SEMANTICS OF TEMPORAL QUERIES AND TEMPORAL DATA

Carole D. Hafner
College of Computer Science
Northeastern University
Boston, MA 02115

Abstract

This paper analyzes the requirements for adding a temporal reasoning component to a natural language database query system, and proposes a computational model that satisfies those requirements. A preliminary implementation in Prolog is used to generate examples of the model's capabilities.

1. Introduction

A major area of weakness in natural language (NL) interfaces is the lack of ability to understand and answer queries involving time. Although there is growing recognition of the importance of temporal semantics among database theoreticians (see, for example, Codd [6], Anderson [2], Clifford and Warren [4], Snodgrass [15]), existing database management systems offer little or no support for the manipulation of time data. Furthermore (as we will see in the next Section), there is no consensus among researchers about how such capabilities should work. Thus, the developer of a NL interface who wants to support time-related queries cannot look to an underlying DBMS for help.

Currently available NL systems such as Intellect [8] have not attempted to support temporal queries, except in a trivial sense. In Intellect, users can ask to retrieve date attributes (e.g., "When was Smith hired?") or enter restrictions based on the value of a date attribute (e.g., "List the employees hired after Jan 1, 1984"); but more complex questions, such as "How long has it been since Smith received a raise?" or "What projects did Jones work on last January?", are not understood. This is a serious practical limitation, since the intended users of NL systems are executives and other professionals who will require more sophisticated temporal capabilities.

This report describes a model of temporal reasoning that is designed to be incorporated into a NL query system. We assume that a syntactic component could be developed to translate explicit temporal references in English (e.g., "two years ago") into logical representations, and restrict our attention to the conceptual framework (including both knowledge structures and rules of inference) underlying such representations. Section 2 analyzes the requirements that the temporal model must satisfy: first describing some of the issues that arise in trying to model time in a computer, then defining four basic semantic relationships that are expressed by time attributes in databases, and finally analyzing the capabilites required to interpret a variety of temporal queries. Based on this analysis, a computational model is described that satisfies many of the requirements for understanding and answering time-related database queries, and examples are presented that illustrate the model's capabilities.

2. Modeling Temporal Knowledge

Modeling time, despite its obvious importance, has proved an elusive goal for artificial intelligence (AI). One of the first formal proposals for representing time-dependent knowledge in AI systems was the "situation calculus" described by McCarthy and Hayes [11]. That proposal created a paradigm for temporal reasoning based on the notion of an infinite collection of states, each representing a single instant of time. Propositions are defined as being either true or false in a particular state, and predicates such as "before (s1, s2)" can be defined to order the states temporally. This approach was used by Bruce [3] in modeling the meaning of tensed verb phrases in English, and it has been refined and extended by McDermott [13].

State space models describe time as being similar to the real number line, with branches for alternative pasts and hypothetical futures. Although this approach is intuitively appealing, there are many unsolved problems from both the logical and the linguistic points of view. A few of the current problems in temporal semantics are very briefly described below:

a. Non-monotonic reasoning. In a system for automated reasoning, conclusions are drawn on the basis of current facts. When a fact that was true becomes false at a later time, conclusions that were based on that fact may (or may not) have to be revised. This problem, which is viewed by many as "the" current issue in common sense reasoning, has been studied extensively by Doyle [7], Moore [14], and McDermott [13], and continues to occupy the attention of John McCarthy [12].

b. Representation of intervals and processes. Another problem for temporal logic is the representation of events that occur over intervals of time. Allen [1] points out that even events which seem to be instantaneous, such as a light coming on, cause problems for the state space model, since at the instant that this event occurs it is impossible to say that either "the light is on" or "the light is not on" is true. As a result, Allen chooses a representation of time that uses intervals as the primitive objects instead of instantaneous states.

c. Temporal distance. Neither the state space model nor the interval model offers a convincing notion of temporal distance. Yet, the ability of a system to understand how long an event took or how much time separated two events is an integral part of temporal reasoning.

d. Periodicity of time. There are many periodic events that affect the way we think and talk about time - such as day and night, the days of the week, etc. McDermott [13] shows how his temporal logic can describe periodic events, and Anderson [2] includes a representation of periodic data in her model of temporal database semantics. However, reasoning about periodic time structures is still a relatively unexplored issue.

e. Vagueness and uncertainty. People are able to reason about events whose temporal parameters are not known exactly - in fact, almost all temporal descriptions incorporate some vagueness. The most direct treatment of this phenomenon was a system by Kahn and Gorry [9], which attached a "fuzz factor" to temporal descriptions. However, Kahn and Gorry recognized that this approach was very crude and more sophisticated techniques were needed.

f. Complex event structures. The situation calculus is not easily adapted to descriptions of complex acts such as running as race, simultaneous events such as hiding something from someone by standing in front of it while that person is in the room (an example dis- cussed by Allen [1]), or "non-events" such as waiting.

g. Metaphorical time descriptions. In naturally occuring NL dialogues, time descriptions are frequently metaphoric. Lakoff and Johnson [10] have shown that at least three metaphors are used to describe time in English: time as a path, time as a resource, and time as a moving object. AI models have yet to adequately deal with any of these metaphors.

Considering all of these complex issues (and there are others not mentioned here), it is not surprising that general temporal capabilities are not found in applied AI systems. However, in the domain of NL query systems, it may be possible to ignore many of these problems and still produce a useful system. The reason for this is, in the world models of computer databases, most of the complexity and ambiguity has already been "modeled out". Furthermore, current NL interfaces only work well on a subclass of databases: those that conform to a simple entity-attribute-relationship model of reality.

The research described in this paper has focused on the design of a temporal component for a NL database query system. This has led to a model of time that corresponds to the structure of time attributes in databases: i.e., a domain of discrete units representing intervals of equal length. (Whether these units are seconds, minutes, days, or years may vary from one database to another.) The description of the model presented in Section 3 assumes that the basic temporal units are days, in order to make the model more intuitively meaningful; however, the model can be easily adapted to time units of other sizes.

2.1 Analysis of Time Attributes in Databases

The primary role of time information in databases is to record the fact that a specific event occurred at a specific time. (It is also possible to represent times in the future, when an event is scheduled to occur, e.g., the date when a lease is due to expire.) Having said this, there are still different ways in which time attributes may be semantically related to the entities in the database, and these require different inferences to be made in translating NL queries into the framework of the data model. The following categories of time attributes are frequently observed in "real world" databases:

1. Time attributes describing individuals
2. Time of a "transaction"
3. Time when an attribute or relationship changed
4. The time of transition from one stage of a process to the next.

The first two categories are quite straightforward. Time attributes of individuals appear in "entity" relations, as shown in Figure 1a; they describe the occurrence of a significant event for each individual, such as an employee's date of birth or the date when the employee was hired. This type of temporal attribute has a unique (and usually unchanging) value for each individual.

The term "transaction" is used here to describe an event (usually involving several types of entities) that does not change the status of the participants, other than the fact that they participated in the event. For example, the date of each treatment (an X-ray, a therapy session, or a surgical procedure) given to a patient by a doctor would be recorded in a medical records database, as shown in Figure 1b.

Attributes in the third category record the time at which some other attribute or relationship changed. Databases containing this type of information are called "historical databases", in contrast to the more traditional "operational" databases, which only record a "snapshot" of the current state of the world. The salary history and student records databases shown in

1a. Time Attributes Decribing Individuals
Employee Database

Emp_ID	Name	Address	Birth_Date	Hire_Date

1b. Time of a Transaction
Medical Records Database

Patient	Doctor	Procedure	Date

1c. Time When an Attribute or Relationship Changed
Salary History Database

Emp_ID	Salary	Date

Student Records Database

Student_ID	Subject	Degree	Date

1d. Time of a Process Transition
Publication Database

Doc_ID	Author	Sub_Date	Disp_Date	Rev_Date	Pub_Date

Figure 1. Examples of Temporal Attributes

1. Which doctors performed operations on June 15, 1983?
2. How many people received PhD's in Math last month?
3. What percent of the employees got raises in the 4th quarter of 1984?
4. Did any authors have more than one paper waiting for publication on Jan 1?
5. How much was Jones making in September of 1984?
6. How long has Green worked here?
7. What was the average review time for papers submitted in 1983?
8. Which patients received operations on each day last week?
9. How many Ph. D's were granted to women during each of the past 10 years?

Figure 2. Examples of Temporal Queries

Figure 1c are examples of this type of temporal data. Within this category, we must recognize a further distinction between exclusive attributes such as salary and non-exclusive attributes such as degree. When a new salary is entered for an employee, the previous salary is no longer valid; but when a new degree is entered, it is added to the individual's previous degrees.

The last category of temporal data is used to record fixed sequences of events that occur in various actiivies. For example, the publication database of Figure 1d records the life-cycle stages of papers submitted to a scientific journal: the date the paper was received, the date it was accepted (or rejected), the date the revised version was received, and the date that is it scheduled to be published. We can view this sequence as a process with several stages ("under review", "being revised", "awaiting publication"), where each temporal attribute represents the time of transition from one stage to the next.

2.2. Analysis of Temporal Queries

This section considers four types of queries involving temporal data, and briefly outlines the capabilites that a temporal knowledge model must have in order to understand and answer queries of each type.

Queries 1-3 in Figure 2 are examples of time restriction queries, which retrieve data about individuals or events whose dates fall into a particular interval of time. Current database systems already support time restrictions, such as Query 1, that use simple, absolute time references. Queries such as (2), which use relative time references, and (3) which refer to intervals not directly represented in the database, require a more elaborate model of time structures than current systems provide. The time domain model described in Section 3.1 can support queries of this type.

The second type of query asks about the state-of-the-world on a given date (Query 4) or during an interval of time (Query 5). Understanding and answering these queries requires rules for deducing the situation at a given time, as a result of the occurrence (or non-occurrence) of events before that time. For example, Query 5 asks about Jones' salary in September of 1978; however, there may not be an entry for Jones in the salary history file during that period. The system must know that the correct salary can be retrieved from the most recent salary change entered for Jones before that date. Section 3.2 describes an event model that can represent this type of knowledge.

Another type of query asks about the length of time that a situation has existed (Query 6), or about the duration of one stage of a process (Query 7). These queries require functions to compute and compare lengths of time, and rules for deducing the starting and ending times of states-of-the-world based on the events that trigger them. Section 3.3 shows how the proposed temporal model handles this type of query.

The last type of query is the periodic query, which asks for objects to be grouped according to one or more attributes. High-level data languages and current NL interfaces are generally able to handle this type of request when it refers directly to the value of an attribute (e.g., Query 8), but not when it requires information to be grouped by time period, as in Query 9. To anwer periodic queries requires a formal representation for descriptions such as "each of the past 5 years"; the "periodic descriptors" defined in Section 3.1 satisfy this requirement.

3. A Temporal Reasoning Model for Databases

In this section, a temporal reasoning model is proposed that can interpret the types of queries described in Section 2.2. The model, which is expressed as a collection of predicates and rules written in Prolog [5], consists of the following components:

1. A time domain model for representing units (days), intervals, lengths of time, calendar structures, and a variety of relative time descriptions.

2. An event model for representing and reasoning about the temporal relationships among events, situations, and processes in the application domain.

3.1 Time Domain Model

The basic structures of the time domain model are days, intervals, calendars, and periodic descriptors. The days (D, D1, D2 . .) form a totally ordered set, with a "distance" function representing the number of days between two days. The distance function satisfies the laws of addition, i.e.:

1) distance (D1 , D2) = 0 <--> D1 = D
2) distance (D1 , D2) = - distance (D2 , D1)
3) distance (D1 , D2) + distance (D2 , D3) = distance (D1 , D3)

Intervals (I, I1, I2 . .) are ordered pairs of days [Ds , De] such that distance (Ds , De) >= 0. If an interval I = [Ds , De] then:

4) start (I) = Ds
5) end (I) = De
6) length (I) = distance (start (I) , end (I)) + 1
7) during (D, I) = "true" <-->
distance (start(I) , D) >= 0 and
distance (D , end(I)) >= 0

Other temporal relations, such as "before (D1, D2)", "after(D1, D2)", and interval relations such as those described by Allen [1], can be defined using the "distance" function in an equally straightforward manner. Also included in the model is a function "today" of no arguments whose value is always the current day.

Formulas (1-7) are repeated below in Prolog notation:

```
1) distance(D1,D2,0) :- D1 = D2.
2) distance(D1, D2, Y) :- distance(D2, D1, X) , Y = -X.
3) distance(D1, D3, Z) :- distance(D1, D2, X),
                          distance(D2, D3, Y), Z=X+Y.
4) start(I,Ds).
5) end(I,De).
6) length(I , Y) :- distance(start(I), end(I), X) ,
   Y = X+1.
7) during (D, I) :- distance(start(I), D , X) , X >= 0 ,
                    distance (D, end(I), Y), Y >= 0.
```

Examples of some natural language concepts:

```
n_days_ago (N , D) :- today(DT) , distance(D, DT, N).
n_days_from_now (N , D) :-
                    today(DT) , distance (DT, D, N).
the_past_n_days (N , I) :-
                    today(DT), end(I,DT), length(I,N).
the_next_n_days (N , I) :-
                    today(DT), start(I,DT), length(I,N).
the_day_before_yesterday (D) :- n_days_ago(2 , D).
```

A calendar is a structure for representing sequences of intervals, such as weeks, months, and years. We will consider only "complete" calendars, which cover all the days, although it would be useful to define incomplete calendars to represent concepts such as "work weeks" which exclude some days. A calendar (CAL) is a totally ordered set of interval descriptors called "calendar elements" (CE, CE1, CE2 . .). The following predicates are defined for calendars:

distcal(CAL, CE1, CE2, N). This is like the distance function for days. It is true if CE2 is N calendar elements after CE1. For example: distcal(year, 1983, 1985, 2) is true.

getcal(CAL, CE, I). This predicate is true if I is the interval represented by the calendar element CE. For example: getcal(year, 1983, [jan011983 , dec311983]) is true.

incal(CAL, D, CE , N). This predicate is true if D is the Nth day of calendar element CE. It is used to map a day into the calendar element to which it belongs. For example: incal(month, jan121983 , [jan,1983] , 12]) is true.

Calendars satisfy the well-formedness rules that we would expect; for example, for each day D and each calendar CAL, there is at most one (for complete calendars, exactly one) calendar element CE and positive integer N such that incal (CAL, D, CE, N) is true. Also, if CE1 is before CE2, then each day in CE1 is before each day in CE2. And, for complete calendars, if CE1 immediately precedes CE2, then the last day of CE1 immediately precedes the first day of CE2.

Although the representation of calendar elements is arbitrary, we have chosen conventions that are both meaningful to the programmer and useful to the implementation. The simplest calendars are those such as "year", containing named elements that occur only once. Years are simply represented as atoms corresponding to their names. Cyclic calendars are those that cycle within another calendar, such as the calendars for "month" and "quarter". The elements of these calendars are represented as 2-tuples, for example: distcal(month, [dec,1983] , [jan,1984] , 1) is true. The calendar for weeks presents the most difficult problem for the time domain model, since weeks are not usually identified by name. We have defined the week calendar so that all weeks begin on Sunday and end on Saturday, with each element of the calendar equal to the interval it represents. While this is not an entirely satisfactory solution, it allows a number of useful "weekly" computations.

More examples of natural language concepts:

```
from_ce1_to_ce2(CAL, CE1, CE2, I) :-
   /* from January, 1983 to July, 1985 */
   getcal(CAL, CE1, I1) , getcal(CAL, CE2, I2) ,
   start(I1 , S), end (I2 , E) , start(I , S), end(I , E).
```

```
n_cal_elts_ago(CAL, N, D) :-
   /* three weeks ago */
    today(DT) , incal(CAL, DT, CE1, X) ,
    distcal(CAL, CE2, CE1, N) , incal(CAL, D, CE2, X).
```

The last structure in the time domain model is the periodic descriptor (PD), used for representing expressions such as "each of the past 5 years" or "each month in 1983". Periodic descriptors are 3-tuples consisting of a calendar (to define the size of each period), a starting element from that calendar (to define the first period), and either an ending element from that calendar (to define the last period) or an integer (to define how many periods are to be computed). Periodic descriptors can run either forward or backward in time, as shown by the following example:

```
each_of_the_past_n_cal_elts(CAL,N, PD) :-
  PD = [CAL, CEP, M], today(DT), incal(CAL, DT, CET, _) ,
   distcal(CAL, CEP, CET, 1), M is -N.
```

To interpret a query containing a periodic descriptor, the NL interface must first expand the structure into a list of intervals (this must wait until execution time in order to ensure the right value for "today") and then perform an iterative execution of the query , restricting it in turn to each interval in the list.

3.2. Event Model

In the event model, each type of event is represented by a unique predicate, as are the situations and process stages that are signified by events. For example, the event of a person receiving a degree is represented by: awarded(Person, Subject, Degree). The situation of having the degree is represented by: holds(Person, Subject, Degree). While the "awarded" predicate is true only on the date the degree was received, the "holds" predicate is true on that date and all future dates. Below we define a straightforward approach to representing this type of knowledge.

Five basic temporal predicates are introduced to relate events and situations of the application model to elements of the time domain model.

timeof(E, D) - succeeds whenever an event that matches E occcurs in the database with a time that matches D. This is the basic assertion that relates events to their times of occurrence.

nextof(E, T, D) - asserts that D is the next time of occurrence of event E after time T.

```
nextof(E, T, D) :- timeof(E, D) , before(T, D) ,
        not (timeof (E, X), before (T, X), before (X, D).
```

startof(S, D) - defines the time when a situation or process stage begins to be true, based on the occurrence of the event that triggers it. Rules of this sort are part of the knowledge base of each application, for example:

```
startof (holds(Person, Subject, Degree), Date) :-
  timeof (awarded( Person, Subject, Degree) , Date).
```

endof(S, D) - defines the time when a situation ceases to be true. For an exclusive attribute such as salary(jones, 40000), the "end-of" a situation is the "next-of" the same kind of event that triggered the situation (i.e., when Jones gets a new salary then salary(jones,40000) is no longer true). For other kinds of situations, a specific "termination" is required to signify the ending; e.g., a publication ceases to be "under review" when it is accepted.

trueon(S, D) - succeeds if situation S is true at time D. Given the predicates described above, the definition of trueon might be:

```
trueon(S, D) :- startof (S, A), not (after(A,D)),
                not (endof(S, B), before (B, D)).
```

This rule asserts that situation S is true at time D if S began at a time before (or equal to) D, and did not end at a time before D.

3.3. An Example Query

We can now bring the two parts of the model together to describe how a temporal query is represented and interpreted using the predicates and rules defined above. We will consider the following query, addressed to the salary history database:

> Which employees are making at least twice as much now as they made 5 years ago.

For experimental purposes, we have defined our database as a collection of Prolog facts, as proposed by Warren[16] ; thus, the database can be queried directly in Prolog. We have also defined the "days", which are the primitive elements of the time domain model, to have names such as jan011982 or jul041776; these names appear in the database as the values of temporal attributes, as shown below:

```
salhistory(jones, 30000, jan011983).
salhistory(smith, 42000, jan151983).
```

Each of the event-model predicates described in the previous section has also been created, with "newsalary(EMPID, SAL)" substituted for E and "makes(EMPID, SAL)" substituted for S. For example:

```
timeof(newsalary(EMPID, SAL), D) :-
                        salhistory(EMPID, SAL, D).
startof(makes(EMPID, SAL), D) :-
                timeof(newsalary(EMPID, SAL), D).
endof(makes(EMPID, SAL), D2) :-
        timeof(newsalary(EMPID,SAL), D),
        nextof(newsalary(EMPID,SAL2), D, D2),
        SAL =< SAL2
trueon(makes(EMPID, SAL), D) :-
                        startof(makes(EMPID,SAL), D.
trueon(makes(EMPID, SAL), D) :-
    startof(makes(EMPID, SAL), D1), before(D1,D),
    not (endof(makes(EMPID, SAL), D2), before(D2, D)).
```

We can now express the sample query in Prolog:

```
result(EMPID, SAL, OLDSAL) :-
  today(DT) ,
  trueon(makes(EMPID, SAL), DT),
  n_cal_elts_ago(year, 5, DFYA) ,
  trueon(makes(EMPID, OLDSAL), DYFA),
  SAL >= 2 * OLDSAL.
```

This Prolog rule would be the desired output of the linguistic component of a NL query system. Paraphrased in English, it says: retrieve all triples of employee id, current salary, and old salary, such that the employee makes the current salary today, the employee made the old salary five years ago, and the current salary is greater than or equal to two times the old salary. If we expand all of the Prolog rules that would be invoked in answering this query, leaving only database access commands, arithmetic tests, and computations of the "distance" function, the complete translation would be:

```
result(EMPID, SAL, OLDSAL) :-
   today(DT) ,
   salhistory(EMPID, SAL, D),
   distance(D, DT, X1),
   X1 >= 0 ,
   not(salhistory (EMPID, SAL2, D2),
       distance(D, D2, X2),
       X2 > 0 ,
       distance (D2, DT, X3),
       X3 >= 0 ,
       SAL ¬= SAL2),
   incal(year, DT, YR1, Y) ,
   distcal (year, YR1, YFYA, -5),
   incal(year, DFYA, YFYA, Y) ,
   salhistory (EMPID , OLDSAL, D3) ,
   distance (D3, DYFA, X4),
   X4 >= 0,
   not(salhistory(EMPID, OLDSAL2, D4),
       distance(D3, D4, X5),
       X4 > 0,
       distance(D4, DYFA, X5),
       X5 >= 0,
       OLDSAL1 ¬= OLDSAL2).
```

4. Conclusions

This paper has proposed a temporal reasoning model based on the use of time attributes in databases, and the types of queries that we would expect in "real-world" applications. The model includes constructs for representing events, situations, and processes that are similar to those found in other temporal reasoning models. It also addresses some issues of particular importance for NL query systems, which are not addressed by other recent work in temporal reasoning, including:

1. Representing the time between two points, and the lengths of intervals.
2. Representing weeks, months, years, and other standard calendar structures.
3. Representing information relative to "today" , "this month", etc.
4. Representing periodic time descriptions.

The use of discrete, calendar-like structures as a basis for representing time in a computer is a simplification that is compatible with the discrete representation of information in databases. Hopefully, this simplification will make it easier to program the model and to integrate it into a state-of-the-art NL query system.

5. References

1. Allen, J. F., "Towards a General Theory of Action and Time. Artificial Intelligence, Vol. 23, No. 2 (1984) 123-154.
2. Anderson, T. L., "Modeling Time at the Conceptual Level." In P. Scheuermann, ed., Improving Database Usability and Responsiveness, pp 273-297. Jerusalem: Academic Press, 1982.
3. Bruce, B., "A Model for Temporal Reference and its Application in a Question Answering System." Artificial Intelligence, Vol 3, No. 1 (1972), 1-25.
4. Clifford, J. and D. S. Warren, "Formal Semantics for Time in Databases." ACM TODS, Vol. 8, No. 2 (1983) 214-254.
5. Clocksin, W.F. and C. S. Mellish, Programming in Prolog. Berlin: Springer-Verlag, 1981.
6. Codd, E. F., "Extending the Database Relational Model to Capture More Meaning." ACM TODS, Vol. 4, No. 4 (1979) 397-434.
7. Doyle, J., "A Truth Maintenance System." Artificial Intelligence, Vol. 12, No. 3 (1979), 231-272.
8. INTELLECT Reference Manual, INTELLECT LEX Utility Reference, Program Offerings LY20-9083-0 and LY20-9082-0, IBM Corp., 1983.
9. Kahn, K. and G. A. Gorry, "Mechanizing Temporal Knowledge." Artificial Intelligence, Vol 9 (1977), 87-108.
10. Lakoff, G., and M. Johnson, Metaphors We Live By. The University of Chicago Press, Chicago ILL (1980).
11. McCarthy, J. and P. J. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence." In B. Meltzer and D. Michie, eds., Machine Intelligence 4. American Elsevier, New York (1969).
12. McCarthy, J., "What is Common Sense?" Presidential Address at the National Conference on Artificial Intelligence (AAAI-84), Austin, TX (1984).
13. McDermott, D., "A Temporal Logic for Reasoning About Processes and Plans." Cognitive Science, Vol. 6 (1982) 101-155.
14. Moore, R. C., "Semantical Considerations on Nonmonotonic Logic." Artificial Intelligence, Vol. 25, No. 1 (1983), 75-94.
15. Snodgrass, R., "The Temporal Query Language TQuel." In Proc. 3rd ACM SIGMOD Symp. on Principles of Database Systems, Waterloo, ONT (1984).
16. Warren, D. H. D., "Efficient Processing of Interactive Relational Database Queries Expressed in Logic." In Proc. 7th Conf. on Very Large Databases, pp. 272-281. IEEE Computer Society (1981).