The Mechanisms of Temporal Inference*

B.R. Fox
McDonnell Douglas Research Laboratories
St. Louis, MO 63166

S.R. Green
McDonnell Douglas Astronautics Company
Kennedy Space Center, FL 32185

Abstract: The properties of a temporal language are determined by its constituent elements: the temporal objects which it can represent, the attributes of those objects, the relationships between them, the axioms which define the default relationships, and the rules which define the statements that can be formulated. The methods of inference which can be applied to a temporal language are derived in part from a small number of axioms which define the meaning of equality and order and how those relationships can be propagated. More complex inferences involve detailed analysis of the stated relationships. Perhaps the most challenging area of temporal inference is reasoning over disjunctive temporal constraints. Simple forms of disjunction do not sufficiently increase the expressive power of a language while unrestricted use of disjunction makes the analysis NP-hard. In many cases a set of disjunctive constraints can be converted to disjunctive normal form and familiar methods of inference can be applied to the conjunctive sub-expressions. This process itself is NP-hard but it is made more tractable by careful expansion of a tree-structured search space.

1. Introduction

An intelligent autonomous system operating in a remote, unstructured environment must have three capabilities. First, it must be able to create a plan or course of action according to an initial state of the world, a goal state of the world, and some knowledge of its own abilities. Second, it must be able to determine a sequence of actions, according to the constraints on the steps of the plan and the evolving state of the world. Finally, it must be able to produce the desired effect of those actions according to its abilities and the present state of the world.

The performance of the planner, the sequencer, and the executor components of such a system can be very much affected by the language used to represent plans. The concepts of action must be suitable for the planner, which must reason about goals and effects, but at the same time be tractable for the executor, which must produce the desired effects. The concepts of order must be sufficient for the planner, which must control undesired interactions between operations, but at the same time they must not impose unnecessary constraint on the sequencer, which must adapt the sequence of actions to the dynamically changing state of the world. The methods of formulation must enable the planner to produce the most general plans possible, yet at the same time it must be feasible for the sequencer to derive a sequence of actions from those plans. The language must be terse. The size of the plan must be proportional only to its complexity.

*Research conducted under the McDonnell Douglas Independent Research and Development Program.

It has been repeatedly suggested that reasoning over time is an essential element of planning and specific temporal representations have been proposed to facilitate the planning process, including a linear programming model of Malik and Binford[1], the space-time maps of Miller[2], the interval algebra of Allen[3], the point algebra of Vilain and Kautz[4], and the end-point representation of Cheeseman[5]. Although not concerned with planning but instead with the problems of sequencing the activities of robots, Fox and Kempf[6] propose a language of temporal constraints as the target representation for planners.

With this abundance of temporal languages for planning and sequencing, it is important to establish the properties of the proposed languages and to understand the inference methods which can be applied to them. Although a complete survey of temporal reasoning is beyond the scope of this paper, an examination of the basic elements of temporal representation and the methods of temporal inference will establish the primary criteria for comparing these languages.

2. Elements of Temporal Representation

The properties of a language are embodied in its syntactic form and its semantic interpretation. The concern here is with the semantic elements of a language rather than with its syntactic details. Nevertheless, in order to discuss the variety of possible temporal languages, it is necessary to introduce some simple syntactic structures which represent abstract semantic entities.

Temporal languages are concerned primarily with temporal objects: instants and intervals of time. Some authors maintain that instants of time present some semantic difficulties and therefore propose that time intervals should be the primitive element of temporal reasoning[3]. Others maintain that intervals of time can be defined by their endpoints and propose that instants should be treated as the basic element of temporal reasoning. Some sequencing problems involve activities, which in reality occur over some interval of time, but for purposes of analysis can be treated as atomic and indivisible. In the following discussion, instants of time will be treated as primitive objects, denoted by alphanumeric symbols such as X, Y, and nine-o'clock. Likewise, intervals will be denoted by alphanumeric symbols, such as Z, W, install-clip, and drill-hole, but when useful or necessary, the initial and final endpoints of an interval will be denoted by a suffix letter i or f attached to the interval name, such as Zi and Zf.

Temporal languages are concerned with the attributes of temporal objects. Some languages may allow the specification of the absolute time of some instant or it may be possible to specify the duration of an interval. Specialized systems may associate the properties of physical processes with intervals, such as rates, loads, or volumes. Planning systems may associate propositional variables and their values with temporal objects. Ultimately, each temporal object is associated with some event, activity, or proposition. For instance, it is possible to refer to the instant which begins an occultation, or the interval of time when the action install-clip is performed, or the interval of time over which the proposition channel-is-available is true.

Temporal languages are concerned with the relationships between temporal objects. The most primitive involve the relationships between instants of time. Two instants may be equal, denoted by the operator =, they may be inequal, denoted by the operator <>, or they may be ordered, as denoted by the operator <. In order to avoid any syntactic ambiguity, such relationships are written in fully parenthesized infix notation, as in the expression (X < Y). The relationships between two intervals of time, as defined by Allen, are shown schematically in Figure 1. The relationships between instants and intervals of time can be defined in a similar fashion. All of these relationships can be specified by their respective endpoint relationships as indicated in the right hand column of Figure 1.

In addition to the facilities for explicitly stating the relationships between temporal objects, a temporal language must include some axioms which define the relationships between objects that are not otherwise constrained. Commonly, it is assumed that, in the absence of other explicit constraints, two instants of time, X and Y, are ordered as either (X < Y) or (Y < X), or they are equal, (X = Y). Likewise, unless otherwise constrained, two intervals can be related in any of the 13 possible ways shown in Figure 1. In some applications involving the serial execution of a set of operations, there is no opportunity for any of the operations to be done concurrently. In such cases an axiom which defines the default relationship between intervals states that, unless otherwise constrained, two intervals, X and Y are disjoint and ordered as either (X before Y) or (Y before X). Such axioms play a significant role in the treatment of negation and the processes of inference.

Temporal languages are subject to certain rules of formulation. The simplest rule is to assume that a given set of primitive constraints is to be treated as a conjunction and that they must all be satisfied simultaneously. In contrast, the language defined by Allen allows a restricted form of disjunction. The relationship between a given pair of intervals can be specified as a disjunction of any of the 13 possible primitive relationships. This makes it possible to circumscribe indefinite relationships or to prescribe some relationships which cannot be expressed as one of the 13. For instance, suppose that two intervals, X and Y, must begin at the same time but that there is no constraint on their termination. It would artificially constrain the intervals to state that (X begins Y) because this primitive relationship requires that X terminate before Y. Likewise it would be an artificial constraint to require that (Y begins X). Using this vocabulary of 13 primitive relationships, the relationship between X and Y can only be stated as a disjunction, ((X begins Y) or (Y begins X)). In Allens's language, disjunction is restricted to phrases that define the relationship between a single pair of intervals and cannot be used to pose constraints such as, ((X before Y) or (Z before W)).

The properties of a temporal language are determined by the combination of these elements: the objects, attributes, relationships, default axioms, and rules of formulation. Together these elements determine the set of problems that can be represented. For instance, the language of strict partial orders is composed of symbols which denote instants of time, the primitive ordering relationship <, the default axiom that states for all X and Y either (X < Y) or (Y < X), and a rule of formulation that allows only conjunctions of primitive ordering constraints. A given constraint expression in this language defines a set of admissible total orderings over a set of instants of time. For example, The conjunction ((X < Z) and (Y < Z)) defines 2 admissible orderings of X, Y, and Z: [X,Y,Z] and [Y,X,Z]. The limited rule of formulation in the language of strict partial orders makes it impossible to state the constraints for a problem which admits the 4 linear orderings [X,Y,Z], [Y,X,Z], [Y,Z,X], and [Z,Y,X]. There is no conjunction of primitive ordering constraints which defines exactly this set of linear orderings! The limited forms of disjunction included in Allen's interval algebra or the point algebra defined by Vilain and Kautz encompass some sense of indefiniteness in the relationship between temporal objects but these forms of disjunction are not sufficient to represent the full range of possible ordering problems.

A number of common temporal representations can be quickly distinguished by their constituent elements. For instance, the language of equivalence classes is composed of symbols which denote atomic temporal objects, the equality and inequality relationships, = and <>, an axiom which states that for all X and Y, (X = Y) or (X <> Y), and a rule of formulation which allows only conjunctions of equality constraints. In contrast, the language of graph coloring problems has a similar structure but the rule of formulation allows only conjunctions of inequality constraints. The language of temporal constraints proposed by Fox and Kempf[6] is composed of symbols which denote atomic temporal objects, the ordering relation- ship <, an axiom of serial processes which states that for all X and Y, (X < Y) or (Y < X), and a rule of formulation which allows arbitrary use of conjunction, disjunction, and negation. This axiom limits the scope of this language to problems that involve activities that must be done one at a time, such as a robot performing an assembly task. However, the unrestricted use of disjunction guarantees that this language can represent any problem within that domain. Portrait, a temporal language under development by Fox and Green allows arbitrary use of equality, ordering, conjunction, disjunction, and negation.

3. Methods of Temporal Inference

Temporal reasoning is a process of deriving the properties of temporal objects and the relationships between temporal objects that are implied but may not be explicitly stated in a given set of temporal constraints. The most familiar form of temporal reasoning is constraint propogation. In the language of equivalence classes constraint propogation is based upon two axioms. The first defines the symmetry of equivalence: for all X and Y, (X = Y) implies that (Y = X). The second defines the method for propogating equivalence: for all X, Y, and Z, (X = Y) and (Y = Z) implies that (X = Z). There is no symmetry in the language of strict partial orders, only an axiom which defines the method for propagating order: for all X, Y, and Z, (X < Y) and (Y < Z) implies that (X < Z). The language of partially ordered sets includes an axiom which defines how a disjunction of order and equality can be propogated: for all X, Y, and Z, (X <= Y) and (Y <= Z) implies that (X <= Z). Coupled with this is an axiom which defines how constraints over a given pair of temporal objects can be resolved: for all X and Y, (X <= Y) and (Y <= X) implies that (X = Y). If, after the complete propogation of constraints, only one of the constraints (X <= Y) or (Y <= X) has been imposed then it can be assumed that the two objects are not

equal. Vilain and Kautz define a language over instants of time which, for a given pair of instants, allows an arbitrary disjunction of the 3 possible relationships between that pair. In this context, the propogation of constraints can be best defined by a matrix as shown in Figure 2. Conjunctions of constraints over a single pair of instants can be resolved by a rule of intersection as shown in the matrix of Figure 4. Vilain has demonstrated that constraint propogation within this language is both complete and correct. Allen's interval algebra relies upon similar, tabular rules of inference, but because of the added complexity of this language, constraint propagation is not guaranteed to be complete.

In most circumstances these constraint propagation axioms can be applied in reverse in order to identify the essential constraints in a problem and to eliminate any implied constraints. Given the complete set of implied and essential constraints it is a simple matter to identify the equivalence class of some temporal object along with all of its predecessors, direct predecessors, siblings, successors, and direct successors. For instance, the axiom which defines the predecessors of a temporal object Z states that X is a predecessor of Z if (X < Z). The direct predecessors of Z include all those temporal objects X such that (X < Z) but there does not exist Y such that (X < Y) and (Y < Z). These can be easily identified by scanning the set of essential constraints.

Reasoning about the admissible ordering of temporal objects is directly related to an analysis of precedessors and successors. For instance, in the language of strict partial orders, the controlling axiom specifies that a temporal object X can occur only after all of the predecessors of X. Of course, those objects which have no predecessors can occur at any time. This axiom can be used to incrementally build sequences of activities. At each step of the process simply choose one of those activities which can occur next.

In most sequencing problems the combined ordering constraints limit the admissible sequences of activities but do not remove every sequencing option. In most problems there are many admissible sequences. The number of admissible sequences can serve as a useful indicator of the available sequencing options. In some problems this may provide an estimate of the effort required to find the best sequence of activities. In other problems it may provide an estimate of the inherent flexibility that can be exploited in sequencing those activities. The naive approach to computing this number would be to explicitly enumerate all of the feasible sequences by exhaustive application of the sequencing axiom or other more sophisticated algorithms[7]. Unfortunately, the simplest of problems will prove the most intractable. Consider a serial task of 15 steps with no sequencing constraints. There exists 15! = 1,307,674,368,000 sequences. Even if one sequence could be generated each microsecond it would still requie 15 days to enumerate the entire set. Fortunately, general methods are available which can determine the number of feasible sequences over a strict partial order without explicit enumeration. These methods are first reported in a textbook by Wells[8] but several refinements of these methods were developed at MDRL by the authors. Generally, this computation can be accomplished by recursive application of 3 simple rules:

(1) if a set of activities can be divided into two subsets such that all of the activities in the first set must precede all of the activities in the second set, then the total number of feasible sequences equals the number of feasible sequences for performing the activities in the first set times the number of feasible sequences for performing the activities in the second set.

(2) if a set of activities can be divided into two subsets such that all of the activities in the first set can be performed independently of the acitvities in the second set, then the total number of feasible sequences equals the total number of feasible sequences for performing the activites in the first set times the number of feasible sequences for performing the activities in the second set times the number of ways that one sequence from the first set can be interleaved with one sequence from the second set.

3) if a set of activites cannot be divided into two subsets according to rules (1) or (2) then that set of activites can be partitioned into two strategies for performing those activies which have no feasible sequences in common, and the the total number of feasible sequences will be the number of feasible sequences under the first strategy plus the number of feasible sequences under the second strategy. The partition is generated by identify a pair of unconstrained activities, X and Y. The first strategy is defined by the orginal set of constraints plus the constraint that X must precede Y, (X < Y), and the second strategy adds the constraint that Y must precede X,

(Y < X). (Repeated application of these 3 rules is guaranteed to work regardless of the X and Y chosen when using rule 3, but the number of partitions generated is significantly affected by the choice. By carefully selecting the steps X and Y, it is possible to control the number of partitions ultimately generated.)

By recursive application of these rules it is possible to determine the number of feasible sequences for performing a set of activiites from start to finish, or it can be used to determine the number of ways of completing the task from any given state. In most circumstances the number of feasible sequences corresponds closely to the degree of flexibility inherent in the sequencing of the activites and it can be used as a valuable metric for comparing different plans or strategies. As a side-effect, application of the 3 rules stated above results in the decomposition of a given task into sets of dependent activities, sets of independent activities, and into disjoint sub-strategies. This decomposition can be used by human analysts to better understand the structure of the tasks that they must plan and coordinate.

Unfortunately, inference over a disjunctive language, such as that developed by Fox and Kempf, is much more difficult. One way of resolving the constraints in a disjunctive constraint expression is to convert a given set constraints into disjunctive normal form, i.e. a disjunction of conjunctions of the primitive ordering constraints, keeping only the satisfiable and non-redundant subexpressions. In that form, the methods of inference sketched above can be applied separately to each conjunction of constraints and the results combined under an appropriate interpretation of disjunction. The production of this reduced disjunctive normal form is very difficult, in fact it is NP-hard, but it is an essential part of more general temporal reasoning

For instance, the constraint expression shown in Figure 4 is typical of the constraints imposed on small assembly problems. Production of the disjunction normal form of that constraint expression, using the distributive law of boolean algebra, (X and (Y or Z)) --> ((X and Y) or (X and Z)), results in a set of 1024 conjunctions. In general, the size of the disjunctive normal form grows exponentially with the number of applications of the distributive law. Some of the resulting conjunctions are inconsistent and should never be considered, others are specific cases of more general sub-expressions in the result and can safely be removed. Other simple methods for producing the disjunctive normal form have the same result. However, all of the admissible sequences for performing the task defined by these constraints are embodied in only 22 conjunctions.

An efficient method for deriving that set of 22 conjunctions is closely related to methods for determining the satisfiabliity of boolean expression and is based on the expansion of a tree structured search space. Each node in the search space consists of 2 parts. The first is a partially formed conjunction, and the second is a constraint expression which remains to be satisfied. The root node consists of an empty conjunction coupled with the initial constraint expression. Successor nodes are formed by propagating primitive constraints from the constraint expression into the conjunction being constructed. The target leaf nodes consist of a completed conjunction which satisfies the original constraint expression and an empty set of constraints remaining to be satisfied. Specific heuristics have been developed which make it possible to prune redundant or inconsistent solutions early in the tree expansion. Using these methods the constraint expression shown in Figure 4 produced 28 consistent conjunctions, 6 of which were subsequently identifed as redundant. Subtree expansion was terminated 58 times because inconsistencies were detected and 12 times because redundancies were detected. This is considerably more efficient than producing 1024 conjunctions and then attempting to prune the inconconsistent and redundant sub-expressions.

4. Conclusion

The properties of a temporal language are determined by its constituent elements: the temporal objects which it can represent, the attributes of those objects, the relationships between those objects, the axioms which define the default relationships, and the rules which define the statements that can be formulated. The methods of inference which can be applied to a temporal language are derived in part from a small number of axioms which define the meaning of equality and order and how those relationships can be propagated. More complex inferences involve detailed analysis of the stated relationships. Perhaps the most challenging area of temporal inference is reasoning over disjunctive temporal constraints. Simple forms of disjunction do not sufficiently increase the expressive power of a language while unrestricted use of disjunction makes the analysis NP-hard. In many cases a set of disjunctive constraints can be converted to disjunctive normal form and familiar methods of inference can be applied to the conjunctive sub-expressions. This process itself is NP-hard but it is made more tractable by careful expansion of a tree-structured search space.

References

[1]Malik, J. and Binford, T.O., Reasoning in time and space, Proceedings Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, 1983.

[2]Miller, D., Scheduling heuristics for problem solvers, Research Report 264, Yale University Computer Science Department, New Haven, Conn., 1983.

[3]Allen, J.F., and Koomen, J.A., Planning using a temporal world model, Proceedings Eighth International Joint Conference on Artificial Intelligence, Karlsruhe, West Germany, 1983.

[4]Vilain, M. and Kautz, H., Constraint propogation algorithms for temporal reasoning, Proceedings Fifth National Conference on Artificial Intelligence, Philadelphia, Penn., 1986.

[5]Cheeseman, P., A representation of time for automatic planning, Proceedings Second IEEE international Conference on Robotics and Automation, Atlanta, Georgia, 1983.

[6]Fox, B.R. and Kempf, K.G., A representation for opportunistic scheduling, Proceedings Third International Symposium on Robotics Research, Paris, France, 1985.

[7]Kalvin, A.D. and Varol, Y.L., On the generation of all topological sortings, Journal of Algorithms, v4, pp. 150-162, 1983.

[8]Wells, M.G., Elements of Combinatorial Computing, Pergamon Press, Elmsford, New York, 1971.

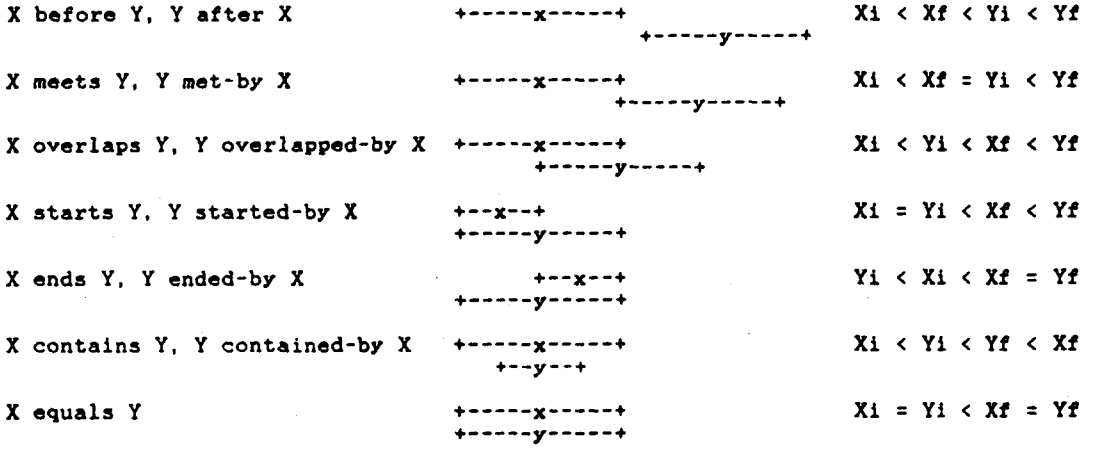

Figure 1.
Thirteen possible interval relationships.

yRz / xRy \ xRz	<	=	>	<=	>=	<>	<=>
<	<	<	<=>	<	<=>	<=>	<=>
=	<	=	>	<=	>=	<>	<=>
>	<=>	>	>	<=>	>	<=>	<=>
<=	<	<=	<=>	<=	<=>	<=>	<=>
>=	<=>	>=	>	<=>	>=	<=>	<=>
<>	<=>	<>	<=>	<=>	<=>	<=>	<=>
<=>	<=>	<=>	<=>	<=>	<=>	<=>	<=>

Figure 2.
Matrix of constraint propagation in the point algebra.

xRy \ xRy \ yRz	<	=	>	<=	>=	<>	<=>
<	<	x	x	<	x	<	<
=	x	=	x	=	=	x	=
>	x	x	>	x	>	>	>
<=	<	=	x	<=	=	<	<=
>=	x	=	>	=	>=	>=	>=
<>	<	x	>	<	>	<>	<>
<=>	<	=	>	<=	>=	<>	<=>

Figure 3.
Matrix of constraint resolution in the point algebra.

```
 (co before cl)                                  and
 (ba before cl)                                  and
((co before st) or (co before dr))               and
((dr before co) or (dr before ba))               and
((ba before dr) or (ba before ca))               and
((ra before co) or (ra before ba))               and
((mi before ra) or (mi before ms))               and
((mi before co) or (mi before ba))               and
((sm before mi) or (sm before ba))               and
((sm before co) or (sm before ba))               and
((ri before co) or (ri before ba))
```

Figure 4.
Typical disjunctive constraints on the steps of an assembly problem.