Proceedings of the 1986 Winter Simulation Conference
J. Wilson, J. Henriksen, S. Roberts (eds.)

Temporal Reasoning

David P. Miller

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

1 Introduction

Temporal reasoning has been a major area of Artificial Intelligence research for a number of years. Reasoning about time is necessary for creating realistic plans and simulations for almost all interesting problems.

Two "types" of temporal relations must be dealt with when modeling a problem. They are:

Absolute: those temporal relationships that are referenced off some fixed coordinate system such as a calendar. The distance between two events that have absolute times associated with them can be trivially calculated without any other knowledge about the events in the world.

Relative: temporal relationships that are referenced off other events going on in the world. The temporal distance between events cannot necessarily be calculated exactly without complete knowledge about other events going on in the world.

The relationships between two events in time can also be absolute or relative. Some relationships specify an exact amount of time that must come between the two events: ten minutes after the plastic has been poured, remove the widgets from their molds. More common are the simple relative relationships which just specify that one task must follow another. In between are tasks that are related by some interval: to ensure unwrinkled clothes remove them within an hour of the dryer finishing.

There have been several attempts to formalize temporal relationships so that inferences from time information could be made. There have also been several programs created which make minimal use of a temporal representation, but rather perform incremental temporal simulations. Both of these types of systems are surveyed in the following pages.

2 Temporal Representations

There have been three basic approaches towards achieving a useful temporal representations scheme. One of these attempts has its origins in the *situation calculus* [5], [6] and has been expanded upon by McDermott (see [7]). This system looks upon time as an infinite number of distinct *states*. A state is an instantaneous description of part or all of the universe. *Actions* and *events* are modeled in this system as operators for getting from one state to another.

Hayes created a logic where the entire set of effects of an action could be represented. The main component of his system was the *history*, a four dimensional space-time object that represented all the relevant parts of an event as it progressed through time. The ways in which different histories interact with one another, and the time over which a given history can propagate, are all determined by a set of axioms given in the first-order logic. Some of the axioms necessary for representing the domain of liquids is presented in [4].

A temporal logic by Allen [1] assumes that intervals are the basic unit of representing temporal information. Allen's system has thirteen relations that can hold between intervals. Rules on how these relationships can be combined and their transitive properties are also given. By assigning intervals to each event and action, and then enumerating all the relationships that are known to exist between various intervals, Allen's system allows additional relationships to be inferred.

The major drawback of Allen's system is that it can do only limited inferences involving metric information. In order to be able to propagate all the metric information that can be associated with an event intervals must be broken into their endpoints and operations done on those points. This combines the interval system with state-based information. Temporal data base systems such as [2] and [9] do just that.

Dean's style of temporal data base, called a *time-map*, is especially useful for storing temporal information for certain planning systems. The time-map is an elegant extension of the effects and preconditions network used in the Nonlin planner [10]. The time-map contains links, between the elements in the data base, that capture all of the ordering information contained in the partial-order of those elements. In this way the time-map can capture all of the information normally contained in the task network associated with least-commitment planners such as Nonlin. Additionally, the time-map can store the amount of time necessary to accomplish each action in the network. By *projecting* the effects associated with each action in the network, the time-map can be used to calculate the *persistence* of any fact in the data-base. For instance, the fact that a loaded gun is loaded will persist until the gun is unloaded or the gun is fired. Either of those two actions is said to have *clipped* the persistence of the gun being loaded. A time-map can be used to detect some adverse plan interactions by noticing places in the data base where the persistence of one plan's prerequisite is clipped by some other plan.

A time-map can be used to detect all of the potential conflicts that could be detected by Nonlin. Additionally, it can detect some conflicts involving metric time constraints. For example, if two tasks must be accomplished within three hours, each requires two hours to be executed, and they must be done sequentially, then the time-map can detect that a deadline violation will occur. However, a partially-ordered network like the time-map is still inadequate to detect conflicts that arise from the *state-transition* delays and effects between tasks. Only a complete ordering can reveal the results of the state-transitions and their effects on the rest of the plan.

3 Temporal Simulations

State-transition delays are those delays in executing a task that are due to changing from the state of the world of the previous task, to that state which is necessary for executing the current task. For example, if the next task to be accomplished is to turn on a factory lathe, and the last task left the worker in a storeroom, then a state-transition-delay of however long it takes the worker to get from the storeroom to the lathe must be executed before the lathe may be turned on.

While it is possible to represent a state-transition delay, with the representations discussed previously, after the fact, they cannot be predicted in advance without first creating a total ordering of all the tasks to be accomplished. Yet a consistent total ordering cannot be simply pulled out of thin air.

Several AI planning systems have been written that perform some sort of search through the space of total orderings, so as to locate a consistent and efficient total ordering. The Deviser system [11] is an extension to the Nonlin planner. The extensions allow the system to recognize when it has found a temporally consistent plan. The program generates plans, using the Nonlin control structure, until a satisfactory plan is encountered.

Isis [3] is a program that does factory job-shop scheduling. Isis uses heuristics about the factory domain to make task-ordering decisions. These heuristics guide the generation of alterenative schedules so that a satisfactory schedule is quickly located. The system has achieved performance levels superior, in schedule efficiency, to that of its human counterparts.

Bumpers [8] uses general heuristics about scheduling to help order tasks. The system also has an interface that allows simple simulations of arbitrary physics to be added to the system. This allows Bumpers to create efficient total-orderings for a wide variety of application domains.

4 Summary

Temporal reasoning is a necessary part of planning for and simulating interesting problems. A number of temporal representations have been created. These representations allow inferences to be made about the temporal effects of executing several tasks while those tasks have yet to be ordered. Some valuable inferences cannot be accurately drawn until the tasks are totally ordered. A variety heuristic search programs have been created for efficiently exploring the effects of different total orderings. These systems draw temporal inferences through the simulation and projection of task orderings.

References

[1] James Allen.
Maintaining knowledge about temporal intervals.
Comm. ACM, 26(11):832–843, 1983.

[2] Thomas Dean.
Time Map Maintenance.
Technical Report 289, Yale University Computer Science Department, 1983.

[3] Mark S. Fox.
Constraint-Directed Search: A Case Study of Job-Shop Scheduling.
Technical Report CMU-RI-TR-83-22, CMU Robotics Institute, December 1983.

[4] Patrick Hayes.
Liquids.
In J. Hobbs, editor, *Formal Theories of the Commonsense World*, Lawrence Erlbaum Associates, 1984.

[5] John McCarthy.
Programs with common sense.
In *Proceedings of the Symposium on the Mechanization of Thought Processes*, National Physiology Laboratory, 1958.

[6] John McCarthy and Patrick J. Hayes.
Some philosophical problems from the standpoint of artificial intelligence.
In *Machine Intelligence 4*, Edinburgh University Press, 1969.

[7] Drew V. McDermott.
A temporal logic for reasoning about processes and plans.
Cognitive Science, 6:101–155, 1982.

[8] David Miller.
Planning by Search Through Simulations.
Technical Report 423, Yale University Computer Science Department, October 1985.
PhD thesis.

[9] Stephen F. Smith.
Exploiting Temporal Knowledge to Organize Constraints.
Technical Report CMU-RI-TR-83-12, CMU Robotics Institute, 1983.

[10] Austin Tate.
Project Planning Using a Hierarchic Non-Linear Planner.
Research Report 25, Dept. of Artificial Intelligence, University of Edinburgh, 1976.

[11] Steven Vere.
Planning in time: windows and durations for activities and goals.
IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI-5(3):246–267, 1983.

Author's Biography

DAVID P. MILLER is an assistant professor of computer science at Virginia Polytechnic Institute and State University. He received a B.A. in Astronomy at Wesleyan University in 1981, and a Ph.D. in Computer Science from Yale University in 1985. His current research interests include the following areas of Artificial Intelligence: Planning, Knowledge Representation, Scheduling, Robotics. He is a member of ACM, IEEE, and AAAI.

David P. Miller
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061, U.S.A.
(703) 961-5605
miller@vpi.csnet