From: AAAI-87 Proceedings. Copyright ©1987, AAAI (www.aaai.org). All rights reserved.

THE CONSISTENT LABELING PROBLEM IN TEMPORAL REASONING

Edward P K TSANG
Department of Computer Science
University of Essex
Colchester CO4 3SQ, U K

ABSTRACT

Temporal reasoning can be performed by maintaining a temporal relation network, a complete network in which the nodes are time intervals and each arc is the temporal relation between the two intervals which it connects. In this paper, we point out that the task of detecting inconsistency of the network and mapping the intervals onto a date line is a Consistent Labeling Problem (CLP). The problem is formalized and analyzed. The significance of identifying and analyzing the CLP in temporal reasoning is that CLPs have certain features which allow us to apply certain techniques to our problem. We also point out that the CLP exists when we reason with disjunctive temporal relations. Therefore, the intractability of the constraint propagation mechanism in temporal reasoning is inherent in the problem, not caused by the representation that we choose for time, as [Vilain & Kautz 86] claims.

I Introduction

Temporal reasoning has recently been the subject of great attention in AI. Natural language understanding systems like [Bruce 72], [Kahn & Gorry 77], etc. and planning systems like DEVISER [Vere 83], TIMELOGIC [Allen & Koomen 83], ISIS [Fox & Smith 84] FORBIN [Dean 85][Miller et al. 85], TLP [Tsang 86b,87a], etc. all in some sense model and reason with time. [Allen 83] suggests modeling time in an interval-based temporal structure. He also presents a formalism for reasoning with disjunctive temporal relations among intervals. In this paper, we shall start by looking at Allen's formalism. Then we shall identify the consistent labeling problem (CLP) in it, and show that this problem exists in point-based approaches as well, whenever we reason with all disjunctive temporal relations at the same time. The significance of identifying the CLP will also be discussed.

II Temporal Reasoning by maintenance of a relation network

In Allen's temporal frame, each assertion is associated with an interval in which it holds. Intervals and their temporal relations can be represented by a complete simple graph which is called a **Relation Network**:

$$G = (N, R)$$

where N is a finite set of intervals (which form the nodes of G) and R is a set of temporal relations (which form the arcs). Between any two nodes X and Y in N, there exists an arc in R which goes from X to Y and another arc which goes from Y to X (hence G is complete). For convenience, we use **Rxy** to represent the temporal relation between intervals X and Y throughout this paper. Ryx is just the inverse relation of Rxy (since Rxy and Ryx must coexist, G is a simple graph). We follow [Allen 83] and use the following notations for primitive temporal relations: (<,m,o,f,d,s,=,si,di,fi,oi,mi,>). Disjunctive primitive relations are represented by a list. For example X [< = >] Y means X is *before, equal to* or *after* Y. For all intervals i and j, if Rij is completely unconstrained, it can take any one of the 13 primitive relations as its value. Every arc Rxy in R must take one of the primitive relations as its value.

Temporal relations are subject to constraints. A **temporal constraint** on Rxy is a restriction on the values that Rxy can take. Therefore, a temporal constraint C can be seen as a set of primitive temporal relations — an enumeration of all the values that the subject temporal relation can take in order to satisfy C. For example, if the proposition P holds in interval X, and ¬P holds in interval Y, then X and Y must not have any common subintervals. In other words, the constraint is: Rxy ∈ {< m mi >}. Because of the linearity property of time in this logic [Tsang 86a], for any three intervals X, Y and Z, the temporal relation Rxz is restricted by Rxy and Ryz jointly. Such constraints are called **transitivity rules**. A constraint propagation algorithm based on these transitivity rules has been presented in [Allen 83].

[Tsang 86b] points out the need for checking consistency in relation networks. In planning, there is a need to map intervals onto *date lines*, simple structures where each time point has a place, and the points are linearly ordered. One way to prove the consistency of a relation network and map the intervals in it onto a date line is to assign a primitive temporal relation to each relation. This is a consistent labeling problem, which will be discussed below.

III The Consistent Labeling Problem (CLP)

A Consistent Labeling Problem (CLP) is defined as follows:

> We have a finite set of variables $Z = \{X_1, X_2, ..., X_n\}$. Cardinality of Z is *n*. Each variable X_i in Z has a finite domain of values. Constraints exist for subsets (of various sizes) of variables in Z. The task is to find a **solution-tuple** (which is a *n*-tuple), which means the assignment of one value to each of the variables in Z such that all the constraints are satisfied.

This problem is called Constraint Satisfaction Problem in some of the literature. In some applications, the task is defined as finding all solution-tuples. We call the assignment of a value to a variable a **label**. For example $<X_i, V_i>$ is a label assigning V_i to X_i. A compound **label** is the combination of more than one label, e.g. $(<X_1, V_1><X_2, V_2>...<X_k, V_k>)$. A **k-constraint** (denoted by C_k, where $1 \leqslant k \leqslant n$) is a mapping of k labels to {true, false}. The label $(<X_1, V_1>...<X_i, V_i>)$ is **admissible** if $C_k(<X_1, V_1>...<X_i, V_i>)$ is mapped to *true*.

For example, the 8-queens problem can be formulated as a CLP: The problem is to place 8 queens on a (8 rows × 8 columns) chess board, subject to the constraint that no two queens appear on the same row, column or diagonal. The 8 rows can be seen as variables X_1 to X_8. Each of them can take an integer value between 1 to 8. X_i taking the value k indicates that the queen in row i is placed on column k. Between each two variables X_i and X_j, the following binary constraints apply:

(1) $V_i \neq V_j$
(2) $V_i + (j-i) \neq V_j$
(3) $V_i - (j-i) \neq V_j$

Most research in the CLP concerns binary constraints. [Freuder 78] introduces the concepts of **k-satisfiability** and **k-consistency**, which apply to general CLPs with constraints of arbitrary arity. A network is k-satisfiable if for any k variables in the network, there exists a compound label on them which satisfies all the constraints amongst them. A constraint network being *k-consistent* implies that [Freuder 82]:

> Choose any set of k-1 variables. If L is a compound label on these variables which satisfies all the constraints on them, then for any kth variable that we choose, there exists a value that this variable can take such that the label of the kth variable together with L satisfy all the constraints on the k variables.

If a constraint network with n variables is n-consistent, a solution tuple exists. Other research on CLPs concerning constraints of arbitrary arity can be found in [Nudel 83][Nadel 85].

IV The CLP in temporal reasoning

The problem of assigning a primitive relation to each temporal relation is a CLP. In this problem, the constraint network (CN) is:

$$CN = (R, T)$$

The set of nodes of CN is R, the set of arcs in the relation network G mentioned in section II. T is the set of constraints on elements of R. Since constraints could have any arity, it is difficult to draw the constraint network graphically. The domain of each variable is the set of all primitive temporal relations, which we call PR:

$$PR = \{<,m,o,fi,di,s,=,si,d,f,oi,mi,>\}$$

For example in some relation network G_0, let the set of nodes N_0 be {A, B, C}. The arcs in G_0 would be R_0 = {Rab, Rba, Rbc, Rcb, Rac, Rca}, which are the nodes of G_0's corresponding constraint network.

T in CN consists of constraints of various arities on the temporal relations in R. The example "X and Y must not have any common subintervals" mentioned above is a unary constraint on Rxy. An example of a binary constraint is: "if A *meets* B, then C *meets* D", which is equivalent to "Rab ∈ {m} —> Rcd ∈ {m}". An example of a 3-ary constraint is:

> "intervals P, Q, and R must not have any common subintervals"

which means:

> Rpq ∈ {< m mi >} V Rqr ∈ {< m mi >} V
> Rpr ∈ {< m mi >}

Each transitivity rule is in fact a set of constraints on labels which have the form:

$$(<Rab,r_{ab}><Rbc,r_{bc}><Rac,r_{ac}>)$$

(Notice that the three labels concern the relations of exactly three intervals). Here the value of Rac is restricted by the values of Rab and Rbc together. Examples of constraints implied by the transitivity rules are:

C(<Rab,m><Rbc,=><Rca,mi>) —(mapped to)—> *true*
C(<Rab,m><Rbc,m><Rca,m>) —(mapped to)—> *false*

For example in the above relation network G_0, these might be the constraints "interval A precedes both intervals B and C, and B and C must start at the same time". In this case, the set of constraints on G_0's corresponding constraint network is:

Rab ∈ {< m} (which implies Rba ∈ {mi >})
Rbc ∈ {s = si} (which implies Rcb ∈ {s = si})
Rac ∈ {< m} (which implies Rca ∈ {mi >})

plus the transitivity rules.

In planning, the temporal relations labeling problem exists only if we do not want to commit ourselves to any primitive relations until we need to do so (i.e. if we apply the least-commitment strategy). In such approaches, building up the relation network (identifying the intervals involved in the problem) and labeling the temporal relations are performed in two separate stages (see [Tsang 86b]). An alternative approach is to label all the temporal relations whenever new intervals are added to the relation network, and backtrack if overall inconsistency is detected. This approach is adopted by planners like NONLIN [Tate 77]. In NONLIN, only temporal relations *before* and *after* are considered: if two actions A and B conflict with each other, a commitment is made to either A *before* B or A *after* B. This approach labels temporal relations before the whole CLP is formulated.

One constructive way to prove the satisfiability of a constraint network is to find a solution-tuple for it. However, this is a NP-complete problem as the search space is exponential in the number of nodes in the constraint network. [Freuder 78] presents an algorithm for finding the set of all solution tuples without needing any searching and backtracking. However, this algorithm takes exponential time and space, and therefore, as Freuder admits, is not useful for practical applications [Freuder 82].

V Specific characteristics of CLPs in general

CLP has specific characteristics in which it differs from general search problems. Some important characteristics of CLPs are:

1. The size of the search space is fixed and finite.
   Assume that there are n variables to be labeled. If we order these variables, the search space can be represented by a tree. Each node of this tree represents the choice point of assigning a value to a variable, and each branch represents the commitment of a label. The depth of this search tree is n and the branching factor of each level is $|d_i|$, where $|d_i|$ is the cardinality of the domain of the variable X_i. The number of leaves of the search tree is:
   $$\Pi_{i=1}^{n} (|d_i|)$$
2. The subtrees under each branch are very similar.
   Assume that the variables are ordered, and X_i, X_j are variables. The same choices of labels for X_j would be available under each branch of X_i, where $i<j$. Constraint propagation may prune some future branches if we use lookahead search strategies. But basically the subtrees are very similar.
3. Choice of a value for a variable propagates through the constraints and might affect the choices of values for other variables.

Because of these characteristics, specific heuristics can be used in the search of solution tuples. Some of them, e.g. lookahead, are summarized in [Haralick & Elliott 80].

VI Search strategies in temporal relations labeling

In searching for solution tuples, at least three orderings have to be decided:

1. Which variable to label next?
   [Freuder 82] presents an algorithm for finding *minimal order graphs*. The basic idea is to order the nodes in the constraint network so that those which have more constraints linked to them are labeled first. By doing so, one can minimize unnecessary commitments. However, this algorithm applies to binary constraint problems only. In the temporal relations labeling problem, every temporal relation is constrained by the same number of transitivity rules. Hence, it is likely that most orderings form a minimal order graph.

   [Haralick & Elliott 80] introduces the *Fail First Principle*. One of the applications of this principle is to label the nodes which have the fewest available labels first. Doing so would minimize the size of the search tree. This principle is applicable to the temporal relations labeling problem.
2. Which value to try next?
   Having decided which variable to label next, we have to choose which of the available values to try next. One heuristic is to try the least restrictive value first, in the hope that unnecessary backtracking can be avoided. Ordering of the values according to their restrictiveness is normally domain-dependent. In the temporal relations labeling problem, primitive temporal relations can be ordered by their restrictiveness. The order is shown below, with the less restrictive relations at the top:

   1. [< >]
   2. [o oi]
   3. [d di]
   4. [m mi]
   5. [fi s si f]
   6. [=]

   A primitive temporal relation between two intervals is more restrictive if it requires more start/end-points of them to be equal. By trying the least restrictive available relation first, there is less chance of having to backtrack.

   However, we sometimes want to pack the intervals as tightly as possible. For example, in planning problems one may want to minimize the overall duration of the schedules generated. In this case, we might want to order the primitive temporal relations as follows:

   1. [=]
   2. [fi s si f]
   3. [o oi]
   4. [di d]
   5. [m mi]
   6. [< >]

   It is likely that the more the relations at the top of this sequence are used in the labeling, the more efficient that the resulting plan would be, though local optimality may not lead to global optimality. Finding optimal schedules (schedules which needs the least amount of time to finish) is a hard problem. This heuristic can only increase our chance of finding efficient plans.
3. Which inference to do next?
   The Fail First Principle suggests that those inferences which are most likely to fail should be performed first. However, there seem to be no general rules as to which inference is most likely to fail in this application domain. Anyhow, such rules will tend to vary from domain to domain.

In order to detect inconsistency at an earlier stage during the search, we can use a lookahead strategy. Looking ahead prevents us from rediscovering inconsistency repeatedly [Mackworth & Freuder 85]. Allen's algorithm in [Allen 83] can be used to maintain 3-consistency in the constraint network during the search.

VII Discussion

VII.1 Interval- versus point-based representation of time

Since points have strict linear ordering [Turner 84] [Tsang 86a], one might wonder whether the CLP still exists when we reason with points rather than intervals; in other words, in a point-based representation, could a constraint network which was locally consistent be unsatisfiable? If it could not, then why should we reason with intervals and get ourselves involved in the CLP?

Assume that we have a relation network of points:

$$G_p = (N_p, R_p)$$

The nodes (N_P) are points and each arc represents the relation of the temporal relation between its connecting points. If the network is totally unconstrained, the values that each arc can take is one of *before*, *equal* or *after*, which we denote by <, = and >.

The constraint network associated with G_P is:

$$CN_P = (R_P, T_P)$$

where T_P is the set of (3×3=) 9 transitivity rules on relations of points, e.g.

$$x < y \ \& \ y = z \ \rightarrow \ x < z$$

plus the problem-specific constraints on R_P.

One can prove that if T_P consists solely of unary constraints plus the transitivity rules, then CN_P is always consistent, provided that 3-consistency is maintained (unlike networks of intervals, see proof in [Tsang 87b]). However, we argue that:

> **IF** we reason with points, AND want to reason with disjunctive temporal relations,
> **THEN** we still have the CLP, which appears in a different form.

This can be illustrated by an example. Assume that we have the following interval-based relation network:

$$G_i = (N_i, R_i)$$

where N_i={A, B} and R_i={Rab} (A and B are intervals). (For simplicity, we treat Rba and Rab as the same element in R_i. This will not affect our discussion below.) We further assume that there exists a unary-constraint on Rab:

(I) A [< >] B

Associated with G_i is the constraint network:

$$CN_i = (R_i, T_i)$$

where T_i is the set of transitivity rules on intervals, together with (I). Let us find the point-based relation network:

$$G_P = (N_P, R_P)$$

and constraint network:

$$CN_P = (R_P, T_P)$$

which correspond to G_i and CN_i. Obviously,

N_P = {start(A), end(A), start(B), end(B)}

(I) in T_i means:

(I1) end(A) < start(B) ; OR
(I2) end(B) < start(A)

Among the 4 points, there are 6 binary temporal relations. (Again we treat Ryx as the same element as Rxy in R_P). Therefore R_P is the set of those 6 binary relations. Let D(x,y) represent the domain of the relation between points x and y. (For all x, y, D(x,y) = [< = >] if it is totally unconstrained.) Then by definition of an interval, we have the following unary-constraints in T_P:

(D1) D(start(A),end(A)) = [<]
(D2) D(start(B),end(B)) = [<]

A little reflection should convince the reader that (I1) and (I2) imply the following unary-constraints in T_P:

(D3) D(start(A),start(B)) = [< >]
(D4) D(start(A),end(B)) = [< >]
(D5) D(end(A),start(B)) = [< >]
(D6) D(end(A),end(B)) = [< >]

The constraint network CN_P now has:

T_P={(D1) to (D6) plus the 9 transitivity rules}

As said before, a CN_P of such form can always be labeled. However, one must note that this CN_P is not equivalent to the above CN_i. This CN_P allows relations that CN_i does not. For example:

start(A) < start(B) < end(B) < end(A)

is a consistent labeling in CN_P, but is not allowed in CN_i. The fact is, in order to represent CN_i by a point-based representation, we need to add to T_P the following binary constraints:

(C1) IF D(start(A),start(B))=[<]
THEN D(end(A),start(B))=[<]
(C2) IF D(start(B),start(A))=[<]
THEN D(end(B),start(A))=[<]
(C3) IF D(end(A),end(B))=[<]
THEN D(end(A),start(B))=[<]
(C4) IF D(end(B),end(A))=[<]
THEN D(end(B),start(A))=[<]

So to represent an interval-based constraint network which has:

> A set of unary-constraints: Dxy = [...],
> and 169 transitivity rules, which are 3-ary constraints.

In a point-based constraint network we need:

> a set of unary-constraints: D(x,y) = [...],
> and (3×3=) 9 transitivity rules (on <, = and >),
> and additional binary constraints like (C1) to (C4) above.

When binary constraints are added, the overall consistency of the constraint network is not guaranteed. One can translate any relation network from an interval-based representation to a point-based representation. But solving the CLP in one representation is as nontrivial as solving it in the other.

In fact, the above CLP exists only when we consider disjunctive temporal relations. Most implementations of point-based temporal reasoning modules consider one conjunctive set of temporal relation (among points) at a time, and therefore do not have to face this problem. [Vilain & Kautz 86] concludes that:

1. determining consistency of statements in Allen's interval algebra is NP-hard, and Allen's constraint propagation algorithm is incomplete;
2. constraint propagation in a "time point algebra" is complete.

where "time point algebra" refers to a point-based representation and its constraint propagation mechanism. Vilain & Kautz suggest that "the tractability of the point algebra makes it an appealing candidate for representing time". We feel that Allen's algebra and Vilain & Kautz's time point algebra cannot be compared in such a straightforward way because in Allen's formalism

disjunctive temporal relations are handled at the same time. Allen's constraint propagation algorithm is incomplete in the sense that it can only maintain 3-consistency, not overall consistency of the constraint network. But disjunctive relations among points are not handled at the same time in the time point algebra — when point A has to be *before* or *after* point B, the problem has to be treated as two separate problems. By avoiding reasoning with disjunctive relations, the time point algebra achieves completeness in the constraint propagation mechanism.

VII.2 Consideration of metric properties of time

In this paper, we have discussed temporal reasoning concerning relative temporal relations (e.g. before, meet, etc.). We must emphasize that a relation network in which consistent labeling exists may not be consistent with regard to the metric properties of time: constraints such as duration of intervals, absolute labels of starting or ending times. We believe that reasoning with metric properties is a nontrivial problem, and linear programming is a general tool for it. Discussion of this problem is beyond the scope of this paper, but see [Tsang 86b,87b].

VIII Summary

In this paper, we have identified and analyzed the CLP in temporal reasoning. We conclude that this CLP arises when we want to reason with disjunctive temporal relations, irrelevant to the choice between point-based or interval-based representation of time. Identifying and formalizing the CLP in temporal reasoning is significant because specific characteristics exist in CLPs which allow us to apply certain techniques for temporal labeling.

Acknowledgements

The author is indebted to Jim Doran and Sam Steel for many invaluable discussions on this topic. Chris Trayner, Richard Bartle, Anthony Cheng and John Bell give useful comments on this paper. This project is supported by the studentship fund of the Department of Computer Science, University of Essex.

REFERENCES

[Allen & Koomen, 83] J.F. Allen, & J.A. Koomen, *Planning using a temporal world model*, IJCAI-83, 741-747

[Allen, 83] J.F. Allen, *Maintaining Knowledge about Temporal Intervals*, CACM vol.26, no.11, November, 1983, 832-843

[Bruce, 72] B.C. Bruce, *A model for temporal references and its application in a question answering program*, AI 3(1972), 1-25

[Dean, 85] T. Dean, *Temporal Imagery: An Approach to Reasoning with Time for Planning and Problem Solving*, Ph D Dissertation, Yale University, October 1985

[Fox & Smith, 84] M.S. Fox, & S.F. Smith, *ISIS — A Knowledge-based system for factory scheduling*, Expert Systems, Vol.1 No.1, 1984, 25-49

[Freuder, 78] E.C. Freuder, *Synthesizing constraint expressions*, CACM November 1978, Vol 21, No 11, 958-966

[Freuder, 82] E.C. Freuder, *A sufficient condition for backtrack-free search*, J ACM Vol.29 No.1 January(1982), 24-32

[Haralick & Elliott, 80] R.M. Haralick, & G.L. Elliott, *Increasing tree search efficiency for constraint satisfaction problems*, AI 14(1980) 263-313

[Kahn & Gorry, 77] K.M. Kahn, & G.A. Gorry, *Mechanizing temporal knowledge*, AI, 9(1977)

[Mackworth & Freuder, 85] A.K. Mackworth, & E.C. Freuder, *The complexity of some polynomial consistency algorithms for constraint satisfaction problems*, AI 25(1985) 65-74

[Miller et al., 85] Miller, Firby & Dean, T., *Deadlines, Travel Time, and Robot Problem solving*, Manuscript, 1985, Yale University, (A shorter version of it appears in IJCAI-85, 1985, 1052-1054)

[Nadel, 85] B.A. Nadel, *The Consistent Labeling Problem, Part 1: Background and Problem Formulation*, The Univ of Michigan, Technical report CRL-TR-13-85, 1985

[Nudel, 83] B.A. Nudel, *Consistent-labeling problems and their algorithms: expected-complexities and theory-based heuristics*, AI 21, July 1983

[Tate, 77] A. Tate, *Generating project networks*, IJCAI-5, 888-893

[Tsang, 86a] E.P.K. Tsang, *The Interval Structure of Allen's Logic*, Technical Report CSCM-24, University of Essex, April, 1986

[Tsang, 86b] E.P.K. Tsang, *Plan Generation using a Temporal Frame*, ECAI-86, July, 1986

[Tsang, 87a] E.P.K. Tsang, *TLP — A Temporal Planner*, Proceedings, AISB-87, Edinburgh, April, 1987

[Tsang, 87b] E.P.K. Tsang, *Planning in a temporal frame: a partial world description approach*, PhD dissertation, University of Essex, in preparation, 1987

[Turner, 84] R. Turner, *Logics for AI*, Ellis Horwood series in Artificial Intelligence, 1984

[Vere, 83] S.A. Vere, *Planning in Time : windows and duration for activities and goals*, IEEE Trans. on Pattern Analysis and Machine Intelligence, May 1983, Vol PAMI-5 No.3, 246-267

[Vilain, 86] M. Vilain, & H. Kautz, *Constraint propagation algorithms for temporal reasoning*, AAAI-86,377-382