An Interval-Based Representation of Temporal Knowledge

James F. Allen
Department of Computer Science
The University of Rochester
Rochester, NY 14627

Abstract

This paper describes a method for maintaining the relationships between temporal intervals in a hierarchical manner using constraint propagation techniques. The representation includes a notion of the present moment (i.e., "now"), and allows one to represent intervals that may extend indefinitely into the past/future.

This research was supported in part by the National Science Foundation under Grant Number IST-80-12418, and in part by the Office of Naval Research under Grant Number N00014-80-C-0197.

I. Introduction

This paper describes a method for representing and reasoning about temporal knowledge. While most work in artificial intelligence would appear to require some reasoning about time, this problem is typically avoided and delegated to "future research." In order to make further progress in the areas of problem solving and natural language understanding, these problems cannot be ignored any longer.

The techniques for modeling time that have been developed in problem-solving systems seem too weak for our purposes. The predominant temporal models in these systems are those using state variables and state spaces. In this world, time is represented as a sequence of instantaneous time slices. Each time slice is described by the set of facts that hold at that time. This has been a successful model in problem solving systems that deal with a single agent operating in a simple discrete world (such as the blocks world). Unfortunately, current problem solving work is attempting to deal with multiple agents, each acting not only on what is true now, but on what each expects to be true in the future. To treat these issues adequately, one needs a more flexible model of time. For further discussion of these issues, see [McDermott, 1978] and [Hayes, 1979].

The techniques for modeling time developed in natural language understanding systems are typically cruder than those in problem solving systems. In part, this is because many efforts have been directed at question answering systems in a static world (with no conception of time), or in domains small enough so that simple before-after chains of relations are computationally feasible.

Recently, there has been a growing interest in systems that use problem solving techniques as an essential part of the natural language understanding task (e.g. [Allen, 1979; Wilensky, 1978; Grosz, 1979]). This work appears to have great promise, but cannot continue much further without a general model of temporal relations.

Let us consider what general characteristics such a temporal model should have. The following seem particularly relevant:

- The model should allow significant imprecision of scale. Many temporal relations are strictly relative and have little relation to some absolute sense of time.

- The model should allow uncertainty. On many occasions, the exact relationship between two time intervals is not known, but some constraints are known about how they could be related.

- The model should have a strong sense of *now*. This must be done in a manner that allows the present moment (now) to change without requiring major changes to the knowledge base.

- The model should allow a sense of *persistence*. It should facilitate default inferences of the form, "if P is true now, it will remain true until I notice otherwise."

With a few notable exceptions, there has been little work done on computer representations of time. The work that there is, however, contains many interesting ideas which will be freely used in this paper. Kahn and Gorry [1977] suggest that the temporal aspects of knowledge can be separated out for individual treatment. They describe a system that is a time specialist; it maintains temporal relations and provides the rest of the system with the tools to test, retrieve, add, and delete temporal information. The time specialist knows nothing else about the knowledge representation. A similar approach is taken here: the model knows only about temporal relations, and provides the tools for manipulating this knowledge. This is one part of a larger effort to design and implement a temporal logic which can describe and reason about temporally qualified propositions, events, actions, and plans [Allen, in progress]. Thus, the long-term goal of this work is similar to that in Hayes [1979] and McDermott [1981].

The model described here differs from the model of Kahn and Gorry, as well as that of Bruce [1972], in that it emphasizes the aspects of temporal reasoning that are not concerned with date lines. In particular, it allows relative temporal relationships to be maintained in a highly structured fashion. Furthermore, it includes a notion of the present time (i.e., "now"), which is maintained in a manner that does not require knowledge of the exact present time, and that does not involve extensive modifications to the knowledge base every time "now" is updated.

The system described here has been implemented in LISP on a VAX 11/780. In particular, the constraint propagation with reference intervals and the manipulations of the "now" are completed and fully tested. All examples in Sections II through V inclusive have actually been run on the system.

II. Time Points vs. Time Intervals

The references to temporal relations in English are often both implicit and vague. In particular, the majority of temporal references are implicitly introduced by tense and by the description of how events are related to other events. Thus we have

"We found the letter while John was away."
"We found the letter after we made the decision."

These sentences are introducing temporal relations between the times (intervals) at which the events occurred. For example, the temporal connective "while" indicates that the time when the find event occurred is during the time when John was away. The tense indicates that John being away occurred in the past (i.e., before now).

Although some events appear to be instantaneous (e.g. one might argue that the event "finding the letter" is instantaneous), the majority of events described take place over a time interval. Thus, our representation is designed to maintain these interval relations conveniently and concisely.

The most apparent scheme for representing intervals seems to be modeling the endpoints. Thus, if we assume a model consisting of a fully ordered set of points of time, then an interval is an ordered pair of points with the first point less than the second. We then can define the following relations between intervals (for any interval t, the lesser endpoint is denoted by t-; the greater by t+):

Interval Relation	Equivalent Relations on Endpoints
t < s	t+ < s-
t = s	(t- = s-) & (t+ = s+)
t *overlaps* s	(t- < s-) & (t+ > s-) & (t+ < s+)
t *meets* s	t+ = s-
t *during* s	((t- > s-) & (t+ =< s+)) or ((t- >= s-) & (t+ < s+))

Figure 1: Interval Relation Defined by Endpoints

Thus we can now map our intervals and interval relations onto the simpler time point representation.

I am not going to implement intervals using this scheme. The main reason is that the representation is too uniform and does not facilitate structuring the knowledge in a way which is convenient for typical temporal reasoning tasks. The central issue here is the importance of the *during* relation. A major part of our temporal knowledge is of the form

"event E' occurred during event E"

and that our knowledge of the *during* relation allows a highly structured representation of time. In particular, a key fact used in testing whether some condition P holds during an interval t is that if t is during an interval T, and P holds during T, then P holds during t. Thus the during relation can be used to define a hierarchy of intervals in which properties can be inherited.

Furthermore, such a *during* hierarchy allows reasoning processes to be localized so that irrelevant facts are never considered. For instance, if one is concerned with what is true "today," one need consider only those intervals that are *during* "today," or above "today" in the *during* hierarchy. If a fact is indexed by an interval wholly contained by an interval representing "yesterday," then it cannot affect what is true now.

Whenever we need to refer to points, they will be explicitly related to an interval. Thus, we can talk about the beginning of an interval t (t-), the end of t (t+), or any arbitrary point during t. The only difference between such points and a standard interval is that points cannot participate in the *overlaps* relation, and the *meets* relation indicates that the point is an endpoint of the interval.

III. Maintaining Temporal Relations: The Simple View

We saw above five relations that can hold between intervals. Considering the inverses of these relations, there are a total of nine ways in which an ordered pair of intervals can be related. These are shown in Figure 2.

Relation	Symbol	Symbol for Inverse	Pictoral Example
X *before* Y	<	>	XXX YYY
X *equal* Y	=	=	XXX YYY
X *meets* Y	m	mi	XXXYYY
X *overlaps* Y	o	oi	XXX YYY
X *during* Y	d	di	XXX YYYYYY

Figure 2: The Nine Possible Relationships

These relationships between intervals are maintained in a network where the nodes represent individual intervals. Each arc is labelled to indicate the relationship between the two intervals represented by its nodes. In cases where there is uncertainty about the relationship, all possible cases are entered on the arc. Note that since the nine possible relationships are mutually exclusive, there is no ambiguity in this notation. Figure 3 contains some examples. Throughout, let N_i be the node representing interval i. Notice that the third set of conditions captures the one sense of the notion of disjoint intervals.

Relation	Network Representation
1. i *during* j	N_i -- (d) --> N_j
2. i *during* j or i *before* j or j *during* i	N_i -- (< d di) --> N_j
3. (i < j) or (i > j)	N_i -- (< >) --> N_j

Figure 3: Representing Knowledge of Temporal Relations in a Network

Throughout this paper, both the above notations will be used for the sake of readability. In general, if the arc asserts more than one possible relationship, the network form will be used, and in the case where only one relationship is possible, the relation form will be used.

In this simple view, it is assumed that the network always maintains complete information about how the intervals in it could be related. Thus when a new fact is entered, its full consequences must be computed. This can be done by computing the transitive closure on the temporal relations as follows: the new fact adds a constraint about how the two intervals could be related, which may in turn introduce new constraints between other intervals because of the transitivity behavior of the temporal relationships. For instance, if the fact that i is *during* j is added, and j is *before* k, then it is inferred that i must be *before* k. This new fact is then added to the network in an identical fashion, possibly introducing further constraints on the relationship between other intervals. A few of the transitivity relations are summarized in Figure 4. The full transitivity table can be found in [Allen, 1981].

As an example of using this table: if A *after* B, and B *during* C, then follow the 2nd row (">") to the 3rd column ("d"), to obtain A - (> oi d mi) --> C.

A nice property of this transitive closure algorithm is that it only continues to operate as long as it is inferring new relations between intervals. If the intervals involved were previously related, then the algorithm only continues while it is further constraining the old values. The bad property is that it requires vast amounts of memory, since every interval is explicitly related to every other interval. This problem is addressed in the next section.

Ar1B \ B r2 C	<	>	d	di
<	<	no info	< o d m	<
>	no info	>	> oi d mi	>
d	<	>	d	no info
di	< o di m	> oi di mi =	o oi d di	di

Figure 4: The Transitivity Table

IV. Maintaining Temporal Relations Using Reference Intervals

We need some method in which to restrict the propagation of constraints throughout the network but to allow the inferences locally. Furthermore, we should later be able to infer the relationships that were not produced by the original propagation of constraints. This latter condition eliminates the possibility of restricting the propagation by only allowing propagation paths of some fixed length, for there seems to be no method available to recover the information not inferred except by a brute force search. A more structured view of the network is needed.

The solution described here allows the user of the system to determine what inferences are important to perform "automatically," and what ones can be deferred. It entails introducing the concept of *reference intervals*. Each time a relation is added, it is added with respect to a reference interval. The user of this system may set up the reference intervals as desired. In general, they probably will often reflect part of the *during* hierarchy, but also may reflect the semantic clustering of events.

The constraint propagation is then restricted to inferring new relationships between points that share a common reference interval. We shall allow time intervals to have more than one reference interval. In the graphical notation, reference intervals will be indicated in parentheses following each node. Thus the situation described above would be captured by the nodes

I1(R1 R3), I2(R1), I3(R2), I4(R3)

Throughout the paper, nodes will indicate their reference intervals only when it is relevant to the discussion. Since there is a fixed set of reference intervals for any one node, no confusion should arise.

Reference intervals are simply other intervals themselves, and so have their own reference intervals as well. While it is up to the user of this system to choose what reference intervals are present, there are some particularly useful techniques worth mentioning.

The first type of interval useful as a reference is an interval that more or less divides a large set of time intervals into those before and those after. Kahn and Gorry [1977] use these types of reference intervals, and give examples such as "the birthdate of a person," "my graduation from school." These events divide the set of intervals relating to a person's life into those before and those after. In the representation here we get this behavior using two reference intervals, one for the time before the event, and one for the time after.

As a simple example, let G be the time of my graduation from school, from which Gb (before G) and Ga (after G) are defined as reference intervals. Let L be the time of my learning to play chess, and W be the time of my winning a state lottery. Assume further that L is before G, so is stored with respect to the reference interval Gb, and that W is after G, so is stored with respect to the reference interval Ga. Since L and W do not share a common reference interval, the explicit relationship between them is not computed or stored. This information is summarized below, using X as a dummy reference interval for Ga and Gb.

Gb(X) *before* Ga(X)
L(Gb) *during* Gb(X)
W(Ga) *during* Ga(X)

To find the relation between L and W, we follow up the reference interval chains until we find an explicit relationship. Thus, from L we first find Gb, and from W we find Ga. It is explicitly asserted that Gb is *before* Ga. The trace of this search path indicates the facts relevant to infer that L is *before* W using the transitivity table (Figure 4). Of course, the real benefit of this approach is only realized when there are many intervals that either reference Ga or that reference Gb.

Reference intervals can also be used to control inference that is more semantically or pragmatically based. For instance, in a dialogue system, one needs to keep track of the focus and topic of discussion. The discussion of one topic may introduce many facts that should be treated as a unit. Explicit signals are often given to indicate changes in topic. In general, we do not want to examine the relationships between time intervals introduced across topics as closely as those introduced within a single topic. Thus we should introduce a new reference interval for each topic.

V. Maintaining the Present

For this representation to be truly useful in a dialogue system it must be able to capture the notions of past, present, and future. Furthermore, since the "the present" is continually moving into "the future," this updating must not involve a large-scale reorganization of the data base each time it is done.

The method used here is simply to map "now" onto an interval that is maintained in a similar manner to all other intervals in the system. If we are clever about using its reference interval, so that most relations are inferred via the reference interval rather than directly, then updating "now" to a new interval that has the same reference interval should not be too expensive.

For example, let "now" be interval N1, which is *during* its reference interval R1. An example state of the data base would be

N1(R1) *during* R1
R1 *before* I1, R1 *after* I2, R1 *during* I3

From this we *can* infer easily that the present (i.e., N1) is during I3, before I1, and after I2. If "now" then is updated (slightly), N2 can be defined as the new "now" using the same reference interval by adding the facts

N2(R1) *during* R1, N2(R1) *after* N1(R1)

Thus, "now" has been updated but most of the relations in the data base have been unaffected, for the effects of N2 will only propagate to intervals referenced by R1. The reference interval R1 has "protected" the rest of the data base from minor- change in the present moment. Of course, eventually "now" will cease to be during R1 and a new reference interval will be needed. This will involve a more major update to the data base, but the amount of work can be reduced if R1 itself has a reference interval that "protects" much of the data base from it.

Thus we need a hierarchy of reference intervals, each containing the present moment. This hierarchy should be designed to mirror the set of English terms that can be used to refer to the present. For example, in English we can refer to the exact moment of an utterance, as well as to larger intervals such as "this morning," "today," "during this lecture," and "while at this bar." These intervals typically have well defined starting and termination points. Thus it is reasonable to assume that the temporal data base will receive explicit notification when one of them ceases to contain the present. This allows the following important assumption:

when updating the "now" interval, unless otherwise stated, its relationship to its reference interval(s) remains constant.

When one of the reference intervals in the hierarchy ceases to contain the present moment, a new reference interval is selected. (This new interval should usually be provided by the user.) This update is done in the identical fashion as described above with "now." In particular, the relationship with the higher level reference interval remains constant. A new "now" interval, below the new reference interval in the hierarchy, must be introduced. For example, the beginning of a new day would make much of the old hierarchy part of the past (i.e., "yesterday").

While many intervals will be generated by this succession of "nows," a large number of them will have been used only to index the time of an utterance. Thus, they generally can be deleted without harm.

Consider the simple example in Figure 5, which describes a possible hierarchy for the present, where my uttering the present sentence is during interval **now2**. Let indentation indicate a *during* relationship and the ordering from top to bottom indicate the *before* relation.

day3(X)	am3(day3)	I1(am3)	
		amtalks(am3)	I2(amtalks)
			I3(amtalks)
	pm3(day3)	I5(pm3)	
		beforetalk(pm3)	I6(beforetalk)
		talk(pm3)	I7(talk)
			now1(talk)
			now2(talk)
		aftertalk(pm3)	

Figure 5: A Possible Hierarchy Indicating the Present

Thus, **day3**, which is today, contains the intervals **am3** and **pm3**, and is also their reference interval. Interval **am3** is *before* **pm3**. In turn, **am3** contains I1 and **amtalks**, and is their reference interval. The intervals of the form **now**i are the intervals that at one time or another were/will be part of the present. The underlined intervals give the structure that is currently present. Thus "now" refers to interval **now2**.

Given this state, we can infer that interval I3 is part of the past by tracing up the hierarchy from I3 until we find an explicit relationship with the present (the underlined nodes). Since I3 is *during* **am3**, which is *before* **pm3**, we can infer that I3 is *before* the present.

To update "now" without changing its reference point, we simply add a new interval **now3**, assert that **now3** *during* **talk** and **now2** *before* **now3**. The fact that I3 is still in the past after this update is calculated in the identical manner as before. The result is shown in Figure 6 (omitting the reference interval information).

day3	am3	I1	
		amtalks	I2
			I3
	pm3	I5	
		beforetalk	I6
		talk	I7
			now1
			now2
			now3
		aftertalk	

Figure 6: The Hierarchy After a Minor "Now" Update

To update a reference interval, say to update the interval **talk** when I finish my talk, the same operation as with a minor update is performed, and then a new interval to represent the new "now" is added. In many cases, however, the user will specify an already known interval to be the new reference (e.g. updating "today" to "tomorrow"). Assuming that **aftertalk** is the newly specified reference interval, we need only create a new "now" node, say **now4**.

In addition, when the reference node talk is updated, some of the old nodes that represented "now" could be deleted. In particular, **now1**, **now2**, and **now3** might not be needed any longer. The new hierarchy is shown in Figure 7.

day3	am3	I1	
		amtalks	I2
			I3
	pm3	I5	
		beforetalk	I6
		talk	I7
		aftertalk	now4

Figure 7: The Hierarchy After a Major "Now" Update

VI. Persistence of Intervals

The last requirement described in the introduction was that the representation should facilitate plausible inferences of the form "if fact P is true now, it will remain true until noticed otherwise." Most of the issues concerning this fall outside the range of this paper, as this system only knows about time intervals. However, a simple trick using this representation makes inferences of the above form easy to implement.

Typically, when a new fact is learned, its exact extent in time is not known. For instance, when I parked my car in the parking lot this morning I knew its location. Sitting at my desk now, I assume it is still there, though I have no proof of that fact. In general, I assume it will remain where it is until I pick it up. Thus, although I don't know the extent of the interval in which my car is parked, I want to be able to assume that this fact holds later in the day.

The temporal representation is already based on the observation that most time intervals do not have precisely defined limits. If we allow the user to specify that some intervals should be assumed to extend as far as possible given the constraints, then we can use such intervals to index facts that are assumed to persist until discovered otherwise.

Thus, if we let a fact P be indexed by a *persistent interval* Tp, then testing P later during an interval t will succeed (by assumption) if it is possible that t is *during* Tp. Checking whether relationships between intervals are possible is easy, since the representation explicitly maintains this information.

For example, let Tp represent the interval in which my car is in the parking lot. I know that Tp is *met by* Tarrive, where Tarrive is the time that I arrived at school today. Then, let "now" be represented by the interval Tnow, where Tnow *after* Tarrive. We can conclude that my car is in the parking lot as follows. Since it is there during Tp, we are interested in whether it is possible that Tnow is *during* Tp. The known constraints allow us to infer the following:

Tp *met by* Tarrive, Tarrive *before* Tnow
=> Tp--(< o di m)-> Tnow

Thus it is possible that Tnow is *during* Tp, since it is possible that Tp contains ("di") Tnow. So the test succeeds.

Of course, if it is later learned that the car was found to be missing during some time interval Tmiss, then Tp is constrained to be *before* Tmiss (even though it is still persistent). If Tnow is then after or during Tmiss, then it is not possible any longer that Tnow is during Tp.

McDermott [1981] introduces a similar notion that he also calls persistence (must be a good term!). His scheme involves specifying a (numeric) duration over which the proposition will be assumed to hold after it becomes true. In my scheme described here, limits are introduced by constraining the persistent interval to be *before* (or some other relation) some other interval. Thus our approaches differ on how central a role explicit duration information plays. I have explicitly avoided all duration scales here, but deal with them briefly in [Allen, 1981].

Managing a system such as this is a difficult problem that requires some form of truth maintenance (e.g. see [Doyle, 1979]). These issues, however, are independent of the temporal representation. All that is shown here is that the necessary temporal calculations are easily done within this framework.

VII. Summary

This paper argues that a model for representing temporal information such as that acquired in dialogues is more naturally interval-based rather than point-based. It describes a method of representing the relationships between temporal intervals in a hierarchical manner using constraint propagation techniques. By using a notion of reference intervals, the amount of computation involved when adding a fact can be controlled in a predictable manner.

Evidence for the viability of this representation is presented by discussing a few important problems that arise in dialogues involving temporal reference. The first problem involved the representation of the present moment, which must be able to be updated frequently and thus must be done efficiently. The use of reference intervals to insulate the present moment from the bulk of the facts in the knowledge base was suggested as a promising technique.

The other problem discussed involved the ability to assume various conditions would hold until explicitly discovered otherwise. Since the techniques suggested already computed the set of possible relationships that could hold between two intervals, it was easy to implement this assumption mechanism. If an interval was labelled as *persistent*, then it was assumed to extend as far as possible given its constraints.

Acknowledgements

Many thanks to Jerry Feldman, Alan Frisch, Don Perlis, Ken Sloan, and the AI Study Group for their comments on a draft version of this paper, and for other general discussions of issues while the research was in progress.

References

Allen, J.F., "A Plan-Based Approach to Speech Act Recognition," Ph.D. thesis, Department of Computer Science, University of Toronto, 1979.

Allen, J.F., "A General Model of Action and Time," in progress.

Allen, J.F., "Maintaining Knowledge about Temporal Intervals," TR86, Computer Science Department, University of Rochester, January 1981.

Bruce, B.C., "A Model for Temporal References and its Application in a Question Answering Program," *Artificial Intelligence 3*, 1972.

Doyle, J., "A Truth Maintenance System," *Artificial Intelligence 12*, 3, November 1979.

Grosz, B.J., "Utterance and Objective: Issues in Natural Language Communication," *Proc.*, 6th IJCAI, Tokyo, August 1979.

Hayes, Patrick J., "The Naive Physics Manifesto," in D. Michie (Ed). *Expert Systems in the Micro-Electronic Age*. Edinburgh: Edinburgh U. Press, 1979.

Kahn, K.M. and G.A. Gorry, "Mechanizing Temporal Knowledge," *Artificial Intelligence 9*, 1977.

McDermott, D., "Planning and Acting," *Cognitive Science 2-2*, 1978.

McDermott, D., "A Temporal Logic for Reasoning About Processes and Plans," Research Report 196, Department of Computer Science, Yale University, 1981.

Wilensky, R., "Understanding Goal-Based Stories," Ph.D. thesis, Yale University, 1978.