* $\in$ *TEset*.

Corresponding temporal relationships among the property instances of an entity instance can be derived through an **instantiation process.**

If *pi1*, *pi2* $\in$ *EVI*, *en* $\in$ *ENI*, and

is-instance(en, entype)
is-instance(pi1, property-type-1)
is-instance(pi2, property-type-2)
is-property-instance(pi1, en)
is-property-instance(pi2, en)

the property instances *pi1* and *pi2* of type *property-type-1* and *property-type-2* respectively are both associated to entity instance *en* of type *entype*.

If the time points of occurrence of events *pi1* and *pi2* are defined as follows:

happens(pi1, tp1)
happens(pi2, tp2)

then, from the *meta-is-prior-to* assertion, the following temporal assertion is derived:

is-prior-to(tp1,te,tp2)

The following rule is defined to derive temporal assertions from meta-level assertions:

is-prior-to(TP1,TE,TP2) $\Longleftarrow$ (R19)

happens(PI1,TP1)
happens(PI2,TP2)
is-instance(PI1,PROPERTY-TYPE-1),
is-instance(PI2,PROPERTY-TYPE-2),
is-property-instance(PI1,PROPERTY-TYPE-1),
is-property-instance(PI2,PROPERTY-TYPE-2),
$PI1 \neq PI2$,
$PROPERTY\text{-}TYPE\text{-}1 \neq PROPERTY\text{-}TYPE\text{-}2$,
is-instance(EN,ENTYPE),
meta-is-prior-to(ENTYPE,PROPERTY-TYPE-1,TE,
PROPERTY-TYPE-2).

where $TP1, TP2 \in TPset$, $TE \in TEset$, $PI1, PI2 \in PI$, $PROPERTY\text{-}TYPE\text{-}1$, $PROPERTY\text{-}TYPE\text{-}2 \in PT$, $EN \in ENI$, $ENTYPE \in ENT$.

Similar axioms have been defined for the other derived and primitive time predicates (*precedes*, *is-beyond*, *follows*). These new axioms and predicates form the meta-level time calculus, that is responsible of deducing temporal knowledge from type level to instance level.

Example (EX13)

In Fig. 11, an instantiation process for the data model of Fig. 10 is shown.
From the type level specifications, it is possible to derive the temporal relationship between the properties of an entity instance, e.g., Mary's noticing event happens six months before her leaving event.

Example (EX14)

An important application of the above approach is the possible control over the time of the **activities of a procedure instance**, from the knowledge about the procedure type.
In Fig. 12, the temporal precedences of the procedure called "Personnel management" is shown. A person may receive a training after being hired. Promotions are given to some trained employees at least 12 months after training. Employees must give notice of their leaving six months before actual leaving. Employees may be fired, in which case they leave after a variable period of time. Not all the activities of the procedure apply to each person, but if two activities exist for a given person, the corresponding temporal relationships are derived from the type level specification.
For instance, it is possible to derive that Mary can be promoted any time after 12

```
/*************** EXAMPLE EX13: R19 ************/

/********** temporal assertions **********/

/* in a company there is a rule stating that
at least 6 months must go by between the time when
a person notifies his/her decision to leave the office
and the actual resignment */

is_property(noticing_ev,person).
is_property(leaving_ev,person).
meta_is_prior_to(person,noticing_ev,6,leaving_ev).


/***************** axioms **************/

/* R19 */

is_prior_to(TP1,TE,TP2) :-
 happens(INST1,TP1),
 happens(INST2,TP2),
 is_instance(INST1,PROPERTY1),
 is_instance(INST2,PROPERTY2),
 is_property_instance(INST1,INST0),
 INST1 \= INST2,
 PROPERTY1 \= PROPERTY2,
 is_property_instance(INST2,INST0),
 is_instance(INST0,ENTITY_TYPE),
 meta_is_prior_to(ENTITY_TYPE,PROPERTY1,TE,PROPERTY2).


/***************** query ****************/

/* if Mary is a person working at that company, how long before
leaving should she give notice her decision leave */

is_instance(mary,person).
is_instance(n1,noticing_ev).
is_instance(l1,leaving_ev).
is_property_instance(n1,mary).
is_property_instance(l1,mary).
happens(n1,tp1).
happens(l1,tp2).
is_prior_to(tp1, TE, tp2).

/***** answer *******/

TE = 6
```

Figure 11: Example of instantiation process

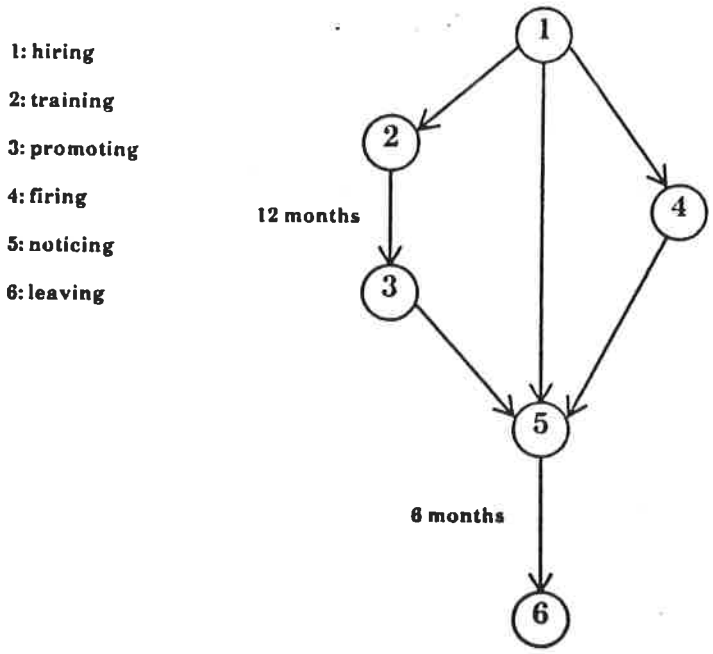


Figure 12: Temporal precedence of the procedure "Personnel management"

months have elapsed from her hiring date (depending on her training).

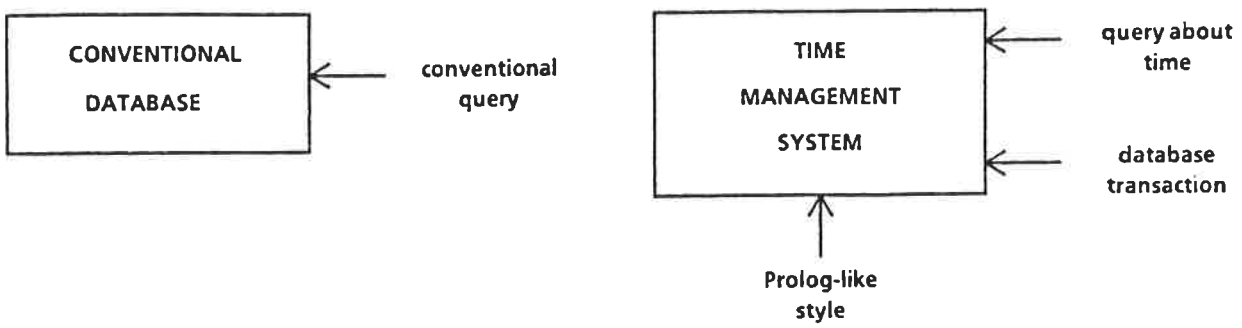
5 Architectures for TSOS Time Systems

The TSOS reasoning system is a fully temporal system according to Snodgrass taxonomy [Sno 87]. In fact, it supports historical queries and rollback. The realization of the TSOS reasoning system presents problems similar to those encountered in the implementation of logic based query languages [Cer 86, Cer 87]. Two different architectural approaches can be considered (Fig. 13):

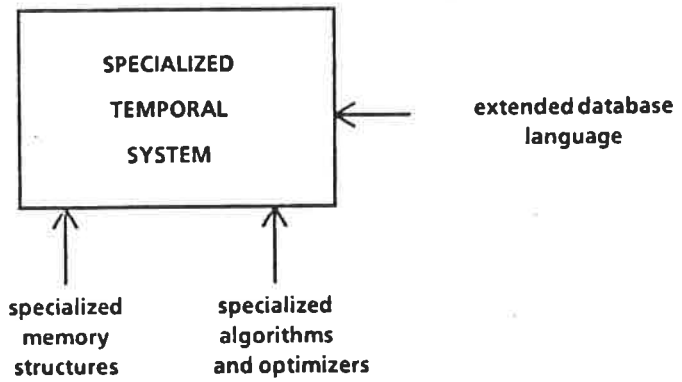
a) providing a **temporal front-end** to a conventional database system (loose coupling);

b) the realization of a **specialized temporal system** (strong coupling).

In Sections 5.1 and 5.2 we discuss the characteristics of the two architectures, and in Section 5.3 we present some prototype implementations.



a) temporal front-end (loose coupling)



b) specialized system (strong coupling)

Figure 13: Time systems architectures

5.1 Temporal front-end to conventional databases

A temporal front-end to a conventional database (e.g., a relational database) accepts queries about time and ordinary database transactions and converts them into queries in a conventional query language (e.g., SQL [Dat 85]).

The conventional database schema includes time related attributes, using one of the time stamping techniques proposed in the literature (e.g., in [Sno 87]). Temporal facts are retrieved from the database and loaded in main memory to be processed by the time reasoning component of the temporal front end. Algorithms are to be studied to minimize the number of interactions with the database, to select only relevant data and to avoid repeated retrieval of the same data for a same query.

The advantage of this type of architecture is that available database management systems can be used to handle non time-related issues (e.g., retrieval of non time-related data, indexing, concurrent accesses, and so on). On the other hand, this type of DBMS do not provide any explicit support for handling time, which is represented using regular attributes of relations.

Updates to the database must be handled by the time management system in order to store the temporal information contained in update statements correctly (i.e., inserting appropriate "valid" and "invalid" data).

5.2 Specialized temporal systems

An alternative solution is that of creating a new type of database management system able to handle directly temporal information.

In this type of solution, the study of appropriate memorization structures, techniques for indexing and compacting temporal data, algorithms and optimizers to extract temporal information efficiently are to be studied.

The result is a system with better performance characteristics compared to those obtained adding a temporal front-end to a conventional database.
Disadvantages are the necessity of re-implementing all non time-related features of existing DBMS in the new system. Moreover, existing query and manipulation languages should be extended with temporal constructs, at the same time offering an unchanged interface to users not interested in temporal aspects of data.

5.3 Prototype realizations

Some experiments have been performed for realizing a time reasoning system based on TSOS model and time calculus. The architecture of the system is presented in Fig. 14. Different modules in the system deal with (incomplete) temporal assertions, considering both type and instance level, and retrieve relevant facts if rollback is requested, in order to produce answers.

The architecture shown in Fig. 14 is close to the "temporal front-end" approach presented in Section 5.1, since temporal assertions could be stored in a conventional database without a big transformation effort.

In the system of Fig. 14, temporal knowledge is entered in the form of atomic formulae, called **Temporal assertions**, using a Prolog-like syntax.
As mentioned in Section 4, temporal assertions can be specified either at the **Instance level** or at the **Type level**. Since reasoning is carried out at the instance level only, the **Instantiation Module** is responsible to derive additional instance level facts from the type level assertions.
The **Totality of Facts** is then screened by the **Time Calculus Module**, according to the **Query Specification**. **Rollback specification** is handled first. It defines the time the system must simulate in its reasoning (e.g., "as of December 1986"). Only the knowledge actually available at the time given in the Rollback specification should be usable to perform the reasoning. Hence, some of the facts are removed and only the remaining subset of **Relevant Facts** is subsequently accessible.
The second portion of the Query specification is the **Temporal goal** (e.g., "Is it necessary that Mary has been trained before 1985?"). The Relevant Facts are used by the **Time Calculus Module** to try to satisfy the Temporal goal. Notice that, for the time independence concept, the Time Calculus Module only deals with temporal information and not with data and events. The **Answers** are the output (either success of failure) of this process.

The TSOS reasoning system is not responsible for maintaining the consistency of the temporal assertions. The user, when new events happen or when temporal requirements change, is responsible for specifying new temporal assertions and, when necessary, for invalidating some of the previously specified temporal assertions.
The problem of automatically preserving the consistency of a set of temporal assertions has been considered in [All 83] and [Bar 87]. Allen shows that the time complexity of the best algorithms solving the problem is exponential in the number of assertions, while [Bar 87] introduces a set of heuristics able to find sub-optimal solutions in polynomial time.

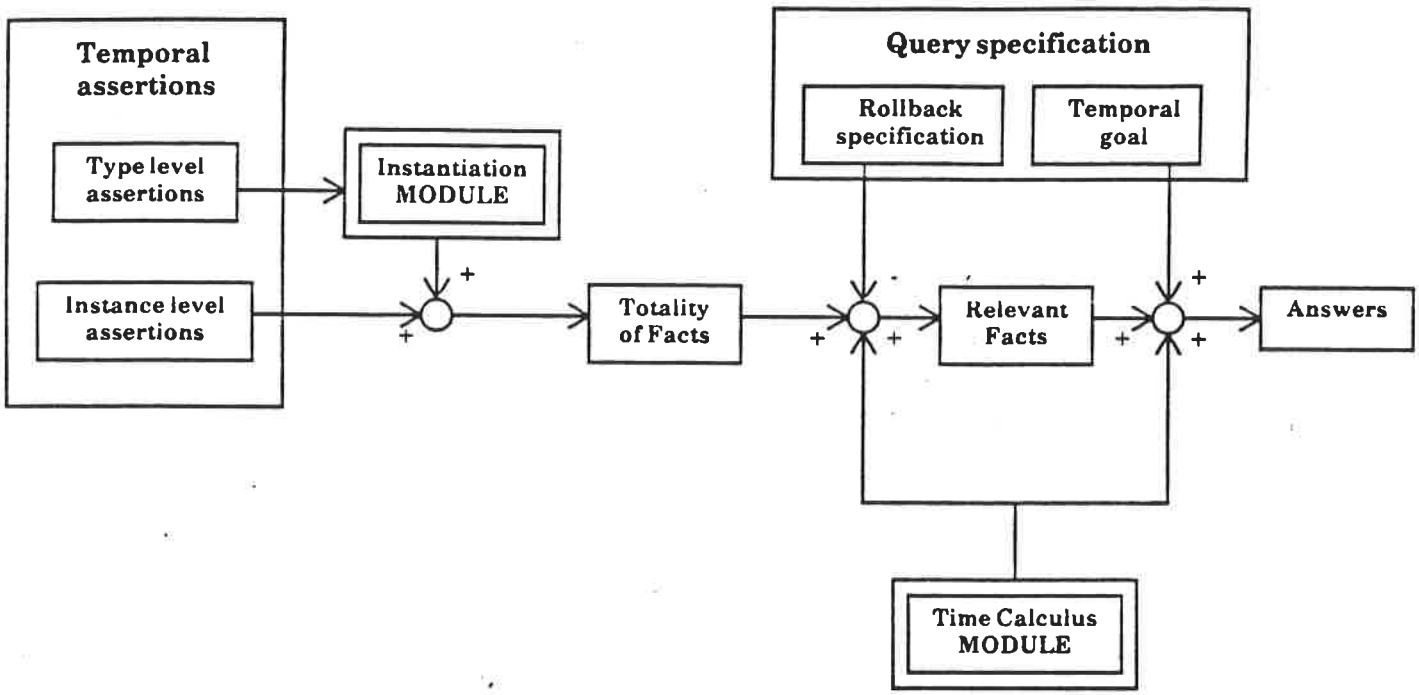


Figure 14: Architecture of the system

TSOS has been implemented both in Prolog and in C [Mar 90]. The advantage of Prolog is the possibility of using directly in the language the inference rules defined in this paper. However, some looping problems exist for some queries. On the other hand, implementation in C is more flexible, and consistency checking support during insertion of assertions based on the time calculus is easily provided, although the performance of the system can become critical in some cases.

6 Comparison with other Work

In this section, we compare some of the most influential approaches to time modeling surveyed in the introduction with TSOS. In the comparison, the emphasis is on the original contributions of TSOS with respect to the literature. For this reason, the following discussion focuses on the distinctive modeling features of our proposal.

There are two aspects that must be taken as a benchmark for comparison of temporal systems: *time representation* and *time reasoning*. Properties of temporal systems can indeed be classified in these two categories no matter what is the application for which they have been designed.

Relevant issues that need to be dealt with for time representation are: the choice of a *primitive time entity* (time points vs. time intervals), the specification of a *time structure* (i.e., mapping time either on Z, or Q, or R), the choice of a *time ordering* (linear, circular, branching time), the *time boundedness* (finite vs. infinite times), the definition of a *time metrics* (i.e., the possibility of adding/subtracting time values and dates).

The specification of such issues provides an ontology for the representation of temporal information. Once these assumptions have been made, temporal data can be associated to conventional data (e.g., properties and events) by the definition of a proper formalism or a model of the data.

The above mentioned time representation issues have been clearly specified in TSOS. Association of time to properties and events has then been defined through the specification of Horn clauses augmented with negation by failure. Such an approach is similar to another proposal in literature, Kowalski and Sergot's Event Calculus. An analogous choice has been made by Allen and by Dean and McDermott introducing first order predicate logic-based formalisms. Although some of the predicates of TSOS appear similar to these proposals, they in fact provide a different semantics.

The main distinctive feature of TSOS is the introduction of temporal modalities, i.e., the possibility of specifying explicitly if a piece of information is always true within a time interval or if it is only sometimes true. This issue has not been

investigated in detail by any of the time models proposed so far in the literature on database systems. The only contribution in this direction has been proposed in [All 84] to distinguish among the concepts of property, event, and process. In that paper, Allen introduces the following predicates:

HOLDS(p,t): if a property p holds over an interval t, it holds over all subintervals of t;
OCCUR(e,t): if an event e occurs over an interval t, there is no subinterval of t over which the event happened;
OCCURRING(p,t): if a process p is occurring over an interval t, it must be occurring over at least one subinterval of t.

From the previous definitions, the HOLDS and OCCUR predicates imply implicitly that property p holds *always* in t and event e happens *always* in t, whereas process p is occurring *sometimes* in t. Such an approach has some limitations with respect to the explicit management of temporal modalities proposed in TSOS. For instance, Allen cannot express time-related information such as: *holds(p, sometimes, t)* where p is a property and t is the time interval over p is true.

In TSOS, we have also introduced the possibility to associate temporal knowledge both to instances of data and to types of data. In [Cli 83], the author discuss the difference between *extensional* and *intensional database constraints*. The following definitions are given:

1. an extensional database constraint is a constraint on individual valid states of the database;

2. an intensional database constraint is a constraint which defines valid state progressions in the database.

Whereas extensional constraints can be considered to refer to instances of data, intensional constraints can refer to types of data. An example of intensional constraint given in [Cli 83] is indeed the following: "No employee can ever be given a cut in pay". The advantage of TSOS over Clifford and Warren's approach is that we use predicates on types not only to express constraints on the evolution of the database, but also to answer queries involving temporal reasoning.

Reasoning on time requires the consideration of the following aspects: the specification of a *time calculus* for the management of temporal information, the development of a language for asking *time-related queries* to extract temporal information from the database, and the design of *mechanisms for database maintenance*, including an organization of temporal data, consistency checking of temporal assertions,

support for data persistency, and consideration of different time dimensions (valid time vs. transaction time). Since TSOS was conceived as a system for reasoning about time to be integrated in environments for broader problem solving domains, major emphasis has been given to the development of temporal reasoning capabilities than to the issues related to temporal data maintenance, which should be consistent with the general information system.

In particular, we have developed a time calculus both for relative and absolute temporal data specified at different time granularities. Although the need of providing a complete time calculus has been addressed by various authors in literature, a global treatment of all aspects involved in dealing with computing time has not been given. For example, a distinction between relative and absolute time statements has been introduced by Kowalski and Sergot and by Allen without any consideration for time granularity. Vice versa, time granularity limited to absolute times has been addressed by Snodgrass and only recently by Clifford [Cli 87] and Date [Dat 88].

An attempt to consider both aspects together in the same model has been proposed by Kahn and Gorry, without a precise formalization of the computation mechanisms. In fact, whereas TSOS provides general inference capabilities for time reasoning, Kahn and Gorry's time specialist performs its reasoning on the basis of specific temporal data organization that do not cover all possible computations that can be necessary to answer a query. For this reason, they indicate the possibility to invoke a breadth-first search as a last resort when all other inference methods fail; however, how this last resort works is not discussed in detail.

Query answering in TSOS is performed using the time calculus to deduce further information on the temporal data stored in the database. Answers are given also if only incomplete temporal information is available. In addition, we have introduced temporal modalities also in the query language, i.e., the system is capable of answering about the possibility and the necessity of the validity of a piece of information at a given time. With respect to the proposals for extensions of conventional query languages to incorporate time, we provide a declarative semantics which is not necessarily linked to any data model. In the previous section, we discussed how we plan to integrate TSOS with a relational model of the data extended to include time. On the other hand, TSOS falls short in providing explanations about its deductions, a feature supported both by Kahn and Gorry's time specialist and by Dean and McDermott's TMM.

The main mechanism for temporal data maintenance supported by TSOS is the management of valid time and transaction time. Such a distinction, first presented by Snodgrass in [Sno 85], has also been recently introduced by Sripada in the Event Calculus [Sri 88].

In TSOS, data are assumed to be persistent until explicitly stated otherwise through the *valid/invalid predicates*. No automatic clipping on the persistence of the validity of data is supported; in fact, extending the system to provide persistency clipping through such predicates is straightforward. References to such an issue can be found both in [Kow 85] and [Dea 87].

7 Concluding Remarks and Future Work

In this paper, we have discussed how we tackled in TSOS the problem of introducing time reasoning capabilities in databases. Main features of the TSOS time model and time calculus are the consideration of metalevel temporal assertions, time granularity, modalities in queries and assertions about facts, and the management of relative and absolute times.

Queries about temporal knowledge have been discussed in Sections 3 and 4. The concept of distinguishing the case of necessity from that of possibility of happening of events and facts during a given time is presented and examined in detail. We discussed the problems of when temporal relationships should be specified and their span of validity. Then, we introduced a way of associating a validity time to temporal assertions themselves, to be able to query about time not only given the present knowledge, but also assuming to be in a situation at any time in the past (rollback). Finally, we introduced a way of specifying temporal constraints both at type and at instance level.

The application range of the proposed concepts is very broad: artificial intelligence, databases, information systems, office systems, job-shop scheduling.

TSOS has been applied to activity planning [Bar 87] and distributed procedures control [Fug 87]. A TSOS interface has been provided to inquire about completion times of office procedures, described with time extended Petri Nets, and relative times between events occurring in the procedure [Per 89]. The system has also been experimented within the Equator ESPRIT Project, in particular to model and reason about time in urban traffic control [Equ 89]. Implementation has been discussed in Section 5.

Several issues are still open for investigation and future work:

- Further research is needed to ensure the *technical feasibility of large time reasoning systems*, choosing among architectures presented in Section 5 and using advanced technology.
- *Answers' expressiveness* should be augmented providing explanation about

the results of the query. Some limited support has been offered in [Mar 90]. When trying to enter in the database a temporal assertion that it is not consistent with those that have already been stored, the system explains why such an assertion cannot be accepted, showing the admissibility interval for a time point or conflicting observation intervals for facts. Future work should concentrate on the study of the types of explanations appropriate for each type of query, and of the appropriate level of detail.

- *Consistency checking* techniques should be investigated, with particular attention to their complexity.
- The time model should take into consideration also other types of times, such as *non-connected intervals* and *periodic times*. However, only limited time reasoning functionalities can be provided for such times and consistency checking is very difficult, in particular when handling sets composed of an infinite number of elements.

Acknowledgments

We are thankful to F.A. Schreiber and S. Navathe for their discussions on this work, and to M.G. Fugini for their comments on an earlier draft of this manuscript. S. Ceri gave valuable suggestions for time reasoning systems architectures.

We acknowledge the contribution of S. Pozzi, N. Speroni, and G. Ottolini, of D. Finke and P. Giudici, and of S. Marcotullio and E. Sanzo for implementing part of the TSOS model and time calculus in their theses.

8 References

/All 83/ Allen, J.F., "Maintaining Knowledge about Temporal Intervals", Communications of the ACM, vol. 26, n. 11, pp. 832-843 (November 1983)

/All 84/ Allen, J.F., "Towards a General Theory of Action and Time", Artificial Intelligence, vol. 23, n. 2, pp. 123-154 (July 1984)

/Ari 84/ Ariav, G. and Clifford, J., "A System Architecture for Temporally Oriented Data Management", Proceedings of the 5th International Conference on Information Systems, pp. 177-186, Tucson, AZ (November 1984)

/Ari 86/ Ariav, G., "A Temporally Oriented Data Model", ACM Trans. on Data Base System, vol. 11, n. 4 (Dec. 1986)

/**Bar 85**/ Barbic, F. and Pernici, B., "Time Modeling in Office Information Systems", in ACM-SIGMOD International Conference on Management of Data, ed. S. Navathe, pp. 51-62, Austin, TX (May 28-31, 1985)

/**Bar 87a**/ Barbic, F. and Maiocchi, R., "Planning in Time", Afcet-IFIP WG8.1 TAIS Conference, pp. 147-161, Sophia-Antipolis (May 1987)

/**Bol 82**/ Bolour, A., Anderson, T.L., Dekeyser, L.J., and Wong, H.K.T., "The Role of Time in Information Processing: A Survey", ACM SIGMOD RECORD, vol. 12, n. 3, pp. 28-48 (1982)

/**Bol 83**/ Bolour, A. and Dekeyser, L.J., "Abstractions in Temporal Information", Information Systems, vol. 8, n. 1, pp. 41-49 (1983)

/**Bub 80**/ Bubenko, J.A., "Information Modeling in the Context of System Development", in IFIP Information Processing 80, ed. Lavington, S.H., pp. 395-411, North-Holland (1980)

/**Cer 86**/ Ceri, S., Gottlob, G., and Wiederhold, G., "Interfacing relational databases and Prolog efficiently", Expert Database Systems Conf., Charleston (1986)

/**Cer 87**/ Ceri, S. and Tanca, L., "Optimization of systems of algebraic equations for evaluating Datalog queries", Very Large Databases Conf., Brighton (1987)

/**Che 86**/ Chen, P.P.S., "The time dimension in the Entity-Relationship Model", in IFIP Information Processing 1986, ed. Kugler, H.-J., North-Holland (1986)

/**Cli 83**/ Clifford, J. and Warren, D.S., "Formal semantics for time in databases", ACM Transactions on Database Systems, vol. 8, n. 2, pp. 214-254 (June 1983)

/**Cli 84**/ Clifford, J., "Towards an algebra of historical relational databases", Center for Research and Information Systems - New York University - #84-91(CR) (December 1984)

/**Cli 87**/ Clifford, J. and Rao, A., "A simple, general structure for temporal domains", Afcet-IFIP WG8.1 TAIS Conference, pp. 17-28, Sophia-Antipolis (F) (May 1987)

/**Clo 81**/ Clocksin, W.F. and Mellish, C.S., Programming in Prolog, Springer-Verlag, New York (1981)

/**Dat 85**/ Date, C.J., An Introduction to Database Systems, Addison Wesley, Reading, Ma (1985)

/**Dat 88**/ Date, C.J., "A Proposal for Adding Date and Time Support to SQL", SIGMOD RECORD, Vol.17, No.2 (June 1988), pp. 53-76.

/**Dea 87**/ Dean, T.L., and McDermott, D.V., "Temporal Data Base Management", Artificial Intelligence 32, 1 (1987), pp. 1-55

/**Doy 79**/ "Doyle, J., "A truth Maintenance System", Artificial Intelligence 12, 3 (1979), pp. 231-272

/**Equ 89**/ Equator (Environment for Qualitative TempOral Reasoning), "How To apply the GRF" Position Paper, (November 1989)

/**Fug 87**/ Fugini, M.G., Maiocchi, R., and Zicari, R., "Time management in the Office-net system", IFIP WG8.4 Workshop on Office Knowledge, Toronto, Canada (August 17-19, 1987)

/**Gad 85**/ Gadia, S.K. and Vaishnav, J.H., "A query language for homogeneous temporal database", Proc. ACM Symposium on Principle of Database Systems (April 1985)

/**Kah 77**/ Kahn, K., and Gorry, G.A., "Mechanizing Temporal Knowledge", Artificial Intelligence 9 (1977), pp. 87-108

/**Klo 83**/ Klopprogge, M.R. and Lockemann, P.C., "Modeling information preserving databases: consequences of the concept of time", in Very Large Data Bases Conference, ed. Schkolnik, M. and Thanos, C., pp. 399-416, Firenze (November 1983)

/**Kow 85**/ Kowalski, R. and Sergot, M., "A logic based calculus of events", New Generation Computing, vol. 4, pp. 67-95 (1986)

/**Lan 75**/ Langefors, B. and Sundgren, B., Information Systems Architecture, Petrocelli/Charter, New York (1975)

/**Lum 84**/ Lum, V., Dadam, P., Erber, R., Guenauer, J., Pistor, P., Walch, G., Werner, H., and Woodfill, J., "Designing DBMS support for the temporal dimension", ACM SIGMOD International Conference on Management of Data, pp. 115-130, Boston, MA (June 1984)

/Mai 86/ Maiocchi, R. and Pernici, B., "Time reasoning in the office environment", in Methods and Tools for Office Systems, Bracchi, G. and Tsichritzis, D. (eds.), North-Holland (1987)

/Mar 90/ Marcotullio, S., and Sanzo, E., "Tesys: Un Sistema per la Rappresentazione e la Gestione di Dati Temporali", Graduation Thesis, Dipartimento di Elettronica, Politecnico di Milano (1990) (in Italian)

/McK 86/ McKenzie, E., "Bibliography: Temporal Databases", ACM SIGMOD RECORD, vol. 15, n. 4, pp. 40-52 (December 1986)

/Nav 87/ Navathe, S.B. and Ahmed, R., "TSQL: A language interface for history databases", Afcet-IFIP WG8.1 TAIS Conference, pp. 113-128, Sophia-Antipolis (F) (May 1987)

/Per 89/ Pernici, B., and Pezze', M., "Temporal Analysis of Office Procedures", CEFRIEL Report, (October 1989)

/Res 71/ Rescher, N., and Urquhart, A., "Temporal Logic", Springer-Verlag (1971)

/Sad 87/ Sadri, F., "Three Recent Approaches To Temporal Reasoning", in Temporal Logics and their Applications, Galton, A. (Ed.), pp. 121-168 (1987)

/Sno 86/ Snodgrass, R. and Ahn, I., "Temporal databases", IEEE Computer, pp. 35-42 (September 1986)

/Sno 87/ Snodgrass, R., "The temporal query language TQuel", ACM Transactions on Database Systems, vol. 12, n. 2 (June 1987)

/Sri 88/ Sripada, S.M., "A Logical Framework for Temporal Deductive Databases", 14th VLDB Conference, Los Angeles (CA), pp. 171-182 (1988)

/Tan 86/ Tansel, A.U., "Adding time dimension to relational model and extending relational algebra", Information Systems, vol. 11, n. 4, pp. 343-355 (1986)

/Woo 83/ Woodfill, J. and Stonebraker, M., "An implementation of hypothetical relations", in Very Large Data Bases Conference, Schkolnick, M. and Thanos, C. (eds.), pp. 157-166, Florence, I (1983)