Qualitative Reasoning about Continuous Processes with the Situation Calculus

Todd G. Kelley
Department of Computer Science
University of Toronto
Toronto, Canada M5S 1A4
tgk@cs.toronto.edu

Abstract

A fundamental aspect of general reasoning about physical systems is qualitative reasoning about continuous processes. The situation calculus is proposed as a framework for qualitative reasoning with some desirable properties that are less evident in the traditional framework, qualitative physics. However, in order to handle qualitative reasoning about *continuous* properties, we extend the concurrent, temporal situation calculus to include function variables. We show how parameters that are functions of time are specified by successor state axioms in terms of pieces of other functions of time. Finally, we present a qualitative bouncing ball example.

Introduction

A formal framework for representing and reasoning about dynamical worlds must address multiple challenges. A sample of the primary issues are agents and their actions driven by free will, perceptual actions and their effects on mental state, nondeterminism and chance, natural actions that are the result of mechanical laws of nature rather than free will, and change due to continuous processes. Also, knowledge may be qualitative or incomplete. A suitable formal framework should handle each of these issues in light of the frame, qualification, and ramification problems, which, in their general forms, are still open research problems.

The situation calculus language (McCarthy & Hayes 1969) is showing promise as a formal framework for representing and reasoning about dynamical worlds in the presence of these challenges. For example, Reiter presents a solution the frame problem for the situation calculus in (Reiter 1991), based on adhering to a particular form for the axioms, called successor state axioms, that represent knowledge of the effects of actions.

In this paper we take advantage of the progress with the situation calculus, while concentrating specifically on how to represent knowledge about natural actions, continuous processes, and discrete events in dynamical worlds, building on previous work by Sandewall (Sandewall 1989), Pinto (Pinto 1994) and Reiter (Reiter 1996b). In particular, we extend the situation calculus language of (Reiter 1996b) to facilitate the representation of qualitative knowledge of continuous processes.

We begin with a brief introduction to the situation calculus, and then list some desirable properties for a formal framework that are more evident in the situation calculus than in the traditional framework of QSIM-style qualitative physics. Next, we present the basics of the situation calculus approach to modeling physical systems that involve continuous change and discrete events. As is argued in the qualitative physics literature (Weld & de Kleer 1990) it is important to be able to represent qualitative or incomplete knowledge about physical systems. Addressing this need, we then introduce an extension to Reiter's concurrent, temporal situation calculus that facilitates the representation of qualitative knowledge of physical systems and their continuous processes. We illustrate the use of the extended language with a qualitative bouncing ball example to show how the extended situation calculus is used to reason qualitatively about physical systems. We conclude with a discussion of future work, namely, implementing a simulator for the extended language.

Situation Calculus

The situation calculus is designed to formalize the behavior of dynamically changing worlds. A situation is a history of action occurrences, and any situation identifies a complete evolution of the world. In particular, a situation does not denote a state. A world state is comprises the values of *fluents* which are functions or relations whose values or true values, respectively, vary according to situation. In a situation calculus specification (i.e. model of a world), *successor state* axioms specify the state of the world after it has evolved according to a particular course of action (i.e. situation), and *precondition axioms* specify which courses of action are possible. The following sections elaborate on these ideas.

Naming courses of action

The mechanism for all change in dynamical worlds is one or more agents, perhaps including Nature, performing named, instantaneous, *actions*. *Situations* are histories of action occurrences, each denoting a different evolution of the world. The constant symbol S_0 denotes the situation in which no actions occur. Other than S_0, all situations have names of the form, $do(\alpha, \sigma)$, intuitively meaning the result of doing action α just after the world has evolved according to situation σ. Actions are denoted by functions, with the time t of the action occurrence being the last parameter. A coherent set of actions is a *concurrent action*, where *coherent* means at least that each action in the set has the same time of occurrence. For example, consider the situation

$$do(\{stop\_talking(T_2)\},\\ do(\{begin\_walking(T_1), begin\_talking(T_1)\}, S_0),$$

which denotes the world history in which an agent begins walking and talking at time T_1, and then stops talking at time T_2. The agent would not be talking after this course of action, but she would still be walking.

The State of the World

The state of a world resulting from a certain course of action is determined by the values of *fluents.*

Relational fluents are denoted by predicate symbols taking a situation term as their last argument. These relations represent what is true about the world after carrying out the course of action specified by their situation argument. For example, the fluent, $happy(p, s)$ might mean that person p is happy in s. Note that a situation is not a state, but a history of action occurrences; so we take "p is true *in* s" to mean, "p is true *after carrying out, in order, all and only the actions specified by* s".

Functional fluents are functions whose value varies from one situation to another. For example, $const\_position(ball, s)$ might denote the real-valued constant position of a ball in situation s.

Specifying Which Courses of Action Can Happen

Precondition axioms determine the conditions under which an action is possible. A situation calculus specification of a world includes one precondition axiom for each simple action. For example, the precondition axiom for a $bounce(t)$ action might state that a ball bounces iff it is falling and it is at the floor.

Consider the function, $diff(f)$, which takes a function, f, as its argument and has as its value the function which is the derivative of f. We might write $diff(h(s))(T) < 0$ to say that the value of the time rate of change of the height function of a ball at time T in s is less than 0. In other words, the ball is falling at time T. Hence, the precondition axiom for the $bounce(t)$ action is

$$\begin{aligned} &Poss(bounce(t), s) \equiv \\ &\quad t \geq start(s) \wedge h(s)(t) = 0 \wedge \\ &\quad diff(h(s))(t) < 0 \end{aligned} \tag{1}$$

where $Poss(a, s)$ means that action a is possible in situation s.

The Results of Courses of Action

The ways in which the values of fluents are affected by action occurrences are determined by *successor state axioms.* A situation calculus specification of a world includes one successor state axiom for each fluent. The axiom specifies all individual conditions under which the fluent will change, and how the fluent changes under those conditions.

Consider a world where a ball can bounce, and it can be caught, and no other actions can affect it. The successor state axiom for the $h(s)$ fluent would state that the value of $h(do(c, s))$ depends upon what simple actions are in c. If c contains a *bounce* and not a *catch*, the ball's velocity reverses. If c contains a *catch*, the ball's position becomes constant where it is caught. If c contains neither a *bounce* nor a *catch*, the behavior of the ball's position remains unchanged. Successor state axioms embody Reiter's (Reiter 1991) solution to the frame problem.

Why the Situation Calculus?

Given that qualitative physics provides a modeling language, and a system for simulating models in that language, why do we look to the situation calculus as a framework? We believe that a framework for qualitative reasoning about physical processes should have the following properties.

The language should be able to handle the full range of degrees of incompleteness of knowledge. Ideally, it should be possible to represent all of, and only, what is known about a physical scenario, whether very little is known about it, or everything is known about it. The qualitative physics modeling language has been designed primarily to represent physical situations where equations, or relationships between quantities, are not known exactly. Hence, it is quite natural for the qualitative physics specification language to incorporate *a priori* abstractions of the quantities and their relationships, namely the abstractions inherent in the qualitative differential equation (QDE). These abstractions may force the modeler to discard knowledge, but the qualitative physics specification writer does not regard this as a problem, but rather as part of a specification methodology. In fact, a procedure for translating from ordinary differential equations into qualitative differential equations is part of the repertoire of qualitative physics.(See, for example, Section 3.3.1, *Abstracting structure from ODE to QDE*, in (Kuipers 1994).) The situation calculus, on the other hand, gives the specification writer the freedom (and responsibility) to choose whether such abstractions are appropriate.

The transition from specifying partial knowledge to specifying complete knowledge should be seamless. The task of enhancing the standard QSIM algorithm to handle *semi*-quantitative information has been addressed (Kuipers & Berleant 1988; 1990; Kay & Kuipers 1993). However, these approaches involve an additional component of processing that takes place after the QSIM algorithm has produced its qualitative behavior representation. The additional processing takes the semi-quantitative information and uses constraint satisfaction to discard some of the behaviors passed on by the basic QSIM algorithm. The quantitative information is not integrated into the base representation language. In situation calculus specifications, all knowledge, whether complete or incomplete, is expressed in the same general manner in the representation language.

The language should have a clear semantics. QSIM simulation has an operational semantics, because much knowledge, for example the transition tables of (Kuipers 1986), is embodied in the QSIM algorithm itself. Therefore, the semantics of a QSIM specification is unclear unless the reader has full knowledge of the rather significant QSIM algorithm. The situation calculus, being a logical language, has a standard logical semantics, accessible to a broad audience who are familiar with mathematical logic.

Specifications should be self-contained. The QSIM specification language is not self-contained, because the semantics of a QSIM specification is in terms of the QSIM algorithm. On the other hand, a situation calculus specification is self-contained because it alone defines the behaviors of the specified scenario.

Wherever possible, standard mathematical notation should be used. The qualitative physics representation language employs a mathematical notation peculiar to its qualitative mathematics. With the situation calculus, it is natural to use standard mathematical notation when the axiomatizer assumes the standard interpretation of the real numbers and their relations.

Modeling physical systems

Processes

Much of the change in a dynamical world is thought of as resulting from *processes*. Bringing water to a boil, and filling a gas tank are examples of processes. Although the notion of a process is not a fundamental aspect of our knowledge representation paradigm, it is an important concept, and in this section we explain how processes are represented.

A process in the context of this paper is a manifestation of continuous change in one or more real-valued quantities as time progresses. Every process is initiated, altered, or terminated by an instantaneous action.

A process is not directly tied to a particular sequence of actions. For example, consider a bathtub-filling process, in which the level of water in the bathtub is the real-valued quantity that changes. If the stopper is in the drain, this process might be initiated by turning on the faucet, and terminated by turning off the faucet. On the other hand, if the faucet is already on and water is running, but the stopper is not in the drain, this process might be initiated by inserting the stopper in the drain, and terminated by removing the stopper from the drain.

An important point to note is that no action has duration—all actions are instantaneous. A proposed "action" such as "*wait_a_minute*" is not an action at all. However, it could be represented as a process, in which the quantity that changes is the time waited, initiated by a *begin_waiting* action, and terminated by an *end_waiting* action after a minute has elapsed.

Continuous processes in the situation calculus are represented using functional fluents. The important idea, due to (Sandewall 1989) and (Pinto 1994), is that although a continuous process involves continuous change in the values of one or more parameters, the value of a particular parameter can be considered to have a *behavior* that does not change in a particular situation. The behavior of the parameter is constant in a situation, although the parameter itself varies according to some (just one) function of time.

To illustrate this idea, consider a ball that, upon begin dropped, falls under the influence of gravity. To represent the time-varying height of the ball, we use a functional fluent, $h(s)$, which denotes a *function of time* (as opposed to a real-valued height). The second-order term, $h(s)(t)$, denotes the real-valued height of the ball at time t in situation s. So, $h(do(drop(T_1), S_0))$ would denote a function of time, t, representing a falling behavior, perhaps $h(S_0)(T_1) - 1/2g(t - T_1)^2$, where $h(S_0)(T_1)$ denotes the position of the ball at the time the $drop(T_1)$ action occurs in S_0.

Natural actions

Natural actions, as opposed to agent-performed actions, are actions that happen mechanically without the "free will" of any agent. Natural actions have the special property that whenever one is possible, it *must* occur, provided that no earlier action prevents it from occurring. For example, when a *bounce* action is possible for a ball (i.e. it is at the floor and moving downward), it must bounce at exactly that time, unless a previous action (perhaps a *catch*) prevents the bounce.

A domain of discourse in which all actions are natural is said to comply with Reiter's (Reiter 1996b) Natural World Condition (*NWC*). This condition is true of worlds that consist of only a physical system that evolves without any agent intervention. The behavior of these worlds is deterministic, and can therefore be simulated.

Natural actions occur in a situation s at the *least natural time point*, $lntp(s,t)$:

$$\begin{aligned} & lntp(s,t) \overset{def}{=} \qquad (2) \\ & \quad (\exists a)[natural(a) \land Poss(a,s) \land time(a) = t] \land \\ & \quad (\forall a)[natural(a) \land Poss(a,s) \supset time(a) \geq t]. \end{aligned}$$

Informally, the least natural time point is the earliest time at which any natural action can possibly occur in a situation. The Least Natural Time Point Condition (*LNTPC*) is the following:

$$(\forall s).(\exists a)[natural(a) \land Poss(a,s)] \supset (\exists t)lntp(s,t). \quad (3)$$

This condition states that every situation in which there is a possible natural action has a least natural time point (i.e. if there is any possible action, then there is an *earliest* possible action).

Reasoning about physical systems in the situation calculus is concerned mostly with *executable* situations. Executable situations are consistent with the laws of Nature, which means all natural actions occur iff they are required to by Nature. The following theorem defines the executable situations for natural worlds.

$$\begin{aligned}&LNTPC \wedge NWC \supset \\ &\quad executable(do(c,s)) \equiv \{executable(s) \wedge \\ &\quad Poss(c,s) \wedge start(s) \leq time(c) \wedge \\ &\quad (\forall a)[a \in c \equiv Poss(a,s) \wedge lntp(s,time(a))]\}. \quad (4)\end{aligned}$$

Informally, this states that in any world for which $LNTPC$ and NWC are true, a situation is executable iff it is the result of doing a possible concurrent action in an executable situation, and every simple action in the concurrent action occurs at the least natural time point of that situation.

Also,

$$executable(S_0). \qquad (5)$$

Therefore, simulating a situation calculus specification of a physical system amounts to finding executable situations: starting with a situation, usually S_0, finding the possible concurrent action that occurs at the least natural time point of that situation, and then recursively repeating this process on the new situation resulting from doing the action. If there is no possible concurrent action in a situation, that situation is stable, and the simulation terminates.

Adding a function sort to the situation calculus

In this section, we discuss an extension to Reiter's concurrent, temporal situation calculus (Reiter 1996a). Also, we discuss how the extended language can be used to express knowledge, qualitative or not, about parameters that are functions of time, and how their behavior is affected by actions.

The extension

The extension consists of adding to the alphabet

- countably infinitely many one-place function variables of sort *function*
- A finite or countably infinite number of function symbols of sort *action* × *situation* → *function*.

The latter are *helper functions*, one for each parameter that varies continuously in the application. Intuitively, a parameter that varies continuously is piecewise equal to a selection of the functions "returned" by its corresponding helper function. The naming convention for helper functions is that they have the same name as the corresponding fluent, but with an h subscript.

Adding the function variables to the language allows the representation of important forms of *qualitative* knowledge about the behavior of a parameter that is a function of time. For example, we will often want to represent the knowledge that an object in the real world follows a classical trajectory without knowing what the trajectory is. Besides the knowledge that its trajectory is classical, often the only knowledge we will have about a parameter's behavior is that the time derivative of its path is greater than or less than zero on some interval. We express this knowledge making use of the following function and predicates:

- function symbol $diff : function \to function$. The intended interpretation of $diff(f)$ (abbreviated f') is that it denotes the function that is the derivative of the function f.
- predicate symbol $continuous : function \times object$. The intended interpretation of $continuous(f,t)$ is that function f is continuous at t.
- predicate symbol $diffable : function \times object$. The intended interpretation of $diffable(f,t)$ is that the derivative of function f exists at t.

To represent the notion of a classical trajectory, we will use the following abbreviation:

$$\begin{aligned}ct(f) \overset{def}{\equiv} &[(\forall t).continuous(f,t) \wedge diffable(f,t) \wedge \\ &continuous(diff(f),t) \wedge diffable(diff(f),t)] \quad (6)\end{aligned}$$

A general example

In this section we explain how pieces of functions of time are used to represent (perhaps incomplete) knowledge about parameters that are functions of time.

Each such parameter is represented by a *function*-valued functional fluent, which has a corresponding helper function. Consider a parameter represented by the *function*-valued functional fluent $h(s)$, with helper function $h_h(s)$. The intuitive idea is that if $S_n = do(a_n, do(a_{n-1}, do(\ldots, S_0)\ldots))$, $h(S_n)$ is a function piece-wise equal to the functions, $h_h(a_i, S_i)$ (and $h(S_0)$), for any a_i that affects the behavior of the h parameter.

Suppose that h is affected by only two kinds of actions, $a_{dn}(t)$, and $a_{up}(t)$. Also, suppose we know that any $a_{dn}(t)$ action causes the h parameter to follow a monotonically decreasing classical trajectory, and any $a_{up}(t)$ action causes the h parameter to follow a monotonically increasing classical trajectory. That is all we know.

The successor state axiom for $h(s)$ would be

$$\begin{aligned}&h(do(a,s)) = f \equiv \\ &\quad (\forall t).t \geq time(a) \wedge f(time(a)) = h(s)(time(a)) \wedge \\ &\quad ct(f) \wedge f(t) = h_h(a,s)(t) \wedge \\ &\quad\quad [(\exists t_1).a = a_{dn}(t_1) \wedge (\forall t_2)diff(f)(t_2) < 0 \vee \\ &\quad\quad\quad a = a_{up}(t_1) \wedge (\forall t_2)diff(f)(t_2) > 0] \vee \\ &\quad [t < time(a) \vee (\forall t_1).a \neq a_{dn}(t_1) \wedge a \neq a_{up}(t_1)] \wedge \\ &\quad\quad f(t) = h(s)(t). \qquad (7)\end{aligned}$$

This says that $h(do(a,s))$ is a function, call it f, which has the following properties:

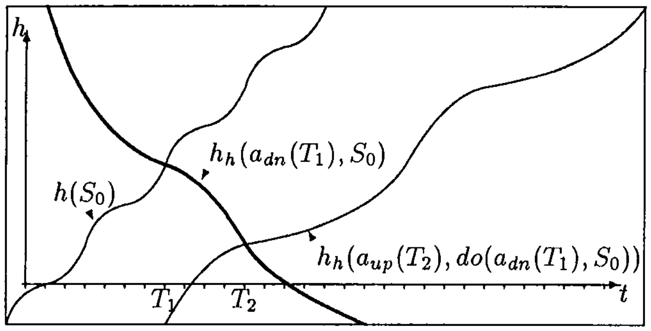

Figure 1: Three functions. Suppose that model U assigns each of the three functions named in the figure the graph indicated by the arrow.

- for times $t \geq time(a)$,
  - the value of f when the action occurs is the same as the value of the parameter at that time in the previous situation,
  - f is a classical trajectory,
  - on the interval $[t, \infty)$, f is equal $h_h(a, s)$,
  - if the action is an a_{dn} action, then f is monotonically decreasing, and
  - if the action is an a_{up} action, then f is monotonically increasing;
- for times $t < time(a)$, or if the action is neither an a_{up} nor an a_{dn} action, then f is the same as in the previous situation, as if no action occurred at all.

This is qualitative knowledge, since many functions satisfy all of the above properties. That is, different models (in the logical sense) will assign different functions to the symbol $h(S_1)$ for some S_1. Consider a model $\mathcal{U}$ such that $\mathcal{U}$ assigns $h(S_0)$, $h_h(a_{dn}(T_1), S_0)$ and $h_h(a_{up}(T_2), do(a_{dn}(T_1), S_0))$ the respective graphs shown in Figure 1. Then, for model $\mathcal{U}$, Figure 2 shows the graph of $h(do(a_{dn}(T_1), S_0))$, and Figure 3 shows the graph of $(h(do(a_{up}(T_2), do(a_{dn}(T_1), S_0)))$. It may seem odd that the function $h(s)$ says things about the parameter h even before the start time of s. This apparent oddness would probably stem from the habit of saying something is true *in a situation* s; however, we must keep in mind that every situation is a complete world history, not a state. Even S_0 is a complete world history in which nothing at all happens. So, as long as the things that happen are only all the actions specified by s, in order, then the value of the parameter h for any $t \in (-\infty, \infty)$ is the value of $h(s)(t)$.

Qualitative bouncing ball

In this section we work through an example where a ball is moving up as if it has just bounced or been tossed up, and we deduce that later, it bounces. We have the following knowledge of the ball world.

There is only one kind of action in this world, the *bounce* actions, all natural, possible only when the ball

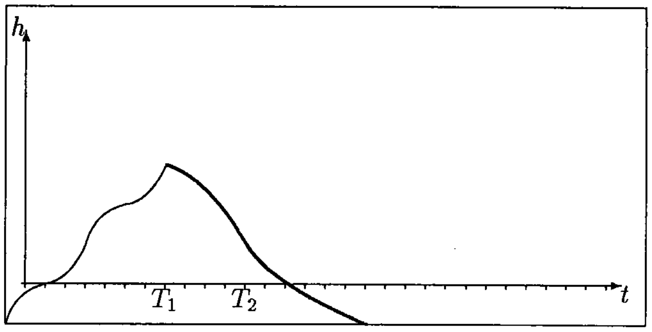

Figure 2: Given Figure 1, this is a graph of the function $h(do(a_{dn}(T_1), S_0)))$ for model U.

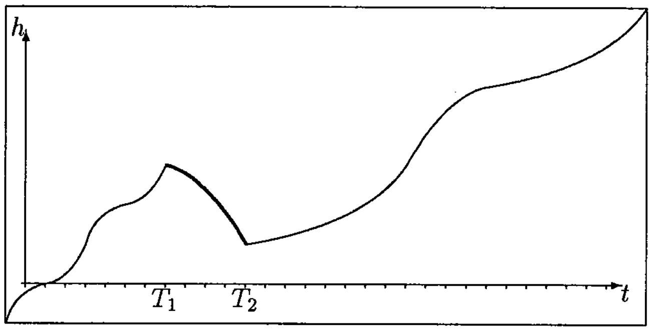

Figure 3: Given Figure 1, this is a graph of the function $h(do(a_{up}(T_2), do(a_{dn}(T_1), S_0))))$ for model U.

is at the floor ($h = 0$), and approaching the floor:

$$Poss(bounce(t), s) \equiv$$
$$t \geq start(s) \wedge h(s)(t) = 0 \wedge h(s)'(t) < 0 \quad (8)$$

In the initial situation, the ball is above the floor, moving up, and has a negative acceleration:

$$h(S_0)(T_0) > 0 \wedge h(S_0)'(T_0) > 0 \wedge \quad (9)$$
$$(\exists c \forall t) h(S_0)''(t) < c < 0, \quad (10)$$

where T_0 is the arbitrary start time of S_0. We do not know exactly what the acceleration is, because air resistance exerts a varying force on the ball.

The *bounce* action causes the ball to have a positive velocity, and a negative acceleration:

$$\begin{aligned}
&\text{h(do(a,s))} = f \equiv \\
&\quad (\forall t).t \geq time(a) \wedge ct(f) \wedge f(t) = h_h(a, s)(t) \wedge \\
&\quad f(time(a)) = h(s)(time(a)) \wedge \\
&\quad (\exists t_1) a = bounce(t_1) \wedge \\
&\quad f'(t_1) > 0 \wedge (\forall t_2) f''(t_2) < C < 0 \vee \\
&\quad [t < time(a) \vee (\forall t_1) a \neq bounce(t_1)] \\
&\quad \wedge f(t) = h(s)(t). \qquad (11)
\end{aligned}$$

We know of classical trajectories that:

$$f'(t) < 0 \supset [t_1 < t_2 \supset f(t_1) > f(t_2)] \quad (12)$$
$$f'(t) < C < 0 \supset$$
$$[f(t_1) > x \supset (\exists t_2) t_2 > t_1 \wedge f(t_2) = x] \quad (13)$$

With these axioms, we can prove the following.

$$executable(S_0) \quad \text{(by definition)} \quad (14)$$

$$(\exists t)h'(S_0)(t) = 0 \quad \text{(from 9,10,13)} \quad (15)$$

$$(\exists t).h'(S_0)(t) < 0 \wedge h(S0)(t) = 0$$
$$\text{(from 9,10,12,13,15)} \quad (16)$$

$$(\exists t)Poss(bounce(t), S_0) \quad \text{(from 8,16)} \quad (17)$$

$$(\exists t)executable(do(bounce(t), S_0) \quad \text{(from 4,14,17)} \quad (18)$$

$$h(do(bounce(T_2), S_0))'(T_2) > 0 \wedge$$
$$(\forall t)h(do(bounce(T_2), S_0))''(t) < C < 0 \quad \text{(from 11)} \quad (19)$$

We have proved that the ball will stop, reverse, reach the floor, and bounce, without knowing the exact functions for its velocity and acceleration.

Future work and discussion

We have shown how the situation calculus can be used to represent knowledge, perhaps qualitative, of dynamical worlds involving discrete events and continuous processes. An obvious next step is to implement a simulator that takes a specification (model) of a world, and prints out its possible evolutions. A PROLOG technology implementation that handles non-qualitative cases is discussed in the context of modeling a toilet in (Kelley 1996b) and a steam boiler controller in (Kelley 1996a).

A simulator that handles qualitative cases is currently being developed. The simulator for qualitative cases operates in basically the same way as non-qualitative cases, but it includes in addition a PROLOG procedure to do qualitative comparisons of the values of quantities. All comparisons are diverted to this procedure, which will attempt to determine the result of the comparison: whether one value is greater than , equal to, or less than the other value. In cases where the answer can be produced, the simulation proceeds normally. In cases where the answer cannot be produced because knowledge of the values is not sufficient, the procedure *assumes* each of the three possible results in turn, asserting the assumption into the PROLOG database, and the simulation proceeds. When the simulation terminates, a possible world evolution is printed, and that assumption, resulting in that particular world evolution, is printed out with any other assumptions that were needed during that evolution's derivation. Then the simulator can backtrack to the point where the qualitative comparison procedure made an assumption, retract it, assume the next alternative, and proceed again. Thus a simulation of a qualitative specification will result in several possible world evolutions, and the simulator prints out the conditions that would give rise to each one.

References

Kay, H., and Kuipers, B. J. 1993. Numerical behavior envelopes for qualitative models. In *Proc. of the 11th National Conf. on Artificial Intelligence*, 606–613. AAAI/MIT Press.

Kelley, T. G. 1996a. Modeling complex systems in the situation calculus: a case study using the dahgstul steamboiler problem. In Aiello, L. C.; Doyle, J.; and Shapiro, S. C., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96).* San Francisco, CA: Morgan Kaufmann Publishers.

Kelley, T. G. 1996b. Reasoning about physical systems with the situation calculus. In *COMMON SENSE '96: the third symposium on logical formalizations of commonsense reasoning.*

Kuipers, B. J., and Berleant, D. 1988. Using incomplete quantitative knowledge in qualitative reasoning. In *Proc. of the 7th National Conf. on Artificial Intelligence.*

Kuipers, B. J., and Berleant, D. 1990. A smooth integration of incomplete quantitative knowledge in qualitative reasoning. Technical Report AI90-122, University of Texas, Austin, Texas.

Kuipers, B. 1986. Qualitative simulation. *Artificial Intelligence* 29:289–338.

Kuipers, B. 1994. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge.* Cambridge Massachusetts: The MIT Press.

McCarthy, J., and Hayes, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 4.* Edinburgh, Scotland: Edinburgh University Press. 463–502.

Pinto, J. A. 1994. *Temporal Reasoning in the Situation Calculus.* Ph.D. Dissertation, University of Toronto, Toronto, Ontario, Canada.

Reiter, R. 1991. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., ed., *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy.* San Diego, CA: Academic Press. 359–380.

Reiter, R. 1996a. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems.* To appear.

Reiter, R. 1996b. Natural actions, concurrency and continuous time in the situation calculus. In Aiello, L. C.; Doyle, J.; and Shapiro, S. C., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96).* San Francisco, CA: Morgan Kaufmann Publishers.

Sandewall, E. 1989. Combining logic and differential equations for describing real-world systems. In *First International Conference on Principles of Knowledge Representation and Reasoning.* San Mateo, CA: Morgan Kaufmann.

Weld, D. S., and de Kleer, J., eds. 1990. *Readings in Qualitative Reasoning About Physical Systems.* San Mateo, California: Morgan Kaufmann.