Constraint Propagation Algorithms for Temporal Reasoning

Marc Vilain
BBN LABORATORIES
10 Moulton ST.
CAMBRIDGE, MA 02238

Henry Kautz
UNIVERSITY OF ROCHESTER
COMPUTER SCIENCE DEPT.
ROCHESTER, NY 14627

Abstract: This paper considers computational aspects of several temporal representation languages. It investigates an interval-based representation, and a point-based one. Computing the consequences of temporal assertions is shown to be computationally intractable in the interval-based representation, but not in the point-based one. However, a fragment of the interval language can be expressed using the point language and benefits from the tractability of the latter.1

The representation of time has been a recurring concern of Artificial Intelligence researchers. Many representation schemes have been proposed for temporal reasoning; of these, one of the most attractive is James Allen's algebra of temporal intervals [Allen 83]. This representation scheme is particularly appealing for its simplicity and for its ease of implementation with constraint propagation algorithms.

Reasoners based on this algebra have been put to use in several ways. For example, the planning system of Allen and Koomen [1983] relies heavily on the temporal algebra to perform reasoning about the ordering of actions. Elegant approaches such as this one may be compromised, however, by computational characteristics of the interval algebra. This paper concerns itself with these computational aspects of Allen's algebra, and of a simpler algebra of time points.

Our perspective here is primarily computation-theoretic. We approach the problem of temporal representation by asking questions of complexity and tractability. In this light, this paper examines Allen's interval algebra, and the simpler algebra of time points.

The bulk of the paper establishes some formal results about the temporal algebras. In brief these results are:

1. Determining consistency of statements in the interval algebra is NP-hard, as is determining all consequences of these statements. Allen's polynomial-time constraint propagation algorithm is sound but not complete for these tasks.

2. In contrast, constraint propagation is sound and complete for computing consistency and consequences of assertions in the time point algebra. It operates in $O(n^3)$ time and $O(n^2)$ space.

3. A restricted form of the interval algebra can be formulated in terms of the time point algebra. Constraint propagation is sound and complete for this fragment.

Throughout the paper, we consider how these formal results affect practical Artificial Intelligence programs.

The Interval Algebra

Allen's interval algebra has been described in detail in [Allen 83]. In brief, the elements of the algebra are relations that may exist between intervals of time. Because the algebra allows for indeterminacy in temporal relations, it admits many possible relations between intervals (213 in fact). But all of these relations can be expressed as vectors of definite simple relations, of which there are only thirteen.2 The thirteen simple relations, whose definitions appear in Figure 1, precisely characterize the relative starting and ending points of two temporal intervals. If the relation between two intervals is completely defined, then it can be exactly described with a simple relation. Alternatively, vectors of simple relations introduce indeterminacy in the description of how two temporal intervals relate. Vectors are interpreted as the disjunction of their constituent simple relations.

Two examples will serve to clarify these distinctions (please refer to figure 2). Consider the simple relations BEFORE and AFTER:

- A BEFORE B
- B AFTER A

A MEETS B
B MET-By A
A OVERLAPS B
B OVERLAPPED-By A
A STARTS B
B STARTED-By A
A DURING B
B CONTAINED-By A
A ENDS B
B ENDED-By A
A EQUALS B
B EQUALS A

Figure 1: Simple relations in the interval algebra

Two examples will serve to clarify these distinctions (please refer to figure 2). Consider the simple relations BEFORE and AFTER: they hold between two intervals that strictly follow each other, without overlapping or meeting. The two differ by the order of their

---

1This research was supported in part by the Defense Advanced Research Projects Agency, under contracts N00014-85-C-0079 and N-00014-77-C-0378.

2In fact, these thirteen simple relations can be in turn expressed in terms of universally and existentially quantified expressions involving only one truly primitive relation. For details, see [Allen & Hayes 86].
arguments: today John ate his breakfast before he ate his lunch, and he ate his lunch after he ate his breakfast. To illustrate relation vectors, consider the vector (before meets overlaps). It holds between two intervals whose starting points strictly precede each other, and whose ending points strictly precede each other. The relation between the ending point of the first interval and the starting point of the second is left ambiguous. For instance, say this each other, and whose ending points strictly precede each other.

To find the relation between intervals $A$ and $B$, that is implied by $V_1$ and $V_2$, the two vectors are summed:

$$V_1 = (\text{BEFORE MEETS OVERLAPS})$$
$$V_2 = (\text{OVERLAPS STARTS DURING}).$$

Then the product of $V_1$ and $V_2$ is

$$V_1 \times V_2 = (\text{BEFORE}).$$

As with addition, the multiplication of two vectors is computed by inspecting their constituent simple relations. The constituents are pairwise multiplied by following a simplified multiplication table, and the results are combined to produce the product of the two vectors. See [Allen 83] for details.
Let Table be a two dimensional array, indexed by intervals, in which Table[i][j] holds the relation between intervals i and j. Table[i][j] is initialized to (BEFORE MEETS ... AFTER), the additive identity vector consisting of all thirteen simple relations; except for Table[i][j] which is initialized to (EQUAL).

Let Queue be a FIFO data structure that will keep track of those pairs of intervals whose relation has been changed. Let Intervals be a list of all intervals about which assertions have been made. */

To Add(R<i,j>):
/* R<i,j> is a relation being asserted between i and j. */

begin
Old ← Table[i][j];
Table[i][j] ← Table[i][j] + R<i,j>;
If Table[i][j] ≠ Old
then Place <i,j> on Fifo Queue;
Intervals ← Intervals ∪ {i,j};
end;

To Close:
/* Computes the closure of assertions added to the database. */

While Queue is not empty do
begin
Get next <i,j> from Queue;
Propagate(i,j);
end;

To Propagate(<i,j>):
/* Called to propagate the change to the relation between intervals i and j. */

For each interval K in Intervals do
begin
Temp ← Table[i][K] + (Table[i][j] x Table[j][K]);
If Temp = 0
then {signal contradiction};
If Table[i][K] ≠ Temp
then Place <K,i> on Queue;
Table[i][K] ← Temp;
Temp ← Table[i][j] + (Table[j][i] x Table[j][j]);
If Temp = 0
then {signal contradiction};
If Table[j][j] ≠ Temp
then Place <j,K> on Queue;
Table[j][j] ← Temp;
end;

Figure 4: The constraint propagation algorithm
assertion \( i \) \((R) j\), where \( R \) is one of the thirteen simple relations. The relation vector that holds between \( i \) and \( j \) is the one containing those simple relations that the oracle didn't reject.

To show that determining consistency follows from determining closure, assume the existence of a closure algorithm. To see if a set of assertions is consistent, run the algorithm, and inspect each of the \( O(n^2) \) relations between the \( n \) intervals mentioned in the assertions. The database is inconsistent if any of these relations is the inconsistent vector: this is the vector composed of no constituent simple relations.

The two preceding theorems demonstrate that computing the closure of assertions in the interval algebra is NP-hard. This result casts great doubts on the computational tractability of the algebra, as no NP-hard problem is known to be solvable in less than exponential time.

**Consequences of Intractability**

Several authors have described exponential-time algorithms that compute the closure of assertions in the interval algebra, or some subset thereof. Valdés-Pérez [1986] proposes a heuristically pruned algorithm which is sound and complete for the full algebra. The algorithm is based on analysis of set-theoretic constructions. Malik & Binford [1983] can determine closure for a fraction of the interval algebra with the exponential Simplex algorithm. As we shall show below, their method is actually more powerful than need be for the fragment that they consider.

Even though the interval algebra is intractable, it isn't necessarily useless. Indeed, it is almost a truism of Artificial Intelligence that all interesting problems are computationally at least NP-hard (or worse)! There are several strategies that can be adopted to put the algebra to work in practical systems.

The first is to limit oneself to small databases, containing on the order of a dozen intervals. With a small database, the asymptotically exponential performance of a complete temporal reasoner need not be noticeably poor. This is in fact the approach taken by Malik and Binford to manage the exponential performance of their Simplex-based system. Unfortunately, it can be very difficult to restrict oneself to small databases, since clustering information in this way necessarily prevents all but the simplest interrelations of intervals in separate databases.

Another strategy is to stick to the polynomial-time constraint propagation closure algorithm, and accept its incompleteness. This is acceptable for applications which use a temporal database to notate the relations between events, but don't particularly require much inference from the temporal reasoner. For applications which make heavy use of temporal reasoning, however, this may not be an option.

Finally, an alternative approach is to choose a temporal representation other than the full interval algebra. This can be either a fragment of the algebra, or another representation altogether. We pursue this option below.

**A Point Temporal Algebra**

An alternative to reasoning about intervals of time is to reason about points of time. Indeed, an algebra of time points can be defined in much the same way as was the algebra of time intervals. As with intervals, points are related to each other through relation vectors which are composed of simple point relations. These primitive relations are defined in Figure 5.

As with the interval algebra, the point temporal algebra possesses addition and multiplication operations. These operations, whose tables are given in Appendix, mirror the operations in the interval algebra. Addition is used to combine two different measures of the relation of two points. Multiplication is used to determine the relation between two points \( A \) and \( B \), given the relations between each of \( A \) and \( B \) and some intermediate point \( C \).

**Computing Closure in the Point Algebra**

As was the case with intervals, determining the closure of assertions in the point algebra is an intractable operation. Fortunately, the point algebra is sufficiently simple that closure can be computed in polynomial time. To do so, we can directly adapt the constraint propagation algorithm of Figure 4. Simply replace the interval vector addition and multiplication operations with point additions and multiplications, and run the algorithm with point assertions instead of interval assertions.

As before, the algorithm runs to completion in \( O(n^2) \) time, where \( n \) is the number of points about which assertions have been made. As with the interval algebra, the algorithm is sound: any relation that it infers between two points follows from the user's assertions. This time, however, the algorithm is complete. When it terminates, the closure of the point assertions will have been correctly computed.

We prove completeness by referring to the model theory of the time point algebra. In essence, we consider any database over which the algorithm has been run, and construct a model for any possible interpretation of the database. If the database is indiscernible, a model must be constructed for each possible resolution of the indiscernibility. We choose the real numbers to model time points. A model of a database of time points is simply a mapping between those time points and some corresponding real numbers. The relations between time points are mapped to relations between real numbers in the obvious way. For example, if time point \( A \) precedes time point \( B \) in the database, then \( A \)'s corresponding number is less than \( B \)’s.

**Theorem 4:** The constraint propagation algorithm is complete for the time point algebra. That is, a model can be constructed for any interpretation of the processed database.

**Proof:** (Sketch) We first note that the algorithm partitions the database into one or more partial order graphs. After the algorithm is run, each node in a graph corresponds to a cluster of points. These are all points related to by the vector \((\text{SAME})\); note that the algorithm computes the transitive closure of \((\text{SAME})\) assertions. As in the graph either indicates precedence (the vectors \((\text{PRECEDES})\) or \((\text{PRECEDES SAME})\), or their inverses) or disequality (the vector \((\text{PRECEDES FOLLOWS})\)). At the bottom of each graph is one or more "bottom" nodes: nodes which are preceded by no other node.

Further, when the algorithm has run to completion the

| \( A \) PRECEDES \( B \) | \( A \) \( \bullet \) | \( B \) \( \bullet \) |
| \( A \) SAME \( B \) | \( A \) \( \bullet \) | \( B \) \( \bullet \) |
| \( A \) FOLLOWS \( B \) | \( A \) \( \bullet \) | \( B \) \( \bullet \) |

*Figure 5: Simple point relations*

This demonstrates completeness in the following sense. If there were an interpretation of the processed database for which no model could be constructed, the algorithm would be incomplete. It would have failed to eliminate a possible interpretation prohibited by the original assertions.
graphs are all consistent, in the following two senses. First, all points are linearly ordered: there is no path from any point in a graph back to itself that cololy traverses precedence arcs (time doesn't curve back on itself). Second, no two points that are in the same cluster were asserted to be disequal with the (PRECEDES FOLLOWS) vector. If the user had added any assertions that contradicted these consistency criteria, the algorithm would have signalled the contradiction.

Note that all of the preceding properties can be shown with simple inductive proofs by considering the algorithm and the addition and multiplication tables.

The model construction proceeds by picking a cluster of points (i.e., a node) at the "bottom" of some graph and assigning all of its constituent points to some real number. The cluster is then removed from the graph, and the process proceeds on with another real number (greater than the first) and another cluster (either in the same graph or in another one). The process is complicated somewhat because some clusters may be "equal" to other clusters (their constituent points may be related by some vector containing the SAME relation). For these cases it is possible to "collapse" several (zero, one, or more) of these clusters together, and assign their constituent points to the same real number. Some other clusters may be "disequal". For these, we must just make sure never to "collapse" them together. Because the choice of which "bottom" node to remove and which clusters to collapse is non-deterministic, the model construction covers all possible interpretations of the database.

### Relating the interval and point algebras

The tractability of the point algebra makes it an appealing candidate for representing time. Indeed, many problems that involve temporal sequencing can be formulated in terms of simple points of time. This approach is taken by any of the planning programs that architecture, this representation problem can be dealt with either by invoking an exponential temporal reasoner, or by bringing to bear planning-specific knowledge about the ordering of actions.

#### Consequences of These Results

Increasingly, the tools of knowledge representation are being put to use in practical systems. For these systems, it is often crucial that the representation components be computationally efficient. This has prompted the Artificial Intelligence community to start taking seriously the performance of AI algorithms. The present paper, by considering critically the computational characteristics of several temporal representations, follows this recent trend.

What lessons may we learn from analyses such as this? Of immediate benefit is an understanding of the computational advantages and disadvantages of different representation languages. This permits informed decisions as to how the representation components of application systems should be structured. We can better understand when to use the power of general representations, and when to set these general tools aside in favor of more application-specific reasoners.

A close scrutiny of the ongoing achievements of Artificial Intelligence enables a better understanding of the nature of AI methods. This process is crucial for the maturation of our field.
Appendix: Algebraic Operations in the Point Algebra

Addition and multiplication are defined in the point algebra by the two tables in Figure 7. These operations both have constant-time implementations if the relation vectors between time points are encoded as bit strings. With this encoding, both operations can be performed by simple lookups in two-dimensional (8 x 8) arrays. Alternatively, addition can be performed with an even simpler 3-bit logical AND operation.

Key to symbols:

0 is (), the null vector
< is (PRECEDES)
<= is (PRECEDES SAME)
> is (FOLLOWS)
>= is (SAME FOLLOWS)
= is (SAME)
-= is (PRECEDES FOLLOWS)
? is (PRECEDES SAME FOLLOWS)

Figure 7: Addition and multiplication in the time point algebra

References

[Allen 83] Allen, J. F.
Maintaining Knowledge About Temporal Intervals.

[Allen 84] Allen, J. F.

[Allen & Hayes 85] Allen, J. F. and Hayes, P. J.
A Common-Sense Theory of Time.
In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, pages 528-531. The International Joint Conference on Artificial Intelligence (IJCAI), Los Angeles, CA, August, 1985.

Planning Using a Temporal World Model.

STRIPS: A new approach to the application of theorem proving to problem solving.

[Malik & Binford 83] Malik, J. and Binford, T. O.
Reasoning in Time and Space.
In Proceedings of the Eighth International Joint Conference on Artificial Intelligence, pages 343-345. The International Joint Conference on Artificial Intelligence (IJCAI), Karlsruhe, W Germany, August, 1983.

[Schmolze 86] Schmolze, J. G.

[Simmons 83] Simmons, R. G.
The Use of Qualitative and Quantitative Simulations.

[Valdes-Perez 86] Valdes-Perez, R. E.
Spatio-Temporal Reasoning and Linear Inequalities.
Unpublished A.I Memo, Massachusetts Institute of Technology Artificial Intelligence Laboratory.