An Approach to Reasoning About Continuous Change for Applications in Planning

Thomas Dean*   Greg Siegle

Department of Computer Science

Brown University

Box 1910, Providence, RI 02912

Abstract

There are many planning applications that require an agent to coordinate its activities with processes that change continuously over time. Several proposals have been made for combining a temporal logic of time with the differential and integral calculus to provide a hybrid calculus suitable for planning applications. We take one proposal and explore some of the issues involved in implementing a practical system that derives conclusions consistent with such a hybrid calculus. Models for real-valued parameters are specified as systems of ordinary differential equations, and constructs are provided for reasoning about how these models change over time. For planning problems that require projecting the consequences of a set of events from a set of initial conditions and causal rules, a combination of numerical approximation and symbolic math routines and a simple default reasoning strategy provide for an efficient inference engine.

Introduction

Many problems in planning, scheduling, and decision support require reasoning about processes that change continuously over time (*e.g.*, determining how long to leave a valve open in order to fill a container without causing it to overflow, or when to schedule the delivery of parts after the start of a machining process so as to minimize the total time spent in fabrication). While there has been some research involving continuous change (*e.g.*, [Hendrix, 1973]), much of the work on temporal reasoning in artificial intelligence has focused on discrete change [Allen, 1984, McDermott, 1982, Shoham, 1988]. Recently, however, researchers [Sandewall, 1989, Rayner, 1989] have noted that the differential and integral calculus provide us

with a perfectly good means of reasoning about continuous change. Sandewall [1989] describes a hybrid calculus that combines an interval temporal logic with the differential calculus. In this paper, we discuss some of the issues involved in implementing a variant of Sandewall's hybrid calculus useful for applications in planning.

Discrete and Continuous Change

Following [McDermott, 1982] and [Shoham, 1988], we treat time points as primitive and reason about intervals in terms of points. Time points are notated t or ti, $i \in \mathbf{Z}$ (*e.g.*, t1, t2). Variables ranging over time points are notated t or t_i, $i \in \mathbf{Z}$ (*e.g.*, t_1, t_2). We introduce a binary relation, $\preceq$, on time points indicating temporal precedence. If t1 and t2 are time points, then $\langle$t1,t2$\rangle$ is an interval. We use the notation holds(t1,t2,p) to indicate that the proposition p is true throughout the interval $\langle$t1,t2$\rangle$. For instance,

$$\text{holds(t1,t2,temp(room32)}>72°)$$

is meant to represent the fact that the temperature in a particular room is greater than 72° throughout the interval $\langle$t1,t2$\rangle$. We use the abbreviation holds(t,φ) for holds(t,t,φ), and $\Box\varphi$ to indicate that φ is always true.

In order to reason about discrete change, the logic has to be extended to deal with the problems that arise due to the frame and qualification problems. As an expedient, we adopt Shoham's semantics of chronological minimization [Shoham, 1988], noting that, while not appropriate for all types of temporal reasoning, chronological minimization is entirely satisfactory for the simple sort of projection problems that arise in many planning applications.

In this paper, we are primarily interested in reasoning about quantities that change continuously as functions of time. Rather than invent new machinery within our temporal logic, we will import into the logic as much of the differential calculus as is needed for our planning applications. Our treatment here roughly follows that of Sandewall [1989].

First, we introduce a set, U, of real-valued parameters closed under the differential operator, ∂. If $u \in U$, then $\partial^n u \in U$, where $\partial^n u$ is the nth partial derivative of u with respect to time. We can trivially extend the syntax to represent statements about the values of parameters at various time points. For instance,

holds(t1,t2,y = 3.1472)

is meant to indicate that the parameter y has the value of 3.1472 throughout the interval $\langle$t1,t2$\rangle$. By restricting y to remain constant throughout the interval $\langle$t1,t2$\rangle$, we also restrict ∂y to remain 0 throughout the same interval.

To guarantee this intended meaning, we have to augment the semantics somewhat. In addition to a set of parameters U, we assume that each interpretation includes a function $Q : (\mathbf{R} \times U) \rightarrow \mathbf{R}$, where we employ the set of real numbers, $\mathbf{R}$, for the set of time points as well as for the set of all parameter values.

Since we will find it convenient on occasion to model abrupt changes in the value of parameters as they change over time, we introduce the notion of a *breakpoint*. We assume that a physical process is modeled using a set of differential equations that describe continuous changes in the parameters over intervals of time, and a set of axioms that determine what equations are appropriate over what intervals. Breakpoints are times at which the axioms signal a change in the differential equations used to model a given quantity or set of quantities. Generally, at a breakpoint there is a discontinuity in some time varying parameter.

We have to augment the semantics to account for the behavior of parameters with respect to breakpoints. Each interpretation must include a set of breakpoints $S \subset \mathbf{R}$, so that for all $u \in U$, $Q(t, u)$ is continuous over every interval not containing an element of S, and for all $t \notin S$, $\frac{du}{dt} = Q(t, \partial u)$. Strange things can happen at breakpoints, but not so strange that we will allow a parameter to take on two different values. To avoid such anomalies, we will have to introduce some additional machinery.

At time t_0, we have a set of differential equations and a set of initial values for all of the parameters; these equations and initial values are known to hold until some indeterminate time t_1, at which point a breakpoint occurs and the axioms determine a new set of differential equations and a new set of "initial" values. In order to establish breakpoints and the values for parameters immediately following breakpoints, we need to refer to the values of parameters "just before" and "just after" breakpoints. To do so, we define the left and right limits of a parameter x at time t as:

$$\lim_{\tau \to t^-} Q(\tau, x) = Q(t, x^l) \quad \text{and} \quad \lim_{\tau \to t^+} Q(\tau, x) = Q(t, x^r)$$

A *discontinuity* occurs at t with regard to a parameter x whenever the left and right limits are not equal:

$$Q(t, x^l) \neq Q(t, x^r)$$

As long as there are no discontinuities, the differential equations tell us exactly how the parameters vary with time. The axioms tell us when breakpoints occur and what differential equations and initial conditions should be used to model processes between breakpoints. Discontinuities play a role in reasoning about real-valued quantities analogous to the role played by clippings in reasoning about the persistence of propositions. Just as the axioms do not rule out spurious models resulting from unexplained clippings, neither do they rule out models resulting from unexplained discontinuities.

Suppose that we have two objects moving toward one another along a horizontal line. Assume that the surface is frictionless, the objects are represented as identical point masses, and there are no external forces acting on the objects. Let x_1 and x_2 represent the parameters corresponding to the position of the first and second objects, respectively, as measured from some reference on the horizontal line. At time 0, the first object is located at position 0, and the second object is located 10 meters to the right. A positive velocity indicates movement to the right. We make use of the standard conventions for notating position (x), velocity $(\partial x = \dot{x})$, and acceleration $(\partial^2 x = \ddot{x})$. Here are the axioms indicating the initial conditions:

holds(0,x_1 = 0)   holds(0,x_2 = 10)

holds(0,$\dot{x}_1$ = 2)   holds(0,$\dot{x}_2$ = −3)

holds(0,$\ddot{x}_1$ = 0)   holds(0,$\ddot{x}_2$ = 0)

where velocity is in units of meters per second. The next axiom determines the new velocities immediately following a collision breakpoint.

$$\Box((x_1 = x_2) \wedge ((\dot{x}_1 - \dot{x}_2) > 0)) \supset ((\dot{x}_1^l = \dot{x}_2^r) \wedge (\dot{x}_2^l = \dot{x}_1^r))$$

For the most part, the propositions corresponding to equations involving the parameters in U are constantly changing. In order for us to make useful predictions, however, certain equations have to persist over intervals of time. Suppose you are told that at time t_0, $x = 0$, $\dot{x} = 2$, and $\ddot{x} = 0$. If $x = 0$ persists, then there will be discontinuities in $\dot{x}$ and $\ddot{x}$. If $\ddot{x} = 0$ persists, then $\dot{x} = 2$ has to persist or be discontinuous in order to avoid a discontinuity in $\dot{x}$, and x is completely determined by $\dot{x} = 2$. However, if none of $x = 0$, $\dot{x} = 2$, or $\ddot{x} = 0$ persist, there need not be a discontinuity in any one of x, $\dot{x}$, or $\ddot{x}$, but neither is there any way of predicting the changes in x over time. In this example, we force an interpretation by stating that the accelerations for the two objects are always 0: $\Box((\ddot{x}_1 = 0) \wedge (\ddot{x}_2 = 0))$.

Using Sandewall's extension of Shoham's chronological minimization, there is a single discontinuity in the acceleration of the objects two seconds after time 0, after which the objects, having exchanged velocities, head in opposite directions forever. We assume that the values of parameters are established in intervals not containing breakpoints by differential equations. In the

following, we distinguish propositions corresponding to real-valued parameters taking on specific values (*e.g.*, $\ddot{x} = 2$) from propositions corresponding to truth-valued parameters (*e.g.*, on(furnace17)).

In the previous example, $\Box((\ddot{x}_1 = 0) \wedge (\ddot{x}_2 = 0))$ serves as the model for x_1 and x_2. In other cases, it may be convenient to infer a change in a model that persists over some indeterminate interval of time, just as we are able to infer changes in propositions that persist over intervals of time. To handle this sort of inference, we introduce a particular type of proposition models(x, m) where x is a real-valued parameter and m is a model for x. If m is an nth-order differential equation, then it is assumed that the nth-order equation determines all higher-order derivatives, and all lower-order derivatives are known as part of the initial conditions. By stipulating $\Box(\ddot{x} = 0)$, we implicitly indicated holds$(0, $models$(x, \ddot{x} = 0))$ and that $x = 0$ and $\dot{x} = 2$ were the initial conditions at 0. Propositions of the form models(x, m) persist according to standard chronological minimization.

Suppose that we want to reason about the temperature in a room heated by a furnace, and suppose that the furnace is controlled by a thermostat set to 70°. To make the example more interesting, suppose further that the thermostat has a 4° differential (*i.e.*, the furnace starts heating when the temperature drops to 68° and stops when the temperature climbs to 72°). To represent parameters "dropping to" ($\downarrow$) or "climbing to" ($\uparrow$) certain values, we define trans$(\downarrow, u, v)$ (similarly for $\uparrow$) where $u \in U$ and $v \in \mathbf{R}$ as follows:

holds$(t, $trans$(\downarrow, u, v)) \equiv$
$$Q(t, u) = v \; \wedge \; \exists t' \prec t, \forall t' \prec t'' \prec t, Q(t'', u) > Q(t, u)$$

Propositions of the form trans$([\downarrow \mid \uparrow], u, v)$ are used to represent point events of the sort that trigger changes.

To model changes in the room's temperature when the furnace is off, we use Newton's law of cooling

$$\frac{dr}{dt} = -\kappa_1(r - a)$$

where r is the temperature of the room, a is the temperature outside the room, and κ_1 depends on the insulation surrounding the room. To model changes in the room's temperature when the furnace is running, we use
$$\frac{dr}{dt} = \kappa_2(f - r) - \kappa_1(r - a)$$

where f is the temperature of the furnace when it is running, and κ_2 depends on the heat flow characteristics of the furnace. The following axioms describe the temperature in the room over time.

$\Box($trans$(\uparrow, r, 72°) \wedge $on(furnace17)$) \supset$
$\qquad$models$(r, \partial r^r = -\kappa_1(r - a))$

$\Box($trans$(\downarrow, r, 68°) \wedge $on(furnace17)$) \supset$
$\qquad$models$(r, \partial r^r = \kappa_2(f - r) - \kappa_1(r - a))$

Suppose that we are interested in the temperature in the room over the interval from time 0 to time 10. We are told that the temperature outside is 32° throughout this interval, and that at time 0 the room is 75° with the furnace on but currently not heating. We represent these facts as follows:

holds$(0, r = 75°)$ $\qquad$ holds$(0, \partial r = -\kappa_2(r - a))$

holds$(0, 10, a = 32°)$ $\quad$ holds$(0, $on(furnace17)$)$

With a little extra work (*e.g.*, in order to eliminate certain unintended models, we have to take steps to avoid simultaneous cause and effect), we can obtain the following inferences. The temperature drops off exponentially[1] from 75° to 68° at which point the furnace starts heating and continues until the temperature reaches 72°, after which the furnace toggles on and off forever with the temperature always between 68° and 72°.

Note that we can always substitute a set of models that persist over different intervals of time for a single model that is true for all time but with additional parameters that make the model behave differently over different intervals of time. In the furnace example, we might state that $\Box(\partial r = \kappa_2(f - a) - \kappa_1(r - a))$ and then have rules that govern the value of f over different intervals of time. The choice of whether to vary the model or employ a single model and vary the parameters of the model is a matter of preference. The system described in the next section supports either approach.

Projection Involving Continuous Change

In this section, we discuss the issues involved in building a temporal inference engine for reasoning about continuously changing quantities. We consider only a limited form of temporal reasoning called *projection* that can be performed by making a single sweep forward in time inferring at each point what things change and what things remain the same. Following [Dean and McDermott, 1987], we distinguish between a general type of event or proposition (*e.g.*, "the furnace came on") and a specific instance of a general type (*e.g.*, "the furnace came on at noon"). The latter are referred to as *time tokens* or simply *tokens*. A token associates a general type of event or proposition with a specific interval of time over which the event is said to occur or the proposition hold. Tokens are notated token(t, i) where t is a type and i is an interval; begin(i) and end(i) indicate the begin and end points respectively of the interval i. Projection uses a set of initial tokens and causal rules

_____

[1] The behavior of the system can be described in terms of a piecewise continuous function in which the specific solutions for each piece are given, alternately, by $r(t) = 32° + (r_0 - 32°)e^{-\kappa t}$ and $r(t) = C + (r_0 - C)e^{-(\kappa_2 + \kappa_1)t}$ where $C = \frac{\kappa_2 500° + \kappa_1 32°}{\kappa_1 + \kappa_2}$, r_0 is the initial temperature of the room for that particular piece and t is the time elapsed from the beginning of that piece.

corresponding to events and propositions to generate additional tokens corresponding to the consequences of the events. Metric constraints are handled as in [Dean and McDermott, 1987]

We require that the interval corresponding to a token persist no further than the first subsequent interval corresponding to a token of a *contradictory type*. For any proposition type φ, φ and $\neg\varphi$ are said to be contradictory. Additional contradictory types have to be explicitly asserted. For instance, the assertion

```
contradicts(location(X,Y),location(X,Z)) ← Y≠Z.
```

indicates that any two tokens of type location(*arg1*, *arg2*) are contradictory if their first arguments are the same, and their second arguments are different. The process of modifying the bounds on token intervals corresponding to propositions to ensure that tokens of contradictory types do not overlap is referred to as *persistence clipping*.

Causal rules for reasoning about discrete change are of the form project(*antecedent_conditions*, *trigger_event*, *delay*, *consequent_effects*) to indicate that, if an event of type *trigger event* occurs, and the *antecedent conditions* hold at the outset of the interval associated with *trigger event*, then the *consequent effects* are true after an interval of time determined by *delay*. The trigger event is specified as a type, the antecedent conditions and consequent effects are specified as types or conjunctions of types, and the delay is optional defaulting to ϵ, a positive infinitesimal. As an example project(¬on(furnace17), toggle(switch42), on(furnace17)) indicates that, if the switch on the furnace is toggled at a time when the furnace is not on, then, after a delay of ϵ, it will come on. The basic algorithm for persistence clipping and projecting the consequences of events is described in [Dean and McDermott, 1987]; in the following, we extend that algorithm to handle continuous change.

Let U be a set of real-valued parameters, and P be a set of boolean-valued propositional variables.[2] In addition, we introduce two mappings $Q : \mathbf{R} \times U \rightarrow 2^{\mathbf{R}}$ and $V : \mathbf{R} \times P \rightarrow 2^{\{true, false\}}$. The task of projection is to determine Q and V for some closed interval of $\mathbf{R}$. We begin by considering the completely determined case in which both Q and V map to singleton sets (*i.e.*, $Q : \mathbf{R} \times U \rightarrow \mathbf{R}$ and $V : \mathbf{R} \times P \rightarrow \{true, false\}$).

At the initial time point, we assume that the values of all parameters and propositional variables are known. In addition, we are given a set of events specified to occur at various times over the time interval of interest. We assume a set of projection rules as before. In addition, we assume a set of modeling rules for parameters in U. A modeling rule is just a special sort of projection rule; the basic form is the same as that introduced

---

[2] It should be noted that, despite the presence of variables and complex terms in our rules, the underlying logic is purely propositional.

earlier in this section, the only difference being that the delay is always assumed to be ϵ, and the consequent effects consist of parameter assignments in the form of ordinary differential equations with constant coefficients (*e.g.*, $\partial u = 2$, or $\partial^2 u = 3\partial u + 5u + 4$).

The projection rule from the last section for reasoning about the temperature of the room in the case that the furnace is on but not running is encoded as follows,

```
project(on(furnace17),trans(↑,r,72°),
        models(r,∂r^r = −κ₁(r − a))).
```

To make sure that persistence clipping is handled correctly, we state that a given parameter can have only one model at a time.

```
contradicts(models(X,M1),models(X,M2)) ← M1≠M2.
```

Now we can state the basic algorithm for performing projection given some set of initial conditions and a projection interval $\langle t_s, t_f \rangle$. To simplify the description of the algorithm, we assume that all events are point events (*i.e.*, if e is a type corresponding to the occurrence of an event, token(e, k) $\supset$ (begin(k) = end(k))), and all events described in the initial conditions begin after t_s. Let $\mathcal{A}$ be the set of all currently active process models (*i.e.*, all m such that holds(t_c, models(x, m)) for some x). Let $\mathcal{E}$ be the set of pending events (*i.e.*, the set of all events, token(e, k), generated so far such that $t_c \prec$ begin(k)). Let $\mathcal{C}$ be the set of current conditions (*i.e.*, all $u^r = v$ such that there exists $m \in \mathcal{A}$ such that holds(t_c, models(x, m)), $u = \partial^n x$ for some n, and holds($t_c, u^r = v$)).

In the cases that we are interested in, we can recast a set of ordinary differential equations and their initial conditions as a system of first-order differential equations. We can then solve these equations using numerical methods based on the Taylor expansion (*e.g.*, the Runge-Kutta methods [Ralston and Rabinowitz, 1978]) and various forms of linear and nonlinear extrapolation (*e.g.*, the Adams-Bashforth and Adams-Moulton methods [Shampine and Gordon, 1975]). In the following, we assume the ability to generate solutions to ordinary differential equations efficiently, and refer to the procedure for generating such solutions as the *extrapolation procedure*. Given a set of initial conditions and a projection interval $\langle t_s, t_f \rangle$ projection is carried out by the following algorithm.

1. Set t_c to be t_s.

2. Set $\mathcal{E}$ to be the set of events specified in the initial conditions.

3. Using $\mathcal{A}$, $\mathcal{C}$, and the extrapolation procedure, find t_n corresponding to the earliest point in time following t_c such that the trigger for some projection rule is satisfied or t_f whichever comes first. If $t_n \neq t_f$, then t_n could be the time of occurrence of the earliest event in $\mathcal{E}$, or it could be earlier, corresponding to the solution of a set of simultaneous equations (*e.g.*, $((x_1 = x_2) \wedge ((\dot{x}_1 - \dot{x}_2) > 0)))$.

4. If $t_n = t_f$, then quit, else set t_c to be t_n.

5. Find all of the projection rules with the trigger found in Step 3.

6. For each rule found in Step 5 whose antecedent conditions are satisfied, create tokens corresponding to the types of the consequent effects, except in the case of consequent effects corresponding to parameter assignments (*e.g.*, $x_1^r = x_2^r$). Constrain the new tokens according to the delay specified in the corresponding rule, and add them to the database.

7. For each token added in Step 6 whose type corresponds to an event, add it to $\mathcal{E}$.

8. For each token added in Step 6 whose type does not correspond to an event, find all tokens of a contradictory type that begin before the newly added token and constrain them to end before the beginning of the new token.

9. If the trigger found in Step 3 corresponds to the type of an event token in $\mathcal{E}$ whose time of occurrence is t_c, remove it from $\mathcal{E}$.

10. Use the consequent effects corresponding to parameter assignments found in Step 6 and the results of extrapolation to determine $\mathcal{C}$. The parameter assignments corresponding to the consequent effects of projection rules take precedence over the extrapolation results. $\mathcal{A}$ is also updated at this time.

11. Go to Step 3.

The above algorithm has been implemented in Prolog and C. We use C-Prolog as a front end and database for storing projection rules. The extrapolation procedure employs Runge-Kutta methods and is written in C. Differential equations are specified using the notational conventions of Maple.[3] Prolog routines are used to preprocess the differential equations converting each one into a system of first-order equations. We assume that all equations are 5th order or less, and that they can be rewritten so that the highest-order term is algebraically isolated on the left-hand side of the equation. The system can make use of analytic solutions when available, but, for the planning problems we are concerned with, the extrapolation routine is more than accurate enough. It is also generally faster to use the extrapolation routine written in C than the analytic solver written in Prolog.

To get a better idea of how the program works, consider the following simple benchmark problem. Figure 1 depicts a pipe leading into a holding tank used to fill portable tanks that are positioned beneath a second pipe leading out of the holding tank. There are rotary valves mounted on the pipes that restrict the flow of

---

[3]Maple is a widely distributed symbolic math package developed by the Symbolic Computation Group in the Department of Computer Science at the University of Waterloo, Waterloo, Ontario.
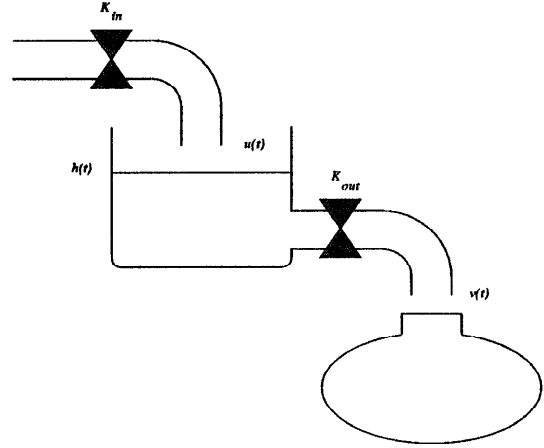

Figure 1: Reasoning about fluid flow

```
% Constants:                % Initial conditions:        % Discrete events:
constant(area,5).           holds(0,u=0).                occurs(1,turn(θ_in,15)).
constant(height,3).         holds(0,v=0).                occurs(2,turn(θ_out,90)).
constant(volume,2).         holds(0,h=0).                occurs(5,turn(θ_in,45)).
constant(k_in,5).           holds(0,θ_in=0).             occurs(10,turn(θ_in,-60)).
constant(k_out,3).          holds(0,θ_out=0).            occurs(11,turn(θ_out,-90)).

% Static models:
holds(T,models(r,r(t)=k_in*θ_in(t)/(k_out*θ_out(t)))).
holds(T,models(u,diff(u(t),t) = k_in*θ_in(t))).
holds(T,models(v,diff(v(t),t) = k_out*θ_out(t)*h(t))).
holds(T,models(sp,sp(t)=sp1(t)+sp2(t))).
holds(T,models(θ_in,diff(θ_in(t),t)=0)).
holds(T,models(θ_out,diff(θ_out(t),t)=0)).
holds(T,models(P,_=C)) :- constant(P,C).

% Dynamic models:
holds(T,models(h,diff(h(t),t)=(k_in*θ_in(t)-k_out*θ_out(t)*h(t))/area)) :-
    holds(T,h < height).
project(always,trans(up,h,height),models(h,diff(h(t),t)=0)).
project(always,trans(down,r,height),
    models(h,diff(h(t),t)=(k_in*θ_in(t)-k_out*θ_out(t)*h(t))/area)).
holds(T,models(sp1,sp1(t)=0)) :- holds(T,h < height).
project(always,trans(up,h,height),models(sp1,sp1(t)=u(t))).
project(always,trans(down,r,height),models(sp1,sp1(t)=0)).
holds(T,models(sp2,sp2(t)=0)) :- holds(T,v < volume).
project(always,trans(up,v,volume),models(sp2,sp2(t)=v(t))).
```

Figure 2: Prolog clauses for the fluid-flow problem

fluid; the valves vary from 0° to 90°. We are interested in the consequences of a plan involving a sequence of adjustments to the two valves. In particular, we are interested in the volume of the fluid in the portable tank after the output valve is finally closed, and the total amount of fluid spilled from either the holding tank or the portable tank during the filling process.

Let K_{in} be the flow rate of the input valve in cubic meters per degree minute, K_{out} be the flow rate of the output valve in square meters per degree minute, H be the height of the holding tank in meters, A be the surface area of the portable tank, and V be its total volume. In addition to these constants, we have the following state variables (functions of time): h is the height of the fluid in the holding tank, θ_{in} is the angle of the input valve, θ_{out} is the angle of the output valve, u is the total volume of fluid to have entered the holding

tank, and v is the total volume of fluid to have left the holding tank. Initially, we have $h(0) = u(0) = v(0) = 0$, $\partial u = K_{in}\theta_{in}$, and $\partial v = K_{out}\theta_{out}h$, where θ_{in} and θ_{out} are determined by the plan being evaluated. As long as $h < H$, we have $\partial h = (K_{in}\theta_{in} - K_{out}\theta_{out}h)/A$. If $\mathrm{trans}(\uparrow, h, H)$, then we have $\partial h = 0$, and, if $\mathrm{trans}(\downarrow, r, H)$ where $r = K_{in}\theta_{in}/K_{out}\theta_{out}$, we are back to $\partial h = (K_{in}\theta_{in} - K_{out}\theta_{out}h)/A$. To determine the total amount of fluid spilled, we have to set up rules to handle the various possibilities for $h < H$ and $v < V$. The complete Prolog representation is shown in Figure 2. During projection, every time that h rises to height, r falls to height, or v rises to volume, the conditions for certain projection rules shown in Figure 2 are met, and these rules are used to generate tokens specifying new models for various parameters.

For many planning problems, it is convenient to define a special function of time for evaluating alternative plans. In our simple example, this evaluation function is sp(t) which is the sum of the fluid spilled from either tank during the evaluation interval. If the plan consists of the five discrete events shown in Figure 2 and the evaluation interval is $\langle 0, 12 \rangle$, then we can evaluate the plan using the query holds(12,sp=S) which returns with S bound to 1.37. The response time is negligible for this query. The algorithm is guaranteed to terminate if the projection interval is finite. The complexity of projection is largely determined by the set of causal rules. For the sorts of rules we have encountered in our planning problems, projection is at worst a small polynomial in the size of the set of rules and initial conditions.

Conclusion

The current implementation of our hybrid calculus is convenient to use and remarkably fast for a prototype system. It still lacks much of functionality of our previous temporal database systems [Dean, 1989]. The current system has only limited ability to reason about uncertainty in either the time of occurrence of events or the initial values of parameters. However, uncertainty is difficult to handle even with discretely changing parameters and boolean variables [Dean and Boddy, 1988], and it appears that many of the techniques we have developed for handling uncertainty involving discrete change also apply in the continuous case.

The primary advantage of the hybrid system described in this paper over most temporal reasoning systems is its increased expressiveness and precision. It is clearly possible to model continuous processes using discrete approximations, but such approximations are often clumsy to formulate and sacrifice precision in order to achieve a reasonable level of performance. In our hybrid system, physical phenomena that are naturally modeled as continuous processes can be done so in a mathematical language designed for that purpose, and discrete processes can be modeled using first-order tem-

poral logic which is well suited for that purpose. The use of numerical methods for solving systems of ordinary differential equations gives the modeler a great deal of flexibility, and provides more than ample precision for the modeling tasks we have encountered so far. In addition, for projection problems of the sort encountered in many planning applications, our system subscribes to Sandewall's semantics up to the precision of the underlying numerical methods. Finally, and perhaps most importantly, we are now able to easily reason about planning problems that were impossible or at least prohibitively complicated to do so previously.

References

[Allen, 1984] James Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[Brachman et al., 1989] Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter, editors. *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*. Morgan-Kaufman, Los Altos, California, 1989.

[Dean and Boddy, 1988] Thomas Dean and Mark Boddy. Reasoning about partially ordered events. *Artificial Intelligence*, 36(3):375–399, 1988.

[Dean and McDermott, 1987] Thomas Dean and Drew V. McDermott. Temporal database management. *Artificial Intelligence*, 32(1):1–55, 1987.

[Dean, 1989] Thomas Dean. Using temporal hierarchies to efficiently maintain large temporal databases. *Journal of the ACM*, 36(4):687–718, 1989.

[Hendrix, 1973] Gary Hendrix. Modeling simultaneous actions and continuous processes. *Artificial Intelligence*, 4:145–180, 1973.

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

[Ralston and Rabinowitz, 1978] A. Ralston and P. Rabinowitz. *A First Course in Numerical Analysis*. McGraw-Hill, New York, 1978.

[Rayner, 1989] Manny Rayner. Did newton solve the "extended prediction problem?". In Brachman et al. [1989], pages 381–385.

[Sandewall, 1989] Erik Sandewall. Combining logic and differential equations for describing real-world systems. In Brachman et al. [1989], pages 412–420.

[Shampine and Gordon, 1975] L. F. Shampine and M. K. Gordon. *Computer Solution of Ordinary Differential Equations*. W. H. Freeman and Company, 1975.

[Shoham, 1988] Yoav Shoham. *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence*. MIT Press, Cambridge, Massachusetts, 1988.