A Temporal Database Model Supporting Relative and Absolute Time

BRIAN KNIGHT AND JIXIN MA

School of Computing and Information Technology, University of Greenwich, Woolwich, London SE18 6PF, UK.

This paper presents a temporal database model which allows the expression of relative temporal knowledge of data transaction and data validity times. The system is founded on an extension to Allen's axiomitization of time, given previously by the authors, which takes both intervals and points as primitive time elements. A general retrieval mechanism is presented for a database with a purely relative temporal knowledge which allows queries with temporal constraints in terms of any logical combination of Allen's temporal predicates. When absolute temporal duration knowledge is added, the consistency checking algorithm upon which the inference mechanism is based reduces to a linear programming problem. A class of databases, termed time-limited databases, is introduced as a practical solution to the problem of computational complexity of retrieval. This class allows absolute and relative time knowledge in a form which is suitable for many practical applications, where relative temporal information is only occasionally needed. The architecture of such a system is given, and it is shown that the efficient retrieval mechanisms for absolute-time-stamped databases may be adapted to time-limited databases.

Received November 1993; revised June 1994

1. INTRODUCTION

The incorporation of time into conceptual database systems has been an active area for research over the past decade. Several approaches to the problem have used a relational model which includes temporal attributes. For example, Jones and Mason (1980), and Sarda (1990), address starting and finishing times as attributes for each whole tuple which define its *validity*; Gadia (1988) proposes a system in which each attribute value is stamped with a *temporal element* which is a finite union of point-based intervals; and Clifford (1985, 1987) includes functions from valid times to attribute values for *time-varying attributes*. Additionally, it has been recognised that there may be more temporal attributes required for temporal database management systems. In fact, in Ben-Zvi's (1982) time relational model, five implicit time attributes have been addressed: *effective-time-start* and *effective-time-stop* are, respectively, the left and right end points of the time interval for the existence of the real-world phenomenon being modelled; *registration-time-start* is the time at which the effective-time-start was stored; *registration-time-stop* is the time at which the effective-time-stop was stored; and *deletion-time* records the time when erroneously entered tuples are logically deleted. Subsequently, Snodgrass and Ahn (1986, 1987), McKenzie (1991), and Clifford and Isakowitz (1993) have proposed systems with temporal attributes, which are semantically similar to Ben-Zvi's, e.g. in Snodgrass (1987), four implicit times, *valid-from*, *valid-to*, *transaction-start* and *transaction-stop*, are addressed. Common to these later approaches is that the attributes encode both valid time and transaction time for each tuple. In the glossary of temporal database concepts, Jensen *et al.* (1992) have given definitions of valid time and transaction time, as below:

- *Valid time*. The valid time of a fact is the time when the fact is true in the modelled reality.
- *Transaction time*. A database fact is stored in a database at some point in time, and after it is stored, it may be retrieved. The transaction time of a database fact is *the time when the fact is stored in* the database. Transaction times are consistent with the serialisation order of the transactions. Transaction time values cannot be after the current time.

Hence, in this paper, we take the transaction time of a database fact as the time over which the fact is (was) taken as part of the current state of the database. It is interesting to note that the time, or the 'point in time', at which a database fact is stored in the database is sometimes termed its *recording time*: in fact, recording time can be taken as the *transaction-start* for the transaction time attribute in Snodgrass' (1987) system.

Some systems, such as those of Jones and Mason (1980), of McKenzie (1991) and Snodgrass (1987), take the tuple as the fundamental transaction unit, so that whenever anything changes within the tuple, the whole tuple is regarded as being renewed. Other systems, such as that of Gadia (1988), take individual attribute values as the fundamental transaction unit. Ahn (1986) has shown that these two views are entirely equivalent, and Ling and Bell (1992) have pointed out that the appropriate transaction units for any application are to do with practical questions of storage and retrieval efficiency. We shall adopt the tuple view in this paper, since it makes for clearer presentation.

Most temporal database systems require absolute time values for the temporal attributes. However, there may be temporal knowledge about a valid/transaction interval even though precise starting and finishing times are unknown. For example, we may know that event A happened *before* event B, without knowing the absolute times when they actually started/finished. Relative temporal knowledge such as this is typically derived from humans, where absolute times are rarely remembered, but relative temporal relationships are often remembered. In James Allen's (1981, 1983) interval based temporal system, 13 primitive relationships between intervals are defined including *equal*, *before*, *meets*, *overlaps*, *starts*, *during*, *finishes* and six inverse relationships. These relationships provide a means to represent relative temporal knowledge, in a form comprehensible to humans.

In order to be capable of representing not only absolute, but also relative and imprecise temporal information, some new temporal database systems have been introduced, examples are those of Chaudhuri (1988) and of Koubarakis (1993). Chaudhuri's graph model is proposed as a tool in identifying generic temporal queries and in describing the process of deduction of temporal relationships. The model represents time elements as nodes and temporal relationships between time elements as arcs of the graph. Queries to the graph are processed by propagation of temporal relationships along arcs of the graph, according to Allen's (1983) transitivity table. The complexity of this process is a major problem of the deductive method and Chaudhuri discusses how heuristics may be used in some cases to solve this problem. However, the model is restricted to a subset of binary temporal relationships only; it cannot represent mixed relative time and absolute time duration information. Additionally, it cannot represent disjunctive constraints and does not address issues such as transaction time/valid time explicitly. Consistency checking, which has been a problematic question in most temporal systems, is not addressed in Chaudhuri's model.

Koubarákis's (1993) system is based on relation-like representations which can contain variables constrained by the formulas of his temporal theory. The underlying time theory for the system is point-based: points are identified with the rationals while intervals are considered as pairs of points. Hence, the time basis for this system is similar to the point-based constraint network of Dechter *et al.* (1991), where temporal predicates over intervals must be expressed in terms of the temporal order over the interval end-points. However, Vilain (1982, 1986) and Van Beck (1989, 1992) have examined the complexity issues relating to interval/point algebra, showing that the computational complexity of the constraint evaluation algorithm may be prohibitive for practical systems. Additionally, as Allen (1983) and Long (1989) have pointed out, modelling intervals by taking their end-points can lead to problems: the annoying question of whether end-points are in the interval or not must be addressed, seemingly without any satisfactory solution. If intervals are all closed then adjacent intervals have endpoints in common; hence, when adjacent intervals correspond to the validity times of truth and falsehood of some property, there will be a point at which the property is both truth and false. Similarly, if intervals are all open, there will be points at which the truth or falsity of a property will be undefined. The solution in which intervals are taken as semi-open, e.g. see the definition of intervals in Maiocchi's (1992) *TSOS*, so that they sit conveniently next to one another, seems arbitrary and unsatisfactory, since for some cases there is nothing to choose between whether an interval should be left-open/right-closed or left-closed/right-open. This problem has been highlighted in Allen (1983) and provides motivation for the use of an interval based system to handle human temporal information in a natural way.

The objective of this paper is to present a temporal database architecture, which allows purely relative temporal knowledge, by using Allen's 13 temporal predicates as part of its user interface. Also, we shall show how such an architecture may be integrated into an absolute-time-stamped database system, to provide a practical solution to the problem of retrieval efficiency. The fundamental theory of Allen's system is interval-based and the exposition of this paper could follow Allen's time theory exactly. However, we prefer to utilize an extension of Allen and Hayes' (1994) interval-based axiomatization, given previously by the authors in Ma and Knight (1994), as the underlying temporal basis for the system. This axiomatization of time has been developed from Allen's interval based theory by means of including points on the same footing as intervals. A critical examination of Allen's time theory has been given by Galton (1990), which shows the problems resulting from the exclusion of the concept of points. For example, consider the motion of a ball thrown vertically into the air. According to classical physics, there is a point p at which the ball is stationary in the air (neither going-up nor going-down). Intuitively, the valid time for the fact that the ball is 'going-up' or 'going-down' is an interval with a positive duration, while the valid time for the fact that the ball is 'stationary' is just a time point. Hence, for general treatment, in the architecture presented in this paper transaction time and valid time will take values from a set of time elements, which may be primitive intervals or primitive points. (The term 'time element' should not be confused with the similar term 'temporal element', which is often used elsewhere to denote a finite union of intervals.)

The need for temporal databases of the kind discussed in this paper occurs in applications where one might possess some temporal knowledge in relative form. We give an illustrative example of such a system, from the medical field, to assist the presentation of the ideas. Consider a patient, who we shall call Lee, attending a clinic: On his first attendance, Lee described that several weeks before, he felt some stomach pain and took a pain-killer (drug a) for some days. However this did no good, and he visited the doctor. A doctor (Dr Major) cancelled drug a, and prescribed drug b and drug c for 3 days, asking Lee to come to the clinic after finishing the drugs. Unfortunately, Dr Major forgot to write down the prescription of drug c on Lee's medical record. The next time Lee attended the clinic, he was examined by another doctor (Dr Long). Since Lee's condition had changed for the better and his medical record wrongly indicated that the treatment had been just drug b, Dr Long simply prescribed drug b for another 3 days. Thereupon Lee's condition deteriorated. Lee came back to the clinic just after finishing drug b. This time, Dr Major discovered the mistake, and corrected Lee's medical record, In addition, Dr Major prescribed drug b and c for another 7 days.

This exmaple involves very little data in the traditional sense, but involves many different temporal references. The data essentially describes a treatment (the list of drugs prescribed). The valid time of the treatment may be characterized as an interval with a positive length. The exact starting-point and finishing-point of each of these periods are not always specified. However, some temporal facts are known relating to the periods. For example, the period of treatment with drug a comes immediately before treatment with b and c, although we do not know its start time, or duration.

Another interesting aspect of temporal databases illustrated by the example is the importance of the transaction time to inference. According to the view of the database at the time when Lee was examined by Dr Long, the database wrongly indicated that Lee's condition changed for the better because of the effect of just drug b, although actually drug b and drug c together were taken. However, it is not appropriate to delete this wrong information from the database, since if we do so we cannot at a later date tell why Dr Long did not prescribe drug c at that time. Instead of over-writing the wrong information, the corrected version must be added to the database, but with its own transaction time.

The main problem involved with relative databases is that of temporal inference since for relative temporal data one often needs deeper reasoning to infer facts. For example, if we have a database with the following facts in terms of absolute time:

Interval i_1:	starting time	= 8.00
	finishing time	= 9.00
Interval i_4	starting time	= 11.15
	finishing time	= 11.30

then we need only to retrieve i_1 and i_4 to find directly that i_1 is before i_4. However, if we only have the following relative information:

Interval i_1 before Interval i_2, Interval i_2 before Interval i_3, Interval i_3 before Interval i_4,

then we need to retrieve facts from the database other than those relating to i_1 and i_4 (i.e. i_2 and i_3), and then we must deduce that i_1 is before i_4. This problem is dealt with in Section 4 below, where a general mechanism for query evaluation is presented. The mechanism is based on the refutation principle, with a general consistency checker for interval and point based system given previously by the authors in Knight and Ma (1993).

In Section 2 of the paper, we outline briefly the underlying time axiomization used. Section 3 presents the conceptual structure for a system involving relative times and gives relational schema. In Section 4 we give the general form for temporal queries and show how they may be resolved in terms of the inference mechanism. Examples of queries to the database are discussed in Section 5. In Section 6 we examine the problem encountered when absolute temporal duration information is added to the database: although absolute-time-stamped systems provide good retrieval mechanisms, mixed systems with some relative time and some absolute durations are intractable. As a proposal, we also introduce a class of databases, termed time-limited databases, which allow absolute time and relative time information in a form which is suitable for many practical applications. It is shown that absolute-time-stamping retrieval methods may be adapted for time-limited databases.

The time theory used in this paper allows both time points and time intervals as primitive, so as to handle human temporal information in a natural way, consistent with Allen's approach. This is different from the approach of Koubarakis, which in fact reduces to a point-based constraint network problem. It is also more comprehensive than the approach of Chaudhuri since it allows mixed relative and absolute time. Also a general retrieval mechanism is given based on refutation by means of a consistency checker, supporting any conventional data conditions, and both conjunctive and disjunctive temporal constraints.

2. THE UNDERLYING TEMPORAL BASIS

In Ma and Knight (1994), we have proposed an extension of Allen and Haye's (1989) interval based time theory. The extended theory takes both intervals and points as primitive time elements on the same footing. A single primitive temporal predicate, i.e. '*meets*', over time elements is formally axiomatized. This time theory characterises a very general temporal frame which may be linear or non-linear, dense or discrete, etc. In what follows, we give a brief outline of the main features of the general time axiomatization.

We use T to denote a nonempty set of time-elements and d to denote a function from T to R_0^+, the set of non-negative real numbers. A time-element, t, is called a (time) interval if d(t) >; 0, otherwise, t is called a (time) point. According to this classification, the set of time-elements, T, may be expressed as $T = I \cup P$, where I is the set of intervals, and P is the set of points.

The primitive order over time elements is a temporal predicate termed '*meets*', which is axiomatized by the following axioms:

⟨A1⟩ $\forall t_1, t_2, t_3, t_4 \in T(meets(t_1, t_2) \wedge meets(t_1, t_3) \wedge meets(t_4, t_2) \Rightarrow meets(t_4, t_3))$

⟨A2⟩ $\forall t \in T \exists t', t'' \in T(meets(t', t) \wedge meets(t, t''))$

⟨A3⟩ $\forall t_1, t_2 \in T(\exists t', t'' \in T(meets(t', t_1) \wedge meets(t_1, t'') \wedge meets(t', t_2) \wedge meets(t_2, t'')) \Rightarrow t_1 = t_2)$

⟨A4⟩ $\forall t_1, t_2 \in T(meets(t_1, t_2) \Rightarrow \exists t \in T \forall t', t'' \in T(meets(t', t_1) \wedge meets(t_2, t'') \Rightarrow meets(t', t) \wedge meets(t, t'')))$

⟨A5⟩ $\forall t_1, t_2 \in T(meets(t_1, t_2) \Rightarrow t_1 \in I \vee t_2 \in I)$

⟨A6⟩ $\forall t_1, t_2 \in T(meets(t_1, t_2) \Rightarrow d(t_1 \oplus t_2) = d(t_1) + d(t_2))$

Axiom ⟨A1⟩ states that the 'place' where two time elements meet is unique. ⟨A2⟩ preserves that every time element has at least one neighbouring time element preceding it and another succeeding it. ⟨A3⟩ simply says that there is a unique time element between any two meeting places. ⟨A4⟩ states that if two meeting places are separated by a sequence of time elements, then there is a time element which connects these two meeting places. Hence, by axiom ⟨A3⟩, for any two adjacent time elements t_1 and t_2, the ordered union of t_1 and t_2 may be written as a time interval, $t = t_1 \oplus t_2$. Axiom ⟨A5⟩ is based on the intuition that points will not meet other points, i.e. between any two time points there is a time interval. ⟨A6⟩ ensures that the operation, '$\oplus$' which combines adjacent time elements, is consistent with the function d, which we shall call the *duration assignment* over T.

In terms of the primitive predicate '*meets*', the complete set of possible temporal relationships over time elements may be classified in terms of the following four groups of predicates:

Point–Point:
equal, before, after
which relates points to other points:

Interval–Interval:
equal, before, meets, overlaps, starts, during, finishes, finished_by, contains, started_by, overlapped_by, met_by, after,
which relate intervals to intervals:

Point–Interval:
before, meets, starts, during, finishes, met_by, after,
which relate points to intervals:

Interval–Point:
before, meets, finished_by, contains, started_by, met_by, after,
which relate intervals to points.

The definition of these temporal predicates in terms of *meets* is straightforward (see Ma and Knight, 1994). For example, *before* can be defined in the theory by:

$$before(t', t'') \Leftrightarrow \exists t(meets(t', t) \wedge meets(t, t''))$$

Note that the above definition for the predicate $before(t', t'')$ accommodates the case that more than one intermediate time elements, $t_1, \ldots, t_m$, stand between t' and t'', since by axioms ⟨A3⟩ and ⟨A4⟩, we can write $t_1 \oplus \ldots \oplus t_m = t$.

In this way, we take the temporal part of a database, consisting of the data of *meets* knowledge alone, to express the relative temporal knowledge. We wish, however, to require the user interface to the system to support the full set of temporal predicates.

Ordinarily, in computer systems we have to store information as a finite set, and the semantics of any database of time elements will usually assume a well-order at some fundamental level. Hence, the computer-based temporal system may be viewed as a model of the theory, in the form of a finite set of temporal facts.

In fact, from a temporal frame axiomatized by axioms ⟨A1⟩–⟨A6⟩ we may form a non-empty finite set $E \subset T = I \cup P$, such that:

1. $E = \{t_1, t_2, \ldots, t_m\}$;
2. $meets(t_i, t_{i+1}), i = 1, 2, \ldots, m-1$;
3. $meets(t_i, t_{i+1}) \Rightarrow t_i \in I \vee t_{i+1} \in I$.

These theorems precisely characterize a finite series, E, of time elements, which is *similar* to an *initial segment* of the set of natural numbers with an *immediate successor* relationship. Additionally, it is easy to see that the limitation of axioms ⟨A4⟩, ⟨A5⟩ and ⟨A6⟩ onto E, well define the *closure* of E under '$\oplus$'/'$+$', the binary operation of combining adjacent time elements, and the corresponding addition of duration defined by the axiomization (see Knight and Ma, 1992, 1993).

Theoretically, the closure of E includes E and all the time elements which can be formed by means of '$\oplus$'/'$+$'. However, in an application neither the complete closure nor the fundamental set E may be entirely known. A database may express temporal knoweldge that is incomplete in several ways. For example, the database may contain knowledge of duration assignments for only some of its elements, and may have only a partial knowledge about the '*meets*' relationship for the corresponding time elements. We term such a temporal system a finite time network.

An intuitive graphical representation of a time network has been introduced previously by the authors in two different forms for cases with, and without, duration knowledge, respectively in Knight and Ma (1992, 1993). Figure 1 shows an example of such a representation in the case where no duration knowledge is addressed. In this graphical representation, time

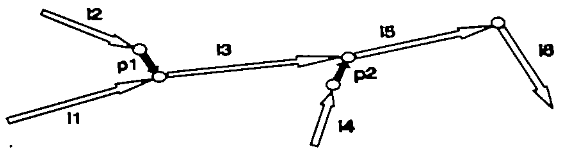

FIGURE 1.

elements are denoted as arcs of the graph, and the '*meets*' relationship is expressed by means of the node structure in the directed graph, where $meets(t_i, t_j)$ is represented by t_i being in-arc and t_j being out-arc to a common node (For the sake of clearer presentation, in Figure 1 points are represented as single barred arcs, and intervals as double barred arcs). The graph of Figure 1 represents the following *meets* relationships:

$$meets(i_1, i_3), meets(i_2, p_1), meets(p_1, i_3), meets(i_3, i_5), meets(i_4, p_2), meets(p_2, i_5), meets(i_5, i_6)$$

3. A RELATIONAL MODEL FOR TEMPORAL DATABASES

In this section, we present the relational model for a temporal database. Following the conventional concepts, a non-temporal relation R with non-temporal attributes, $A_1, \ldots, A_n$ defines a non-temporal relation scheme, denoted as $R(A_1, \ldots, A_2)$, where each attribute A_i takes values from its domain, D_i, a set of data. A non-temporal relation is a subset of the Cartesian product of one or more domains. Conventionally, a relation is envisioned as a table of data values, where the rows of such a table are termed tuples, and values of an attribute associated with column i of the relation are taken from domain D_i. In the model presented here, temporal reference of the system is made by assigning two time-elements to each nontemporal tuple, which denote the valid time and transaction time respectively. Knowledge about the temporal order over time elements is represented by a table of *meets* relationships over the corresponding time elements. Hence, corresponding to the non-temporal relation scheme $R(A_1, \ldots, A_n)$, we can define the temporal relation scheme as:

$$TR(T_{\text{transaction}}, A_1, \ldots, A_n, T_{\text{valid}}),$$

together with a *meets* table:

$$meets(T_{\text{first-argument}}, T_{\text{second-argument}}),$$

where instances of $T_{\text{transaction}}$, T_{valid}, $T_{\text{first-argument}}$ and $T_{\text{second-argument}}$ are all taken from the set of time element which can be intervals or points.

In what follows, we shall illustrate the ideas in terms of the simple medical record example outlined in the introduction.

For the example, if we take non-temporal attributes:

Patient, **Prescriber**, **Drug** and **Status**,

then the database may be represented by the following schema:

medical-record ($T_{\text{transaction}}$, Patient, Prescriber, Drug, Status, T_{valid}),

$$meets(T_{\text{first-argument}}, T_{\text{second-argument}}),$$

Correspondingly, we may illustrate the medical-record representing Lee's history as in Table 1 (Medical Record).

In this example there are five validity intervals shown:

- i_{v1} the time when Lee first felt pain.
- i_{v2} the time when Lee was treated by means of drug a, administered by himself.
- i_{v3} the time when Lee was treated by drug b and drug c, prescribed by Dr Major.
- i_{v4} the time when Lee was treated by drug b, prescribed by Dr Long.
- i_{v5} the time when Lee was again treated by drug b and drug c, prescribed by Dr Major.

There are also three transaction times shown:

- i_{t1} transaction interval following data entry after first appointment with Dr Major.
- i_{t2} transaction interval following data entry after first appointment with Dr Long.
- i_{t3} transaction interval following data entry after second appointment with Dr Major.

The graphical representation of the corresponding time network is given in Figure 2.

The record shows three transaction intervals for the validity interval i_{v3}. The third (interval i_{t3}) corrects the second (interval i_{t2}) which incorrectly records that Lee was taking drug b alone. However, the reason for Dr Long's prescription of drug b alone for interval i_{v4} (i.e. that he thought drug b alone had led to improvement over i_{v3}) may still be inferred from the database. This is because the transaction time allows us to retrieve the state of the database at any time in the past. Also, these transaction intervals 'are consistent with the serialisation

TABLE 1. Medical record

$T_{TRANSACTION}$	*Patient*	*Prescriber*	*Drug*	*Status*	T_{VALID}
i_{t1}	Lee	null	null	pain	i_{v1}
i_{t1}	Lee	Lee	a	worse	i_{v2}
i_{t1}	Lee	Major	b	null	i_{v3}
i_{t2}	Lee	null	null	pain	i_{v1}
i_{t2}	Lee	Lee	a	worse	i_{v2}
i_{t2}	Lee	Major	b	better	i_{v3}
i_{t2}	Lee	Long	b	null	i_{v4}
i_{t3}	Lee	null	null	pain	i_{v1}
i_{t3}	Lee	Lee	a	worse	i_{v2}
i_{t3}	Lee	Major	b&c	better	i_{v3}
i_{t3}	Lee	Long	b	worse	i_{v4}
i_{t3}	Lee	Major	b&c	null	i_{v5}

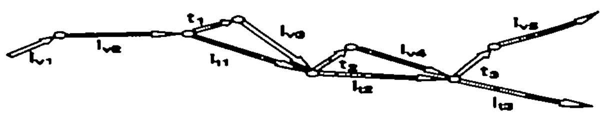

FIGURE 2.

order of the transactions' (see Jensen *et al.*'s definition given in Section 1). Temporal queries depending on both validity time and transaction time will be discussed in Section 4 below.

The temporal part of the database is represented by the eight time elements: i_{v1}, i_{v2}, i_{v3}, i_{v4}, i_{v5}, i_{t1}, i_{t2}, and i_{t3}, along with the *meets* predicate imposed over them. As a table, it may be represented as:

$$\begin{gathered} meets(i_{v1}, i_{v2}),\ meets(i_{v2}, t_1),\ meets(i_{v2}, i_{t1}), \\ meets(t_1, i_{v3}),\ meets(i_{v3}, t_2),\ meets(i_{v3}, i_{t2}), \\ meets(i_{t1}, t_2),\ meets(i_{t1}, i_{t2}),\ meets(t_2, i_{v4}), \\ meets(i_{v4}, t_3),\ meets(i_{v4}, i_{t3}),\ meets(i_{t2}, t_3), \\ meets(i_{t2}, i_{t3}),\ meets(t_3, i_{v5}) \end{gathered}$$

where t_1, t_2, and t_3 represent delays between the doctor prescribing the drug and the patient beginning to take it. These delays are generally unknown in duration, e.g. $meets(i_{v2}, t_1)$ and $meets(t_1, i_{v3})$ express the fact that the valid time interval i_{v3} is 'after' i_{v2}, by reason of the delay interval, t_1, standing between them.

4. THE INFERENCE MECHANISM

In this section, we consider the inference mechanism for the system. For queries to the database, we wish to support all the possible temporal predicates over time elements. For example, 'show me all facts *before* time t_1, or *after* time t_2', or 'Is it true that fact A holds *during* time i and *before* time t?'. All these temporal queries can be characterized in a general query form. The general method for query evaluation depends upon a consistency checker for the database in terms of the necessary and sufficient condition for the consistency of the corresponding time network (see Knight and Ma, 1993).

For temporal queries, temporal constraints will be added as additional querying conditions. Although the relationships between time elements are characterized in terms of the single predicate, '*meets*', we should allow a more general form of query. This should allow temporal constraints, in terms of all the possible temporal predicates presented in Section 2, over both transaction times and valid times, and any other given reference time elements.

Formally, we may specify a query as a general expression of the tuple relational calculus (see Elmasri and Nabathe, 1994) in terms of the following form:

$$\{x_1 \cdot A_1, x_2 \cdot A_2, \ldots, x_n \cdot A_n | \\ \text{COND}(x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m})\}$$

where $x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+m}$ are tuple variables, each A_i, is an attribute of the temporal relation to which x_i belongs, and COND is a **condition** or well-formed **formula** (wff) of the tuple relational calculus (see Elmasri and Navathe, 1994). In the temporal calculus proposed here, a non-temporal wff is made up from atoms in the conventional way, where an atom can be one of the following forms:

1. $TR(x_i)$, where TR is a temporal relation name and x_i is a tuple variable.
2. $x_i \cdot A\, op\, c\, x_j \cdot B$, or $x_i \cdot A\ c\ op\ x_j \cdot B$, where *op* is one of the binary comparison operators in the set $\{=, \neq, <, \leqslant, >, \geqslant\}$, x_i and x_j are tuple variables, A is an attribute of the relation to which x_i belongs, B is an attribute of the relation to which x_j belongs, and c is a constant value.

However, to address the temporal constraints over time elements, we need to extend the wffs to include temporal atoms of the form $r(x_i \cdot A, x_j \cdot B)$, where r is one of the binary temporal predicates classified in Section 2, A and B are temporal attributes (from the domain of time elements), of the relation(s) to which x_i and x_j belong, respectively.

The key problem in evaluating a query with temporal conditions is to test whether for a given pair of time elements, t_1 and t_2, a constraint $r(t_1, t_2)$ is satisfied. In principle, we can do this by showing that, $\neg r(t_1, t_2)$, the converse constraint to $r(t_1, t_2)$, is *inconsistent* with the time network, by means of a consistency checker given in Knight and Ma (1993). It has been shown that the temporal predicates given in Section 2 are mutually exclusive to each other (see Knight and Ma, 1993; Ma and Knight, 1994). So that, for example, if p is a point and i an interval, we know that precisely one of the temporal predicates in the set:

$$R = \{before, meets, starts, during, finishes, met\_by, after\}$$

must apply for p and i. Hence for $r_0 \in R$:

$$\neg r_0(p, i) \Leftrightarrow r_1(p, i) \vee r_2(p, i) \vee \ldots \vee r_6(p, i)$$

where $\{r_1, r_2, \ldots r_6\} \cup \{r_0\} = R$. Hence, to show $r_0(p, i)$ we simply show that $r'(p, i)$ is inconsistent for $r' = r_1, r_2, \ldots, r_6$, i.e. r' can be any one of the temporal relationships between point p and interval i, other than r_0. For instance, we may show *before*(p,i) by means of showing that *meets*(p,i), *starts*(p,i), *during*(p,i), *finishes*(p,i), *met-by*(p,i) and *after*(p,i) are all inconsistent with the database.

The general treatment of the temporal constraints can then be handled by conventional logical operations over the results of individual constraint evaluations.

It is interesting to note that improvement may be made to the evaluation of the individual converse constraint $\neg r(k_1, k_2)$. The method given above equates $\neg r(k_1, k_2)$ with the disjunction of $r'(k_1, k_2)$ for $r' \in R \backslash \{r\}$. This leads to several different consistency checks. However, for each temporal predicate the number of tests may be reduced considerably (to a maximum of 2) by expressing $\neg r$ directly in terms of *meets*. For instance:

$$\neg before(k_1, k_2) \Leftrightarrow meets(k_0, k_2), \wedge\, meets(k_0, k) \\ \wedge\, meets(k_1, k_n) \wedge meets(k, k_n) \vee meets(k_1, k_2)$$

Hence, for example, the six tests for the constraint $\neg r_0(p, i)$ illuminated above may be reduced to two.

5. EXAMPLES OF RETRIEVAL

In this section we give some examples of temporal predicates used as temporal constraints. For each query we give the tuple calculus specification, and a table representing the result in the case of the medical record example given in Section 3. Here, since only a single relation is addressed, these examples seem to be quite simple. However, by means of the general expression of the temporal tuple relational calculus proposed in the above section, complex cases which involve more than one relation can be addressed in a straightforward way.

Query 1 Retrieve Lee's medical history, as known before the second appointment with Dr Major

$$\begin{aligned} \{x|\ & \text{MEDICAL-RECORD}(x) && \text{AND} \\ & x.\text{PATIENT} = \text{'Lee'} && \text{AND} \\ & (\textit{before}(x.T_{\text{TRANSACTION}}, i_{t3}) && \text{OR} \\ & \textit{meets}(x.T_{\text{TRANSACTION}}, i_{t3}) &&)\} \end{aligned}$$

Result:

$T_{TRANSACTION}$	*Patient*	*Prescriber*	*Drug*	*Status*	T_{VALID}
i_{t1}	Lee	null	null	pain	i_{v1}
i_{t1}	Lee	Lee	a	worse	i_{v2}
i_{t1}	Lee	Major	b	null	i_{v3}
i_{t2}	Lee	null	null	pain	i_{v1}
i_{t2}	Lee	Lee	a	worse	i_{v2}
i_{t2}	Lee	Major	b	better	i_{v3}
i_{t2}	Lee	Long	b	null	i_{v4}

Query 2 Is it true that Lee took drug a after i_{t1}?

$$\begin{aligned} \{x|\ & \text{MEDICAL-RECORD}(x) && \text{AND} \\ & x.\text{PATIENT} = \text{'Lee'} && \text{AND} \\ & x.\text{DRUG} = a && \text{AND} \\ & x.T_{\text{TRANSACTION}} = i_{t3}) && \text{AND} \\ & \textit{after}(x.T_{\text{VALID}} = i_{t1}) && \} \end{aligned}$$

Result: Null

Query 3 Was it known over i_{t2} that Lee got better using drug b alone?

$$\begin{aligned} \{x.\text{STATUS}, x.T_{\text{VALID}} | & \text{MEDICAL-RECORD}(x) && \text{AND} \\ & x.\text{PATIENT} = \text{'Lee'} && \text{AND} \\ & x.\text{DRUG} = b && \text{AND} \\ & x.T_{\text{TRANSACTION}} = i_{t2} && \} \end{aligned}$$

Result:

Status	T_{VALID}
Better	i_{v3}
Null	i_{v4}

This query shows exactly the state of Dr Long's knowledge according to the transaction time i_{t2}. It shows his belief concerning the past interval, i_{v3}, and his prescription for the coming interval, i_{v4}.

Of course, the tables given in the examples cannot present all of the information required by a user of the system, since the temporal relationship of the terms, i_{v1}, $i_{v2}, \ldots, i_{t1}, i_{t2}, \ldots, t_1, t_2, \ldots$, needs the graph of Figure 2 as part of the user interface. As a matter of presentation, it may be convenient to display more than just the given time elements in order to allow the user to relate them to other special reference time elements which exist in the network. Particularly, if absolute-time-stamped elements are added to the network, as in the next section, these may be displayed as reference elements.

6. INTEGRATING ABSOLUTE AND RELATIVE TEMPORAL INFORMATION

Section 4 outlines an inference mechanism for retrieval subject to temporal constraints, from the purely relative temporal database. This inference mechanism is based on a consistency checking algorithm given in Knight and Ma (1993). The necessary and sufficient condition for a purely relative time network without any duration constraints to be consistent, proved in Knight and Ma (1993), are two-fold. The first is the requirement that points do not meet points. This condition makes sense on the intuitive level, where we expect points to be separated by intervals. The second is best given in terms of the graphical representation. If we first modify the graph of the time network to remove points from the graph, by simply treating the point arcs as nodes, then the second condition states that the modified graph must be acyclic. If both of these conditions are satisfied than the database is consistent, otherwise it is inconsistent.

The inclusion of absolute temporal information into the representation may be handled by the addition of known numerical durations to weight the arcs of the graph. In this case, an alternative consistency checker, proved in Knight and Ma (1992), may be employed. This general consistency checking algorithm, addressing mixed absolute and relative time, involves a search for cycles and the construction of a numerical constraint for each cycle. The existence of a solution to this set of constraints implies the consistency of the system. Hence, the consistency checker for a random set of known durations is in fact a linear programming problem. Much work has gone into algorithms for linear programming. As Cormen *et al.* (1989) examine in their textbook, many linear programs can in practice be solved very quickly by means of the so-called *simplex algorithm*. However, with some carefully contrived inputs, the simplex method can lead to exponential complexity. General linear programs can be solved in polynomial time by either *Karmarkar's algorithm* which in practice is often competitive with the simplex method, or the *allipsoid algorithm* which, however, runs slowly in practice (see Cormen *et al.*, 1989).

However, the retrieval situation for an absolute-time-stamped system is computationally efficient, since all start- and end-times are known, and hashing or tree-search schemes may be based on known time points. In order to reduce the computational overhead of consistency checking for the case supporting absolute duration information, in what follows, we introduce a class of databases, termed *time-limited* databases. Databases of this type allow both absolute time value and relative time information in a form suitable for many practical applications. The main idea is that, in many application, the relative temporal knowledge may be very limited, while most data is stamped with absolute time values. The graphical structure of such a system is that of a single absolute-time-stamped chain, C, with occasional attached groups of purely relative time knowledge, Q_i, $i = 1, 2, \ldots, g$, which we shall term 'relative sub-networks'. An assumption is made that each relative sub-network is *time-limited*, i.e. two definite times with absolute values may be determined that are earlier and later respectively that the sub-network. In this case, each relative sub-network may be spanned by an absolute-time-stamped interval which starts and finished simultaneously with it. We shall show that retrievals may be made first from the absolute-time-stamped chain including spanning intervals and then, if necessary, by relative inference over the union of two relative sub-networks.

In an absolute-time-stamped temporal system the time elements consist of a sequence of time points with absolute time values (reals or rationals), separated by time intervals. We shall term this set of elements the *absolute-time-stamped chain*, and any ordered union of elements in an absolute-time-stamped chain as an *absolute-time-stamped element* (especially, any element in an absolute-time-stamped chain is an absolute-time-stamped element).

Definition Let C be a special time network consisting of a sequence of time elements of which all points are stamped with absolute time values. C is called an *absolute-time-stamped chain* if:

$$\begin{aligned}\forall c \in C(\ & c \in I \Rightarrow \exists p_1, p_2 \in C \cap P(meets(p_1, c) \\ & \wedge\, meets(c, p_2)) \\ & \wedge\, c \in P \Rightarrow \exists i \in C \cap I(meets(i, c) \\ & \vee\, meets(c, i))\ )\end{aligned}$$

where I is the set of intervals, and P is the set of points.

We now consider a time network graph G containing a single absolute-time-stamped chain C. If we let Q be the subgraph of G consisting of non-absolute-time-stamped elements in G, we may present G as $G = C \sqcup Q$, where $C \sqcup Q$ represents the graph union of C and Q. Additionally, let Q be decomposed into subgraphs $Q_1, Q_2, \ldots Q_g$, such that:

- Q_i is a connected sub-graph of G.
- Q_i and Q_j are only connected through nodes in C.
- For each element t in Q_i there exists a directed path from a node in C to t, and a directed path from t to another node in C,

where $i, j = 1, 2, \ldots, g$.

The first two of these properties define what is meant by a relative sub-network, Q_i. Such a sub-network is a connected set of non-absolute-time-stamped arcs, which is isolated from other such sets in the sense that they are only connected in G by means of their connection to the absolute-time-stamped chain itself. Practically it is often the case that the temporal relationships between several linked events are known but these these events are unconnected with other sets of events. For example, in the illustration of Section 5, the events are connected through a single patient. Any connection with other patient events is made through absolute value times, i.e. through the absolute-time-stamped chain.

It should be noted that all graphs $G = C \sqcup Q$ may be decomposed into Q_i satisfying the first two properties, the only question being the size of the relative subgraphs. In the extreme case, there is the trivial decomposition with just Q itself satisfying the properties.

The third property is the reasonable practical assumption that each relative sub-network may be time-limited in some way. That is, for any set of linked temporal events, some absolute time bounds, however wide, may be established. For example, in the medical illustration it may not be known exactly when the events took place, but the month or year, will surely be known.

We shll term a relative sub-graph Q_i which satisfied all the above three properties *time-limited*. Figure 3 shows a time network containing an absolute-time-stamped chain C:

$$C = p_1, i_1, p_2, i_2, p_3, i_3, p_4, i_4, p_5$$

with the following two time-limited sets:

$$Q_1 = \{i_5, t_1, t_2, i_6\}$$

$$Q_2 = \{i_7, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, i_8\}.$$

Note that in the graphical representation in Figure 3, while points are still denoted as single barred arcs and intervals as double barred arcs, time elements which are not known to be points or intervals are represented as shadowed double barred arcs.

We term the decomposition:

$$G = C \sqcup Q_1 \sqcup Q_2 \sqcup \ldots \sqcup Q_g$$

a time-limited decompostion of G, where $Q_1, Q_2, \ldots, Q_g$ are time-limited relative subgraphs.

We now need to define a spanning element S(t) for each time element t in G. First, for each relative subgraph Q_i, we define Pre(Q_i) as the latest point on C from which there is a directed path to each element of Q_i, that is, Pre(Q_i) is the latest absolute-time point which is earlier than all elements in Q_i. Similarly, we define Suc(Q_i) as the earliest absolute-time point on C to which there is a directed path from each element of Q_i. For

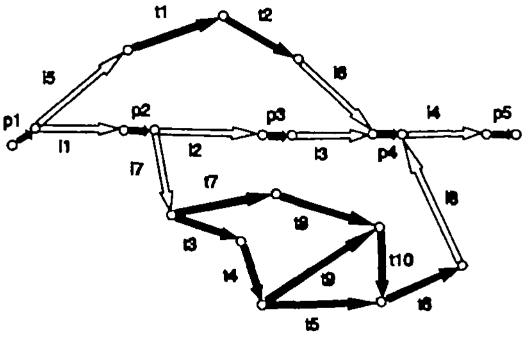

FIGURE 3.

example, in Figure 3, $\mathrm{Pre}(Q_i) = p_1$, $\mathrm{Suc}(Q_1) = p_4$; $\mathrm{Pre}(Q_2) = p_2$, $\mathrm{Suc}(Q_2) = p_5$. Then we can define the spanning element S(t) for t in the following way. If t is a time element on the absolute-time-stamped chain C, then we define S(t) as t itself. If t belongs to a relative subgraph Q_i, then we define S(t) as the absolute-time-stamped interval which '*meets*' $\mathrm{Suc}(Q_i)$, and is '*met-by*' $\mathrm{Pre}(Q_i)$.

We are interested in the evaluation of constraints such as: *before*(t_1,t_2), *during*(t_1,t_2), ..., etc., from a network A. For clarity, we introduce the network name as a third argument to these temporal predicates, so that *before*(t_1,t_2,A) means that t_1 is *before* t_2 in network A.

It is now straightforward to demonstrate the following three theorems, which may form the basis of retrieval mechanisms for a time-limited database. Theorems (T1) and (T2) show that we can evaluate $r(t_1, t_2, G)$ by first testing *before*(S(t_1),S(t_2),C) and *after*(S(t_1),S(t_2),C) over the absolute-time-stamped network C, using the absolute-time-stamping retrieval algorithms (since S(t_1) and S(t_2) are both stamped with absolute time values). If these predicates are not satisfied, then we need the full refutation mechanism to evaluate the other predicates. However, theorem (T3) shows that we only need to do this over the union of C with the two time-limited sets $Q(t_1)$ and $Q(t_2)$, where Q(t) denotes the relative subnetwork containing t.

T1. $before(S(t_1), S(t_2), C) \Rightarrow before(t_1, t_2, G)$
T2. $after(S(t_1), S(t_2), C) \Rightarrow after(t_1, t_2, G)$
T3. $\neg(before(S(t_1), S(t_2), C) \vee after(S(t_1), S(t_2), C))$
$\wedge r(t_1, t_2, C \sqcup Q(t_1) \sqcup Q(t_2) \Rightarrow r(t_1, t_2, G)$

where r is any one of the possible temporal predicates between time elements t_1 and t_2.

The 'relative' retrieval algorithm is thus reduced to a search for cycles over the graph $C \sqcup Q(t_1) \sqcup Q(t_2)$. The complexity of this algorithm is dependent on the size of $Q(t_1)$, and $Q(t_2)$ only, since there can be no cycles involving the part of C that is '*before*' the earlier one of $\mathrm{Pre}(Q(t_1))$ and $\mathrm{Pre}(Q(t_2))$, or '*after*' the later one of $\mathrm{Suc}(Q(t_1))$ and $\mathrm{Suc}(Q(t_2))$.

7. CONCLUSION

In this paper we have shown that it is possible to base a database systems on relative time. The system uses an extension of Allen's interval-based time theory, and allows the use of Allen's relative temporal predicates for data input and retrieval. A general retrieval mechanism has been proposed and examples of retrievals using relative temporal constraints have been presented.

The paper also demonstrates the integration of relative time into an absolute-time-stamped database, for a useful class of temporal data, where all time elements may be given some absolute-time-stamped limits. Such a system is referred to as a time-limited database. Theorems which form the basis of efficient retrieval mechanism for a time-limited database have been presented.

ACKNOWLEDGEMENTS

We would like to express our thanks to the referees for their helpful comments and suggestions.

REFERENCES

Ahn, I. (1986) Towards an implementation of database management systems with temporal support, IEEE CH2261-6/86/0000/0374$01.00, pp. 374–381.
Allen, J. F. (1981) An interval-based representation of temporal knowledge. In *Proc. 7th Int. Joint Conf. on AI*, 221–226.
Allen, J. F. (1983) Maintaining knowledge about temporal intervals. *Commun. ACM*, **26**, 832–843.
Allen, J. F. and Hayes, P. J. (1989) Moments and points in an interval-based temporal-based logic, *Comput. Intell. (Canada)*, **5**, 225–238.
Beek, P. V. (1989) Approximation algorithms for temporal reasoning. In *Proc. 11th IJCAI*, pp. 1291–1296.
Beek, P. V. (1992) Reasoning about qualitative temporal information. *Artif. Intell.*, **58**, 297–326.
Ben-Zvi, J. (1982) *The Time Relational Model*. PhD dissertation, Computer Science Department, University of California, Los Angeles.
Chaudhuri, S. (1988) Temporal relationships in databases. In *Proc. 14th VLDB Conf.*, pp. 160–170, Los Angeles, CA.
Clifford, J. (1985) Toward an algebra of historical relational databases. In *Proc. ACM-SIGMOD Int. Conf. on Management of Data*, pp. 247–265. Austin, TX.
Clifford, J. (1987) *Historical Databases — It's About Time*. Graduate School of Business Administration, New York University, NY.
Clifford, J. and Isakowitz, T. (1993) On the semantics of transaction time and valid time in bitemporal databases. In *Proc. Int. Workshop on an Infrastructure for temporal database*, pp. 125–138. Arlington, TX.
Cormen, T. H., Leiserson, C.E. and Rivest, R. L. (1989) *Introduction to Algorithms, The MIT Electrical Engineering and Computer Science Series*. MIT Press, Cambridge, MA.
Dechter, R., Meiri, I. and Pearl, J. (1991) Temporal Constraint Networks, *Artif. Intell.*, **49**, 61–95.
Elmasri, R. and Navathe, S. B. (1994) *Fundamentals of Database Systems*, 2nd edn., Benjamin/Cummings, New York.
Gadia, S. K. (1988) A homogeneous relational model and query languages for temporal databases. *ACS Trans Database Syst.*, **13**, 18–448.
Galton, A. (1990) A critical examination of Allen's theory of action and time. *Artif. Intell.*, **42**, 159–188.
Jensen, C. S., Clifford, J., Gadia, S. K., Segev, A. and Snodgrass, R. T. (1992) A glossary of temporal database concepts. *SIGMOD RECORD*, **21**, 35–43.

Jones, S. and Mason, P. (1980) Handling the time dimension in databases. In *Proc. Int. Conf. on Databases*, pp. 66–83. British Computer Society.

Knight, B. and Ma, J. (1992) A general temporal model supporting duration reasoning. *AI Commun. J.*, **5**, 75–84.

Knight, B. and Ma, J. (1993) An extended temporal system based on points and intervals. *Inform. Syst.*, **18** 111–120.

Koubarakis, M. (1993) Representation and querying in temporal databases: the power of temporal constrains. In *Proc. IEEE Conf. on Data Engineering*, pp. 327–334.

Ling, D. H. O. and Bell, D. A. (1992) Modelling and managing time in database systems, *Comp. J.*, **35**, 332–341.

Long, D. (1989) A review of temporal logics. *Knowledge Eng. Rev.*, **4**, 141–162.

Ma, J. and Knight, B. (1994) A general temporal theory. *Comp. J.*, **37**, 114–123.

Maiocchi, R. (1992) Automatic deduction of temporal information. *ACM Trans. Database Syst.*, **4**, 647–688.

McKenzie, L. E. and Snodgrass, R. T. (1991) *Supporting Valid Time in an Historical Relational Algebra.* Techical Report TR91-15. Department of Computer Science, University of Arizona,.

Sarda, N. (1990) Algebra and query language for a historical data model. *Computing*, **33**, 11–18.

Snodgrass, R. T. and Ahn, I. (1986) Temporal databases, *IEEE*, 0018-9162/86/0900-0035, pp. 35–42.

Snodgrass, R. T. (1987) The temporal query language Tquel. *ACM Trans Database Syst.*, **12**, 247–298.

Vilain, M. B. (1982) A system for reasoning about time. *Proc. 1st AAAI*, pp. 197–201. Pittsburgh, PA.

Vilain, M. B. and Kautz, H. (1986) Constraint propagation algorithms for temporal reasoning, *Proc. 5th AAAI*, pp 377–382.