Constraint Propagation Algorithms for Temporal Reasoning: A Revised Report

Marc Vilain
The MITRE Corporation
Burlington Rd.
Bedford, Mass. 01730

Henry Kautz
AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill NJ 07974

Peter van Beek
Dept. of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

Abstract: This paper revises and expands upon a paper presented by two of the present authors at AAAI 1986 [Vilain & Kautz 1986]. As with the original, this revised document considers computational aspects of interval-based and point-based temporal representations. 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. The present paper departs from the original primarily in correcting claims made there about the point algebra, and in presenting some closely related results of van Beek [1989].

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 1983]. 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 the computational aspects of Allen's algebra, and of two variants 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 establishes some formal results about these temporal algebras. In brief these results are:

- Determining consistency of statements in the interval algebra is NP-hard, as is determining the deductive closure of these statements. Allen's polynomial-time constraint propagation algorithm for deductive closure is thus incomplete.
- We define a restricted form of the interval algebra, concerned with measuring the relative durations of events. This algebra can be formulated in terms of a time point algebra without disequality (≠). Allen's propagation algorithm is sound and complete for this fragment, and operates in $O(n^3)$ time and $O(n^2)$ space.
- We also define a broader interval algebra fragment, corresponding to the time point algebra with ≠. A variant propagation algorithm performs closure in this fragment in $O(n^4)$ time.

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 1983]. In brief, the elements of the algebra are *relations* that may exist between intervals of time. Because the algebra allows for indefiniteness in temporal relations, it admits many possible relations between intervals (2^{13} in fact). But all of these relations can be expressed as *vectors* of definite *simple relations*, of which there are only thirteen. The thirteen simple

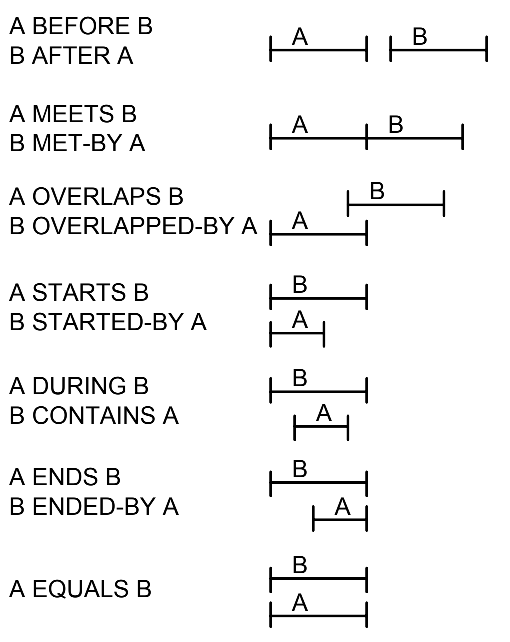

Figure 1: Simple Interval Relations

relations, whose illustration appears 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.[1] Alternatively, vectors of simple relations introduce indefiniteness in the description of how two temporal intervals relate. Vectors are interpreted as denoting the set of possible simple relations that hold between two intervals. Informally, a vector of simple relations can be understood as the “disjunction” of its constituent relations.

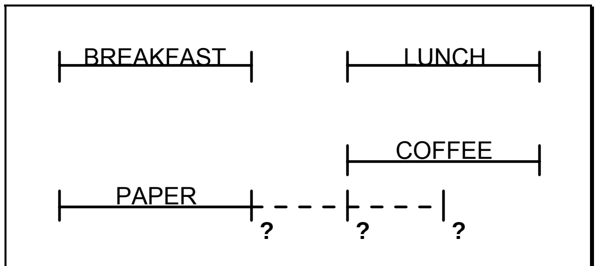

Figure 2: Simple Relation and Relation Vector

Two examples will serve to clarify these distinctions (see Figure 2). Consider the simple relations *BEFORE* and *AFTER*: they hold between intervals that strictly follow each other, without overlapping or meeting. The two differ by the order of their arguments: today John ate breakfast *BEFORE* he ate lunch, and he ate lunch *AFTER* he ate 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 morning John started reading the paper before starting breakfast, and he finished the paper before his last sip of coffee. If we didn't know whether he was done with the paper before starting his coffee, at the same time as he started it, or after, we would then have the paper reading *(BEFORE MEETS OVERLAPS)* the coffee drinking.

Returning to our formal discussion, we note that the interval algebra is principally defined in terms of vectors. Although simple relations are an integral part of the formalism, they figure primarily as a convenient way of notating vector relations. The mathematical operations defined over the algebra are given in terms of vectors; in a reasoner built on the temporal algebra, all user assertions are made with vectors.

Two operations, an addition and a multiplication, are defined over vectors in the interval algebra. Given two different vectors describing the relation between the same pair of intervals, the addition operation “intersects” these vectors to provide the least restrictive relation that the two vectors together admit. The need to add two vectors arises from situations where one has several independent measures of the relation of two intervals. These measures are combined by summing the relation vectors for the measures. For example, say the relation between intervals *A* and *B* has been derived by two valid measures as being both

V_1 = *(BEFORE MEETS OVERLAPS)*
V_2 = *(OVERLAPS STARTS DURING)*.

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

V1 + V2 = *(OVERLAPS)*.

Algorithmically, the sum of two vectors is computed by finding their common constituent simple relations.

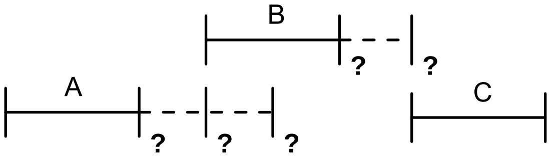

Figure 3: Multiplying Relation Vectors

Multiplication, or vector composition, is defined between pairs of vectors that relate three intervals *A*, *B*, and *C*. More precisely, if V_1 relates intervals *A* and *B*, and V_2 relates *B* and *C*, the product of V_1 and V_2 is the least restrictive relation between *A* and *C* that is permitted by V_1 and V_2. Consider, for example, the situation in Figure 3. If we have

V_1 = *(BEFORE MEETS OVERLAPS)*
V_2 = *(BEFORE MEETS)*

then the product of V_1 and V_2 is

$V_1 \times V_2$ = *(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 1983a] for details.

Determining Closure in the Interval Algebra

To an application reasoning with Allen's interval algebra, the primary operation of interest is determining the closure of a set of temporal assertions. This can be understood as a deductive closure. Given as premise a set of temporal assertions, the closure consists of all the temporal relations which follow from the premises.

[1] In fact, these thirteen simple relations can be in turn precisely axiomatized using only one truly primitive relation. For details, see [Allen & Hayes, 1985].

```
{   Table is 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
    the additive identity vector consisting of all thirteen
    simple relations;  except for Table[i,i], which is
    intialized to (EQUALS).
    Queue is a FIFO data structure that keeps track of
    pairs of intervals whose relation has been
    changed.
    Intervals is 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 pair <i,j> on fifo Queue;
        Intervals ♦ Intervals ≈ {i,j};
    end;

To Close
    {  Compute 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)
    {  Propagates the change to the relation between i
       and j to all  other intervals. }
    for each interval k in Intervals do
        begin
            Temp ♦ Table[i,k] + (Table[i,j] x Table[j,k]);
            if Temp = □   { □ is the inconsistent vector. }
                then Signal contradiction;
            if Table[i,k] ≠  Temp
                then Place pair <i,k> on Queue;
            Table[i,k] ♦ Temp;
            Temp ♦ Table[k,j] + (Table[k,i] x Table[i,j]);
            if Temp = □
                then Signal contradiction;
            if Table[k,j] ≠  Temp
                then Place pair <k,j> on Queue;
            Table[k,j] ♦ Temp;
        end;
```

Figure 4: Constraint propagation algorithm.

To formalize this notion, we need to turn to some model-theoretic considerations. For our purposes, temporal intervals can be modelled as pairs of distinct numbers on the real line. (Other axiomatizations exist: the rational numbers [Ladkin 1987] or the integers [Allen & Hayes 1985], but these distinctions are not crucial here.) Given a set of intervals I with assertions relating the intervals in I, an interpretation of these temporal relations is thus a mapping from each interval in I to a consistent model, i.e., to some pair of reals on the time line which is consistent with the premise assertions.

Computing the closure of the premise relations on I consists of determining the minimal relation vectors between each *i* and *j* in I. Such a minimal vector between *i* and *j* consists only of the *consistent* simple relations of the premise vector, i.e., those which are satisfied by some interpretation of the premises. We can think of this as discarding the inconsistent simple relations from all the premise assertions on I. See [van Beek 1989] for details.

In Allen's model, closure is computed with a constraint propagation algorithm. The operation of this forward-chaining algorithm is driven by a queue. Every time the relation between two intervals *i* and *j* is changed, the pair <*i,j*> is placed on the queue. The closure algorithm, shown in Figure 4, is initiated by calling procedure *Close*, and operates by removing interval pairs from the queue. For every pair <*i,j*> that it removes, the algorithm determines whether the relation between *i* and *j* can be used to constrain the relation between *i* and other intervals in the database, or between *j* and these other intervals. If a new relation can be successfully constrained, then the pair of intervals that it relates is in turn placed on the queue. The process terminates when no more relations can be constrained.

As Allen suggests [1983a], this constraint propagation algorithm runs to completion in time polynomial with the number of intervals in the temporal database. He provides an estimate of $O(n^2)$ calls to the *Propagate* procedure. A more fine-grained analysis reveals that when the algorithm runs to completion, it will have performed $O(n^3)$ multiplications and additions of temporal relation vectors.

Theorem 1: Let I be a set of n intervals about which m assertions have been added with the *Add* procedure. When invoked, the *Close* procedure will run to completion in $O(n^3)$ time.

Proof: A pair of intervals <*i,j*> is entered on *Queue* when its relation, stored in *Table[i,j]*, is non-trivially updated. First note that no more than $O(n^2)$ pairs of intervals <*i,j*> are ever entered onto the queue. This is because there are only n^2 relations possible between the *n* intervals, and because each relation can only be non-trivially updated a constant number of times. This constant bound arises because every non-trivial update by definition removes at least 1 simple relation from the vector encoding the relation between *i* and *j*. Since there are only 13 such relations, <*i,j*> can only be updated at most 13 times.

Next, note that every time a pair <*i,j*> is removed from *Queue* for updating, the algorithm performs O(n) vector operations. These operations occur in the *Propagate* procedure when comparing the relation between intervals *i* and *j* to that between *j* and *k*, and also to that between *k* and *i*. There are *n* such *k*, and each set of comparisons requires 2 vector additions and 2 vector multiplications, leading to an overall cost of 2n vector additions and 2n vector multiplications to update <*i,j*>.

To complete the proof, we see that each of the $O(n^2)$ updates required for computing closure in turn requires O(n) computation, leading to an overall complexity of $O(n^3)$ vector operations.

The vector operations can be considered here to take constant time. By encoding vectors as bit strings, addition can be performed with a 13-bit integer *AND* operation. For multiplication, the complexity is actually $O(|V_1| \times |V_2|)$, where $|V_1|$ and $|V_2|$ are the "lengths" of the two vectors to be multiplied (i.e., the number of simple constituents in each vector). Since vectors contain at most 13 simple constituents, the complexity of multiplication is bounded, and the idealization of multiplication as operating in constant time is acceptable.

Note that the polynomial time characterization of the constraint propagation algorithm of Figure 4 is somewhat misleading. Indeed, Allen [1983] demonstrates that the algorithm is sound, in the sense that it never infers an invalid consequence of a set of assertions. However, Allen also shows that the algorithm is incomplete: he produces an example in which the algorithm does not make all the inferences that follow from a set of assertions. He suggests that computing the closure of a set of temporal assertions might only be possible in exponential time. Regrettably, this appears to be the case. As we demonstrate in the following paragraphs, computing closure in the interval algebra is an NP-complete problem.

Intractability of the Interval Algebra

To demonstrate that computing the closure of assertions is NP-complete, we first show that determining the consistency (or satisfiability) of a set of assertions is NP-hard. We then extend this to NP-complete and show the consistency and closure problems to be equivalent.

Theorem 2: Determining the satisfiability of a set of assertions in the interval algebra is NP-hard.

Proof (Due to Kautz): The theorem is proven by reducing the 3-clause satisfiability problem (or 3-SAT) to the problem of determining satisfiability of assertions in the interval algebra. To do this, we construct a (computationally trivial) mapping between a formula in 3-SAT form[2] and an equivalent encoding of the formula in the interval algebra. Conceptually, this is done by creating three groups of intervals. One group enforces the law of excluded middles; the second one encodes the literals of the formula; and the third encodes the clauses of the formula.

The first group consists of the single interval **middle**. This interval determines the truth assignments for all other intervals: those that fall before **middle** correspond to false terms, and those that fall after correspond to true terms.

Turning to the second group of intervals, we create for each literal *P* in the formula, and its negation ¬*P*, a pair of intervals, **P** and **notP**. These intervals are then related to **middle** by creating the middle-excluding interval **PXnotP** (for *P* excludes ¬*P*), and asserting:

middle	*(DURING)*	**PXnotP**
P	*(MEETS MET-BY)*	**PXnotP**
notP	*(MEETS MET-BY)*	**PXnotP**
P	*(BEFORE AFTER)*	**notP**

The effect of the second and third assertions is to cause **P** and **notP** to either meet or be met by **PXnotP**. The fourth assertion makes this choice mutually exclusive. Since any interval preceding **middle** is taken to be false, the first assertion ensures the falseness of whichever of **P** and **notP** ends up meeting **PXnotP**. The other of the two will be met by **PXnotP**, and hence follow **middle** and be true.

The encoding of the formula's clauses is handled by a third group of intervals, and proceeds as follows. For each clause *P ∆ Q ∆ R*, we create a clausal interval **PorQorR** which is used to impose a truth assignment on the literals of the disjunct. The key to this encoding is that no more than two of the literals' intervals are allowed to precede **middle** (and be false). This guarantees that one of the literals' intervals must follow **middle**, and hence be true, and hence cause the clause to be satisfied. This encoding is accomplished through place-holder intervals **forP**, **forQ**, and **forR**, one set of which is generated for each clause. The following assertions are made of the place-holders.

Place-holders contain their literals:

forP	*(CONTAINS)*	**P**
forQ	*(CONTAINS)*	**Q**
forR	*(CONTAINS)*	**R**

Place-holders must be true or false:

forP	*(BEFORE AFTER)*	**middle**
forQ	*(BEFORE AFTER)*	**middle**
forR	*(BEFORE AFTER)*	**middle**

At most two false place-holder positions:

forP	*(MEETS STARTS AFTER)*	**PorQorR**
forQ	*(MEETS STARTS AFTER)*	**PorQorR**
forR	*(MEETS STARTS AFTER)*	**PorQorR**

Placement of the clausal interval:

PorQorR	*(CONTAINS)*	**middle**

Place-holders don't overlap:

[2] 3-SAT formulæ are of form (A ∆ B ∆ C) □…□ (X ∆ Y ∆ Z)

forP *(BEFORE AFTER MEETS MET-BY)* **forQ**
forQ *(BEFORE AFTER MEETS MET-BY)* **forR**
forR *(BEFORE AFTER MEETS MET-BY)* **forP**

The first group of assertions relates the place-holders to their literals. The second, third, and fourth groups of assertions ensures that a place-holder interval can only be in one of two positions on the false side of **middle**. The fifth group makes the place-holders mutually exclusive, and guarantees that only one of the place-holders can be in each of the allowed false positions.

To complete the proof, we note that the interval encoding of the 3–SAT formula can clearly be performed in time polynomial in the length of the formula. From the preceding discussion, it also follows that the formula is satisfiable just when its encoding as interval assertions is satisfiable too. Since 3-SAT is NP-complete, it follows that determining satisfiability of assertions in the interval algebra is in turn NP-hard.

This NP-hardness result can be strengthened somewhat to NP-completeness by the following proposition.

Theorem 3: Determining the satisfiability of a set of assertions in the interval algebra is in NP, and is hence NP-complete.

Proof: To show that satisfiability of a set of interval assertions is in NP, we only need show that we can guess an interpretation for the assertions and then verify it in polynomial time. To construct the interpretation, we just choose a random ordering of the intervals' endpoints, possibly making some of them the same. To verify the interpretation we just check that the original assertions are satisfied by the ordering. If we started with *n* intervals, there will be $O(n^2)$ assertions to check, each of which is verifiable in constant time. Interval satisfiability is thus in NP, and being NP-hard, it is thus NP-complete.

The following theorem extends the NP-completeness result for the problem of determining satisfiability of assertions in the interval algebra to the problem of determining closure of these assertions.

Theorem 4: The problems of determining the satisfiability of assertions in the interval algebra and determining their closure are equivalent, in that there are polynomial-time mappings between them.

Proof: First we show that determining closure follows readily from determining consistency. To do so, assume the existence of an oracle for determining the consistency of a set of assertions in the interval algebra. To determine the closure of the assertions, we run the oracle thirteen times for each of the $O(n^2)$ pairs *<i,j>* of intervals mentioned in the assertions. Specifically, each time we run the oracle on a pair *<i,j>*, we provide the oracle with the original set of assertions and the additional 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 three preceding theorems demonstrate that computing the closure of assertions in the interval algebra is NP-complete. This result casts great doubts on the computational tractability of the algebra, as no NP-complete 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 [1987] 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 (worst-case) exponential *Simplex* algorithm. As we shall see below, the fragment that they consider is actually tractable, and the expected performance of *Simplex* would be polynomial for their application.

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-complete! There are several strategies that can be adopted to put the algebra to work in practical systems. The first is to cluster intervals into small self-contained groups, with limited interconnection between the clusters. Within a cluster, 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 performance of their *Simplex*-based system. More recently, Koomen [1989] has developed algorithms for *automatically* clustering intervals according to Allen's reference interval strategy. Unfortunately, clustering techniques such as these are most applicable only in domains with little global interconnectivity between time intervals.

Another overall 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.

Restricting the Interval Algebra

An alternative approach to containing the computational cost of interval reasoning is to consider fragments of the full interval algebra for which closure is tractable. These fragments may be naturally matched to certain problems.

One fragment of which this is true arises in the context of relating the duration of events observed to occur in the world. This class of problems imposes a significant reduction in the degree of representational ambiguity that is required of the interval algebra. Indeed, assuming that one is simultaneously observing several events as they occur, representational ambiguity only arises as a result of one's inability to resolve the exact duration of each event. In terms of the interval algebra, this corresponds to an inability to resolve the relative position of two intervals' endpoints. Returning to an earlier example, while observing John eating his breakfast, we might have clearly seen him opening his newspaper before starting his coffee. However, we might not be able to tell whether he was done reading before he started his coffee, along with starting it, or afterwards (see Figure 2).

Viewing measurement uncertainty of this type in terms of interval endpoint uncertainty allows us to produce an algebraic encoding of the phenomenon. This kind of encoding is especially of interest in the context of qualitative reasoning, as qualitative applications favor these kind of algebraic methods. We should also note that in formalizing the encoding, we must ensure that it has the property that uncertainty be continuous in the following sense. Although we can place upper and lower bounds on when we may have observed an event to start or end, we typically can not exclude any time within that range as a possible start or end point. For example, we can't exclude John's coffee drinking from starting at the same time as his paper reading finishes. As we shall see below, the class of interval relations that display this continuous endpoint uncertainty has a tractable closure algorithm. Before proving this result, however, we first need to consider time points per se.

Time Point Algebras

For our purposes, time points can be modelled by the real numbers, and their relations can be entirely expressed as inequalities. As with intervals these relations can be decomposed into disjunctions of simple relations, which in this case number three: <, =, and >. Seven consistent vectors can then be formed : *(<)*, *(=)*, *(>)*, *(< =)*, *(= >)*, *(< = >)*, and *(< >)*. Abusing notation, we will refer to these vectors as <, =, >, ≤, ≥, *?*, and ≠ respectively. Again, as with intervals, the algebra of points supports an addition and a multiplication. These operations are defined by the following two tables, in which the *0* entry represents the inconsistent vector.

+	<	≤	>	≥	=	≠	?
<	<	<	0	0	0	<	<
≤	<	≤	0	=	=	<	≤
>	0	0	>	>	0	>	>
≥	0	=	>	≥	=	>	≥
=	0	=	0	=	=	0	=
≠	<	<	>	>	0	≠	≠
?	<	≤	>	≥	=	≠	?

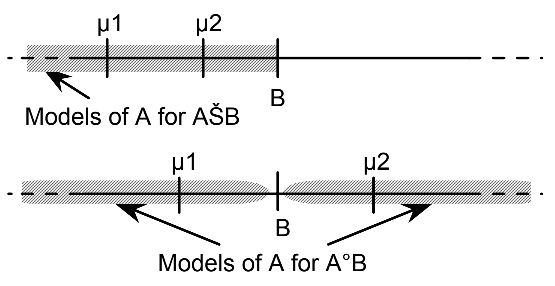

Figure 5: Models of point relations.

x	<	≤	>	≥	=	≠	?
<	<	<	?	?	<	?	?
≤	<	≤	?	?	≤	?	?
>	?	?	>	>	>	?	?
≥	?	?	>	≥	≥	?	?
=	<	≤	>	≥	=	≠	?
≠	?	?	?	?	≠	?	?
?	?	?	?	?	?	?	?

As with 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*.

We can define a property of a subset of the time point algebra which is of particular interest in formalizing the restricted interval algebra presented above. We say that the relation between two time points is continuous if the set of models that it admits for each time point, as a function of the other, is convex. The models of a time point are a set of real numbers, and for this set to be convex, the range of numbers between any two models must also all be models. For example, if we have $A \leq B$, the models of *A* are all the real numbers less than or equal to *B*. Take any two of these models, calling them μ_1 and μ_2. The range of real numbers between them are also all less than or equal to *B*, and are hence also models of *A*. In contrast, if we have $A \neq B$, *A*'s models are all the reals except *B*: $\{ x \mid x < B \} \approx \{ x \mid x > B \}$ This set is not convex, since one can pick two models of *A*, $\mu_1 = B - \partial$ and $\mu_2 = B + \partial$ for example, which span the gap imposed by *B*'s exclusion. Since *B* is a real number in the range between μ_1 and μ_2, but is not a model of *A*, the models of *A* are not convex . See Figure 5.

This property of continuity is true of all relations in the point algebra except ≠. What is more, the point algebra restricted to continuous relations is in turn an algebra with a well-defined addition and multiplication. We will refer to this algebra as the continuous point algebra.

Continuous Endpoint Uncertainty

Having developed these properties of time points, we can now return to formalizing the restricted interval algebra we discussed above. Recall that we were interested in interval relations that could be modelled by a continuous uncertainty in the relationship of their endpoints. This property can in fact be characterized in terms of continuous point relations. To this extent, we define the continuous endpoint algebra as that subset of the interval algebra which can be entirely encoded as conjunctions of continuous time point relations between the endpoints of intervals.

This algebra includes relations such as the vector *(BEFORE MEETS OVERLAPS)* from the coffee and newspaper example of Figure 2. The relation between these intervals is described by the conjunction of the following point relations, in which (e.g.) coffee^- and coffee^+ denote the start point and end point of interval *coffee* respectively.

$$\begin{array}{lcl} \mathit{paper}^- & < & \mathit{paper}^+ \\ \mathit{paper}^- & < & \mathit{coffee}^- \\ \mathit{paper}^+ & ? & \mathit{coffee}^- \\ \mathit{paper}^+ & < & \mathit{coffee}^+ \\ \mathit{coffee}^- & < & \mathit{coffee}^+ \end{array}$$

Interval relations precluded from the restricted algebra include, for example *(BEFORE OVERLAPS)*. Indeed, encoding *(BEFORE OVERLAPS)* in terms of interval endpoints requires the non-continuous point relation ≠:

(BEFORE OVERLAPS)

$$+\ \lambda\, x,y.\quad (x^- < x^+)\square(x^- < y^-)\square(x^+ \neq y^-)\square (x^+ < y^+)\square(y^- < y^+)$$

Note that this restriction on what can be expressed with continuous endpoint uncertainty matches the intuitive requirements we gave above for our encoding of measurement uncertainty.

Another class of interval relations that lie outside the scope of the continuous endpoint algebra are the truly disjunctive relations such as*(BEFORE AFTER)*. This particular example can not even be encoded as a conjunction of non-continuous point relations. Indeed to model *(BEFORE AFTER)* with interval endpoints requires explicit disjunction:

(BEFORE AFTER)

$$+\ \lambda\, x,y.\quad (x^+ < y^-)\ldots\Delta\ldots(x^- > y^+)$$

The proof of NP-hardness for deductive closure in the interval algebra was critically dependent on just such disjunctions of interval relations as these. Since the continuous endpoint algebra precludes all but the simplest forms of disjunction, it is natural to ask whether computing closure in the continuous endpoint algebra is in fact tractable. As we alluded to above, the following theorem is true.

Theorem 5: The constraint propagation algorithm of Figure 4 computes the closure of assertions in the continuous endpoint algebra.

A proof of this theorem appears in [van Beek 1989] and is elaborated in [van Beek & Cohen 1989]. What follows is a model-theoretic variant of the proof.

Given a set of intervals I, whose relations are initially described by a set of assertions R, we need to show that the algorithm computes the minimal relation between all *i* and *j* in I, given R. The proof notes that the intervals in I and the relations computed to hold between them form a graph $\mathcal{G}$, in which the vertices are intervals and the arcs are relations. Underlying the proof is a notion of graph consistency adapted from Freuder [1982]. We say that an interval graph $\mathcal{G}$ is k-consistent if given any consistent assignment of k-1 of its intervals to pairs of real numbers, there is a consistent assign-ment of any kth interval. The first assignment is an interpretation of $\mathcal{G}_{k-1}$, the subgraph of $\mathcal{G}$ defined by the k-1 intervals, and the second is an interpretation of $\mathcal{G}_k$, the subgraph defined by all k intervals. Continuing, we define *strong* k-consistency as j-consistency for all j≤k.

An important consequence of this definition is that any strongly k-consistent graph has the following property. Take $\mathcal{G}_k$, a k-sized subgraph of $\mathcal{G}$ containing intervals i and j. Then for any model of i and any model of j which are consistent with the intervals' relation in $\mathcal{G}_k$, there is an interpretation of $\mathcal{G}_k$ which maps i and j to these models. Now, given any simple relation holding between i and j, we can always construct models of i and j which satisfy the relation, and which thus appear in some interpretation of $\mathcal{G}_k$. In the case where k=n, the size of $\mathcal{G}$, this is equivalent to saying that any simple relation on any arc between any two intervals in $\mathcal{G}$ appears in some interpretation of $\mathcal{G}$. That is, if $\mathcal{G}$ is strongly n-consistent, then the relations holding between its intervals are minimal and closure of its premises has been computed.

We now use this property to prove Theorem 5, by showing that the constraint propagation algorithm computes k-consistency on an interval graph of size n, for all k≤n.

Proof (of theorem 5): The proof is by induction on the size of subgraphs of an interval graph $\mathcal{G}$.

Basis: k=1, 2, or 3. It is clear that for k=1 (single intervals) or k=2 (pairs of intervals), any labelling of the arcs is k-consistent. For k=3, we note that Mackworth [1977] and Montanari [1974] have shown that algorithms equivalent to that in Figure 4 achieve 3-consistency.

Induction: Assuming the graph is strongly (k-1)-consistent, we need to show that it is also k-consistent for any k such that 4≤k≤n. That is, given an interpretation for $\mathcal{G}_{k-1}$, a (k-1)-sized subgraph of $\mathcal{G}$, and given any interval k not in $\mathcal{G}_{k-1}$, we need to find a model of k that yields an interpretation for $\mathcal{G}_{k}$, the graph produced by expanding $\mathcal{G}_{k-1}$ to include k.

To do so, we note that the interpretation we were given for $\mathcal{G}_{k-1}$ assigns a model to each of its constituent intervals i. Given such a model, the relation between i and k constrains the possible models of k to forming a set in $\leftarrow^2$. The set is in $\leftarrow^2$ because the models of k are pairs of reals. To produce our desired model of k, we need to show that all the constraint sets have some model in common.

We proceed by noting that the relation between any i in $\mathcal{G}_{k-1}$ and k is from the continuous endpoint algebra, and so encodes one or more continuous point relations between the intervals' endpoints. The models of k induced by each such endpoint relation $\mathcal{R}_{i\pm,k\pm}$ are thus convex sets. For example, the endpoint relation $i^+ < k^-$ induces the following convex set of models for k:

$$\{ <x,y> \; □ \leftarrow^2 \mid x > i^+ \text{ and } y \; □ \leftarrow \}$$

The models of k being convex allows us to apply a theorem of linear programming due to Helly [Chvátal 1983]. Stating the theorem, let F be a finite family of at least n+1 convex sets in $\leftarrow^n$, where every n+1 sets in F have a common point (in $\leftarrow^n$). Then all sets in F have a common point.

The model sets for k are in $\leftarrow^2$ (i.e., n=2 in Helly's theorem), so to prove Theorem 5 we only need show that any three model sets for k have a common point. This is equivalent to showing that their three corresponding endpoint constraints are consistent. There are two cases depending on whether one of the constraint sets corresponds to $k^- < k^+$.

Case 1: The three constraints have form $i^\pm \; \mathcal{R}_{i\pm,k-} \; k^-$, $k^- < k^+$, and $k^+ \; \mathcal{R}_{k+,j\pm} \; j^\pm$. Note that these three constraints define a subgraph of size 3 which is 3-consistent by virtue of having run the algorithm. Hence the three constraints are consistent and admit some common model for k.

Case 2: The constraints are all of form $i^\pm \; \mathcal{R}_{i\pm,k\pm} \; k^\pm$. In this case, we note that the constraints are in fact over $\leftarrow$, not $\leftarrow^2$, and hence we only need to show that any two of them are consistent. Again, they define a subgraph of size 3, are consistent by virtue of running the algorithm, and thus admit some common model for k.

To summarize what we have shown: (1) Any three endpoint constraints on k admit a common model of k. (2) Hence all endpoint constraints on k admit a common model. (3) Hence for any given interpretation of $\mathcal{G}_{k-1}$, we can construct an interpretation of $\mathcal{G}_{k}$. (4) Hence $\mathcal{G}$ is k-consistent, which proves the induction step and thus the theorem.

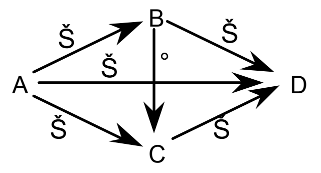

Figure 6: 3- but not 4-Consistent Points

Additional Results

In the preceding discussion, time points have primarily been of interest in formalizing and defining a restricted fragment of the interval algebra. But the full point and the continuous point algebras are of interest too, and we summarize here some results concerning them.

As can be expected, the constraint propagation algorithm of Figure 4 computes closure in the point algebra. The proof is similar to that of closure for the continuous endpoint algebra (Theorem 5), but simpler, since it does not involve translations from intervals to points. Details can be found in [van Beek & Cohen, 1989].

Given this, it is natural to ask whether the algorithm also computes closure in the full point algebra. In fact, it does not. The full point algebra includes the ≠ relation which is not continuous (see Figure 5). This discontinuity is exploited by a counterexample presented as a point relation graph in Figure 6. The graph can be produced by asserting $A \le B$, $B \le D$, $A \le C$, $C \le D$, and $B \neq C$. The algorithm makes the graph 3-consistent by leaving these five relations untouched and inferring $A \le D$.

Although this graph is 3-consistent it fails to achieve 4-consistency. Intuitively, this is so because one of the options allowed by 3-consistency is that *A* and *D* are equal, which does not “leave room” between them for the disequal points *B* and *C*. More formally, say points *A* and *D* are equal, so *A*, *B*, and *D* are assigned the same model, some real number μ. The constraints between these points and C, $A \le C$, $C \le D$, and $B \neq C$ respectively admit the following sets of models for *C*: (-∞, μ], [μ, +∞), and (-∞, μ) ≈ (μ, +∞). Pairwise, these sets have non-empty intersections, as reflected in the 3-consistent labelings of the arcs of the graph. However, their common intersection is empty, and hence there can be no interpretation of the premises of the example in which *A* and *D* are equal.

To make the graph in the example 4-consistent and close its premises, the relation between *A* and *D* would have to be <. It can be shown [van Beek 1989, van Beek & Cohen 1989] that achieving 4-consistency in fact computes closure for the full point algebra. This can be accomplished through a variant of the algorithm of Figure 4 which operates in $O(n^4)$ time. Alternatively, Ladkin & Maddux [1988] show that satisfying a 3-consistent full point graph can be accomplished in $O(n^2)$ time. This process maps the graph onto a model (if it has one) and can thus be used to tell if a 3-consistent graph has no interpretation.

Additionally, we should note that just as the continuous point algebra induces the continuous endpoint algebra on intervals, the full point algebra defines an algebra of *pointisable* interval relations (Ladkin & Maddux' term). This algebra is a proper superset of the continuous endpoint algebra, and includes such relations as *(BEFORE OVERLAPS)* which can't be expressed in the continuous endpoint algebra. Closure in the pointisable interval algebra can be computed with the same 4-consistency algorithm as is used for the full point algebra (again see van Beek's articles).

Finally, we should note that unlike the continuous endpoint algebra, the pointisable algebra is not motivated, in the authors' minds, by a broad class of problems such as measurement uncertainty. Although such problems surely must exist, we do not have any good characterization of them at this time.

Applying Temporal Representations

The need for explicit temporal representations in AI applications, though widely acknowledged, is also widely finessed. Few applications actually incorporate explicit reasoning about time, relying instead on heuristic representational short cuts.

Among applications which do use explicit temporal reasoning, a significant number use representations isomorphic to the continuous endpoint algebra. This is the case, for example, with Simmons' geological reasoning program [Simmons 1983]. This fragment of the interval algebra is also the one used by Malik and Binford [1983] in their spacio-temporal reasoning program. In their case, though, reasoning is performed with linear programming tecniques (in particular, the *Simplex* algorithm). While linear programming algorithms may be useful for deriving conclusions from a fixed set of temporal assertions, constraint propagation is proba-bly more appropriate for domains where constraints are added incrementally.

Although many applications may be able to restrict their interval temporal reasoning to a tractable fragment of Allen's algebra, others may not. One program that requires the full interval algebra is the planning system of Allen and Koomen [1983] in which the time extent of actions is modeled with intervals. A basic operation of the planner is to require of two actions that they be non-overlapping. This is accomplished by restricting their temporal relation to being:

(BEFORE MEETS MET-BY AFTER)

As we noted above, disjunctive relations such as this fall outside of the tractable fragment of the interval algebra. As a result, this planning architecture requires one to consider completeness issues directly, either by relying on approximate algorithms, by invoking an exponential temporal reasoner, or by applying planning-specific knowledge about the ordering of actions.

Another area of research in which temporal reasoning plays an important role is the semantics of natural language. Understanding event references in language is in fact one of the original motivations for Allen's algebra (see [Allen 1984]). Song and Cohen [1988] use a subset of the continuous endpoint algebra to capture the temporal relations between events in a narrative.

In closing, we should note that the importance of considering specific applications in the context of temporal reasoning is in the constraints they place on the general representation problem. It is in this interaction between application areas and knowledge representation that new representation areas are defined, and new questions are formulated.

Acknowledgements

This research was supported in part by the Defense Advanced Research Projects Agency of the United States under contracts N00014-85-C-0079 and N00014-77-C-0378, and by the Natural Sciences and Engineering Research Council of Canada.

References

Allen, J. F. (1983). Maintaining Knowledge about Temporal Intervals. *Communications of the ACM* 26(1): 832-843.

Allen, J. F. (1984). Towards a General Theory of Action and Time. *Artificial Intelligence* 23(2): 123-154.

Allen, J. F. & Hayes, P. J. (1985). A Common-Sense Theory of Time. In *Proceedings of the Ninth IJCAI*, Los Angeles, Calif., pp. 528-531.

Allen, J. F. & Koomen, J. A. (1983). Planning Using a Temporal World Model. In *Proceedings of the Eighth IJCAI*, Karlsruhe, W. Germany, pp. 741-747.

Chvátal, V. (1983). *Linear Programming*. New York: W. H. Freeman and Company.

Freuder, E. C. (1982). A Sufficient Condition for Backtrack-Free Search. *Journal of the ACM* 29: 24-32.

Koomen, J. A. (1989). Localizing Temporal Constraint Propagation. In *Proceedings of KR 89*, pp. 198-202.

Ladkin, P. B. (1987). Models of Axioms for Time Intervals. In *Proceedings of the Sixth AAAI*, Seattle, Wash., pp 234-239.

Ladkin, P. B. & Maddux, R. (1988). On Binary Constraint Networks. Technical Report, Kestrel Institute, Palo Alto, Calif.

Mackworth, A. K. (1977). Consistency in Networks of Relations. *Artificial Intelligence* 8: 99-118.

Montanari, U. (1974). Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Science* 7: 95-132.

Malik, J. & Binford, T. O. (1983). Reasoning in Time and Space. In *Proceedings of the Eighth IJCAI*, Karlsruhe, W. Germany, pp. 343-345.

Simmons, R. G. (1983). The Use of Qualitative and Quantitative Simulations. In *Proceedings of the Third AAAI*, Washington D.C., pp. 364-368.

Song, F. & Cohen, R. (1988). The Interpretation of Temporal Relations in Narrative. In *Proceedings of the Seventh AAAI*, Saint Paul, Minn., pp. 745-750.

Valdés-Pérez, R. E. (1987). The Satisfiability of Temporal Constraint Networks. In *Proceedings of the Sixth AAAI*, Seattle, Wash., pp 256-260.

van Beek, P. (1989). Approximation Algorithms for Temporal Reasoning. In *Proceedings of the Tenth IJCAI*, Detroit, Mich.

van Beek, P. & Cohen R. (1989). Approximation Algorithms for Temporal Reasoning, Research Report CS-89-12, Dept. of Computer Science, U. of Waterloo.

Vilain, M. B., & Kautz, H. (1986). Constraint Propagation Algorithms for Temporal Reasoning. In *Proceedings of the Fifth AAAI*, Philadelphia, PA, pp.377-382.