In *13th National Conference on Artificial Intelligence (AAAI-96). AAAI Press, Portland, August 1996.*

Trajectory Constraints in Qualitative Simulation

Giorgio Brajnik*
Dip. di Matematica e Informatica
Università di Udine
Udine — Italy
giorgio@dimi.uniud.it

Daniel J. Clancy
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712
clancy@cs.utexas.edu

Abstract

We present a method for specifying temporal constraints on trajectories of dynamical systems and enforcing them during qualitative simulation. This capability can be used to focus a simulation, simulate non–autonomous and piecewise–continuous systems, reason about boundary condition problems and incorporate observations into the simulation. The method has been implemented in TeQSIM, a qualitative simulator that combines the expressive power of qualitative differential equations with temporal logic. It interleaves temporal logic model checking with the simulation to constrain and refine the resulting predicted behaviors and to inject discontinuous changes into the simulation.

Introduction

State space descriptions, such as differential equations, constrain the values of related variables within individual states and are often used in models of continuous dynamical systems. Besides continuity, which is implicit, these models cannot represent non–local information constraining the behavior of the system across time. Because qualitative simulation (Kuipers 1994; Forbus 1984) uses an abstraction of ordinary differential equations, it is based on a state space description too. The discretization of system trajectories into abstract qualitative states, however, makes the representation used by qualitative simulation amenable to the application of temporal formalisms to specify explicit across–time constraints. In general, these trajectory constraints can be used to restrict the simulation to a region of the state space in order to focus the simulation, simulate non–autonomous systems, reason about boundary condition problems and incorporate observations into the simulation.

TeQSIM (Temporally Constrained QSIM, pronounced *tek'sim*) restricts the simulation generated by QSIM (Kuipers 1994) to behaviors (*i.e.* sequences of qualitative states) that satisfy continuous and discontinuous behavioral requirements specified via *trajectory constraints*[1]. Figure 1 describes the relationship between the sources of constraining power within TeQSIM.

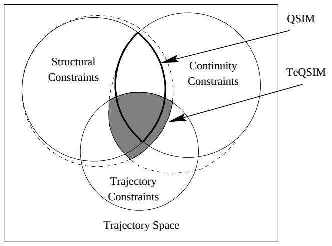
Trajectory constraints are formulated using a combination of temporal logic expressions, a specification of discontinuous changes and a declaration of external events. Temporal logic expressions are written using a variation of a propositional linear–time temporal logic (Emerson 1990) that combines state formulæ specifying both qualitative and quantitative information about a qualitative state with temporal operators, such as *until*, *always*, and *eventually*, that quantify such properties over a sequence of states. Temporal logic model checking is interleaved with the simulation to ensure that all and only the behaviors satisfying temporal logic expressions are included within the resulting description. Our logic extends the work done by Shults and Kuipers (Kuipers & Shults 1994; Shults & Kuipers 1996) in two ways. A three–valued logic is used to allow an expression to be conditionally entailed when quantitative information contained within the expression can be applied to a behavior to refine it. In addition, the model checking algorithm is designed to handle the incremental nature of a qualitative simulation. An undetermined result occurs whenever the behavior is not sufficiently determined to evaluate the truth of a temporal logic expression.

Discontinuous change expressions define when a particular discontinuity can occur and specify the new values for the variables that change discontinuously. This information is propagated through the model to determine the variables that are indirectly affected by the change.

Finally, *external events* enable the modeler to refer to otherwise unpredictable events and to provide a quantitatively bounded temporal correlation between the occurrence of these events and distinctions pre-

*The research reported in this paper has been performed while visiting the Qualitative Reasoning Group at the University of Texas at Austin.

[1] A *trajectory* for a tuple of variables $<v_1, \ldots, v_n>$ over a time interval $[a, b] \subseteq \Re^+ \cup \{0, +\infty\}$ is defined as a function τ mapping time to variable values defined over the set of the extended reals, *i.e.* $\tau : [a, b] \rightarrow (\Re \cup \{-\infty, +\infty\})^n$.



TeQSIM uses three sources of information to constrain a simulation: **structural constraints** are specified as equations relating model variables within individual states; implicit **continuity constraints** restrict the relationship between variable values across time to ensure the continuity of each variable; and **trajectory constraints** restrict the behavior of individual variables and the interactions between variables.

- Each point in the above diagram represents a real valued trajectory. A qualitative behavior corresponds to a region within this space of trajectories.
- Discontinuous changes specified by the user cause a relaxation of the continuity constraints applied during simulation (dotted line surrounding the continuity constraints). Incorporating external events into the simulation extends the set of trajectories consistent with the structural constraints (dotted line surrounding the structural constraints).
- The qualitative behaviors generated by QSIM correspond to the trajectories consistent with both the unextended structural constraints and the unrelaxed continuity constraints (thick boundary region), while the set of behaviors generated by TeQSIM corresponds to those trajectories consistent with all three constraint types (shaded region).

Figure 1: TeQSIM constraint interaction.

dicted by the model.

A Control Problem

TeQSIM has been applied to a variety of problems to address a range of tasks (Brajnik & Clancy 1996b). This section provides a simple example to demonstrate how trajectory information can be used to constrain a qualitative simulation in a realistic setting. Suppose that operators of a dam are told of a forecasted perturbation to the flow of water into the lake. When involved in risk assessment decision making, they face the control problem of determining how to react, in terms of operations on gates and turbines, in order to avoid flooding neither the lake nor the downstream areas.

To show some of TeQSIM's capabilities, we use a simple model of a lake, consisting of a reservoir, an incoming river and an outgoing river; lake level and outflow are regulated through a dam that includes a single floodgate. The implemented model uses quantitative information concerning Lake Travis, near Austin (TX), obtained from the Lower Colorado River Authority. Quantitative information is provided by numerical tables which, in this specific case, are interpolated in a step–wise manner to provide lower and upper bounds for any intermediate point. Table 2 shows a portion of the rating table of a floodgate of Lake Travis. Its columns indicate the lake *stage*, *i.e.* level with respect to the mean sea level, the gate *opening*, and the gate *discharge rate*. A similar table correlates the lake stage with its volume.

Stage (ft)	Opening (ft)	Discharge rate (cfs)
665.00	1.00	638.82
665.00	2.00	1277.65
...		
720.00	8.50	6200.00

Figure 2: Rating table for floodgates of Lake Travis.

The simulation starts from a state with initial values for stage and gate opening that guarantee a steady outflow in the downstream leg of the river. It is forecasted that in 2–3 days the inflow will increase and that for the subsequent 15–21 days there will be no substantial change. The task is to determine if there is any risk of overflowing the dam and, if so, what actions can be taken to prevent this.

We use TeQSIM to specify trajectory constraints on input variables: input flow rate and gate opening. The following trajectory constraints specify the perturbation to the inflow rate (`Colorado-up`).

```
(EVENT step-up :time (2 3))
(EVENT step-down :time (17 24))
(DISC-CHANGE (event step-up) ((Colorado-up (if* inf)
                                            :range (1500 1800))))
(DISC-CHANGE (event step-down) ((Colorado-up if*)))
```

The declaration of an event (*e.g.* `(EVENT step-up :time (2 3))`) defines a name for a time–point and provides quantitative bounds (*i.e.* between days 2 and 3 from the start of the simulation). The expression `(DISC-CHANGE (event step-up ((Colorado-up (if* inf):range (1500 1800))))` states that when event `step-up` occurs, the qualitative magnitude of `Colorado-up` will instantaneously change into the interval `(if* inf)` and its value will be bounded by the range [1500, 1800].

A simulation using these trajectory constraints shows that an overflow of the lake is indeed possible if no intervening action is taken. To guarantee that no overflow occurs, a significant opening action is required. To this end, we postulate that an opening action to at least 4 feet occurs after `Stage` reaches the `top-of-pool` threshold. We are interested in knowing the latest time at which such an action can occur to prevent an overflow. The previous trajectory specification is extended by including an additional event (corresponding to the opening of the gate), the corresponding discontinuous action (the gate opening changes from its initial value `op*`=1 to an intermediate value within `(op* max)` constrained to be greater than or equal to 4) and the ordering with respect to the threshold (using the temporal operator `BEFORE` applied to a state formula referring to the qualitative value of `Stage` and the external event). By focusing the simulation on behaviors that lead to an overflow condition (using the `EVENTUALLY` temporal operator applied to a state formula stating that `Stage` reaches the value `Top`), TeQSIM determines a lower bound for the temporal occurrence of actions leading to an overflow.

```
(EVENT open)
(DISC-CHANGE (event open)
             ((opening (op* max) :range (4 NIL))))
```

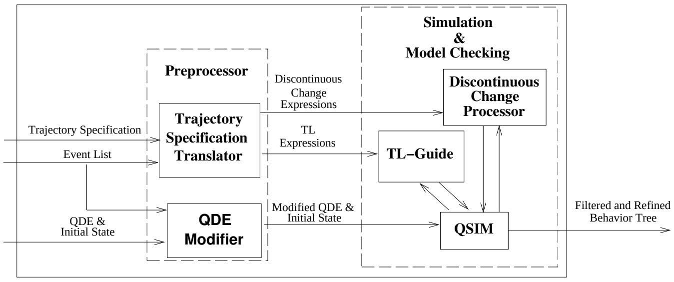


Figure 4: TeQSIM architecture.

```
(BEFORE (qvalue stage (top-of-pool NIL)) (event open))
(EVENTUALLY (qvalue stage (top NIL)))
```

This simulation tells us (figure 3) that if the gate is opened to at least 4 feet *after* 15.5 days then an overflow may occur. Taking the action *before* 15.5 days, however, will prevent such an outcome. After constraining the action to occur before 15.5 days and removing the `eventually` constraint, a third simulation produces only two behaviors, verifying that an overflow cannot occur. In a similar manner, we infer an upper bound of 6 ft for the size of the opening action, given a restriction on the outflow rate expressed via the temporal logic expression `(always (value-<= Colorado-dn 350))`.

It is worth noting the amount of uncertainty present in even such a simple problem: functions (especially the discharge rate) may be non–linear, numeric envelopes are based on a rough step–wise interpolation of tables, and the specification of input trajectories is uncertain (*i.e.* ranges for times and values). Nevertheless, with a few simple simulations a reasonably useful and reliable result has been achieved.

TeQSIM Architecture and Theory

TeQSIM can be divided into two main components. The *preprocessor* modifies the qualitative differential model and decomposes the trajectory specification into temporal logic and discontinuous change expressions. The *simulation and model checking* component integrates temporal logic model checking into the simulation performed by QSIM by filtering and refining qualitative behaviors according to a set of temporal logic expressions; it also injects discontinuous changes into the simulation. Figure 4 provides an overview of the system architecture.

The user provides trajectory constraints to TeQSIM in the form of a trajectory specification that consists of an external event list and a set of extended temporal logic and discontinuous change expressions. The external event list is a totally ordered sequence of named, quantitatively bound time points. Events are represented as landmarks of an auxiliary variable added to the model. The additional variable causes QSIM to branch on different orderings between external events and internal qualitative events identified during the simulation. The occurrence of external events is restricted by their quantitative bounds and the trajectory constraints specified by the modeler.

The following subsections include a summary of the formal framework developed for the trajectory specification language. A more detailed treatment of the language and main theorems, along with proofs and additional lemmas, is given in (Brajnik & Clancy 1996a; 1996b).

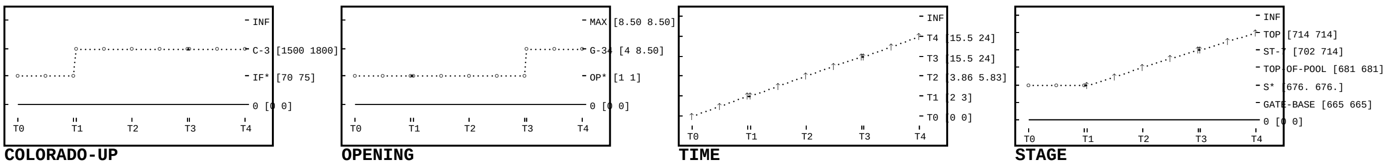
Guiding and refining the simulation

Model checking and behavior refinement are performed by the *Temporal Logic Guide* (TL–Guide). Each time QSIM extends a behavior by adding a new state, the behavior is passed to the TL–Guide. The behavior is refuted if it contains sufficient information to determine that each of its completions fail to satisfy the set of TL expressions. If the behavior conditionally models the TL expressions, then it is refined by incorporating relevant quantitative information contained within the TL expressions. Otherwise, the behavior is retained unchanged.

The trajectory specification language includes propositional state formulæ that can refer to qualitative and quantitative values of variables within states. Qualitative value information is specified using (`qvalue` v ($qmag$ $qdir$)) where v is a model variable, $qmag$ is a qualitative magnitude, and $qdir$ is one of {`inc`, `std`, `dec`}. `NIL` can be used anywhere to match anything. Such a proposition is true for a state exactly when the qualitative value of v in the state matches the description ($qmag$ $qdir$).

Path formulæ are defined recursively as either state formulæ or combinations of path formulæ using temporal and boolean operators. A state formula is true of a behavior if it is true for the first state in the behavior. The path formula (`until` p q), where both p and q are path formulæ, is true for a behavior if p holds for all suffixes of the behavior preceding the first one where q holds, while (`strong-next` p) is true for a behavior if it contains at least two states and p holds in the behavior starting at the second state. Other temporal operators can be defined as abbreviations from these two. We have extended those defined in (Shults & Kuipers 1996) to provide a more abstract language to simplify the specification of assertions.

Temporal logic formulæ are given meaning with respect to linear–time interpretation structures. These structures are extended from their typical definition (*e.g.* (Emerson 1990)) in order to accommodate the refinement of behaviors with quantitative information. In addition to defining a sequence of states and a propositional interpretation function, means for representing, generating and applying refinement conditions are provided. Refinement conditions are needed because the language provides quantitative propositions whose truth value cannot always be determined. When ambiguity occurs for a formula, then the interpretation is required to provide necessary and sufficient re-



TeQSIM produces two behaviors where `Stage` reaches `Top`. The first behavior is shown above. The opening action occurs at `T3`. The numeric bounds on this time–point shows that an overflow can occur only if the opening action is performed after 15.5 days. The second behavior provides similar results.

Figure 3: Lake simulation with opening actions leading to overflow.

finement conditions on quantitative ranges to disambiguate the truth value of the formula. A *refinement condition* is a boolean combination of inequalities between the partially known numeric value of a variable in a state and an extended real number.

The trajectory specification language contains potentially ambiguous state formulæ like (`value-<=` v n), where v is a variable and $n \in \Re \cup \{-\infty, +\infty\}$. The formula is true in a state s iff $\forall x \in \mathcal{R}(v,s) : x \leq n$, where $\mathcal{R}(v,s)$ denotes the range of possible numeric values for variable v in state s; it is false iff $\forall x \in \mathcal{R}(v,s) : n < x$; otherwise, it is conditionally true. In such a case, the refinement condition is that the least upper bound of the possible numeric values of v is equal to n (*i.e.* $v_s \leq n$, where v_s is the unknown value of v in s).

Applying a refinement condition to a state yields a new, more refined state. For example, the formula `(value-<= X 0.3)` generates the condition $X_s \leq 0.3$ when interpreted on a state s where $\mathcal{R}(X,s) = [0, 1.0]$. Applying the condition to s leads to a new state s' where $\mathcal{R}(X,s') = [0, 0.3]$.

Notice that ambiguity is not a purely syntactic property, but rather it depends on state information. For example, `(value-<= X 0.3)` is (unconditionally) true on a state where $\mathcal{R}(X,s) = [0, 0.25]$, but only conditionally true if $\mathcal{R}(X,s) = [0, 1.0]$. Due to potential ambiguity, two entailment relations are used to define the semantics of formulæ. The first one, called *models*, characterizes non–ambiguous true formulæ while the second one, called *conditionally models*, characterizes ambiguous formulæ.

To avoid hindering the simulation process, the usage of ambiguous formulæ must be restricted. The problem is that an arbitrary formula may yield several alternative refinement conditions. A disjunction of refinement conditions can be applied to states, but it requires the introduction of a new behavior that is qualitatively identical to the original behavior. For example, when interpreted on a particular state `(or (value-<= X 0.5) (value-<= Y 15))` may yield the condition $(X_s \leq 0.5 \vee Y_s \leq 15)$. Applying such a condition yields a state s' in which $\mathcal{R}(X,s') = [\ldots, 0.5]$ and a state s'' where $\mathcal{R}(Y,s'') = [\ldots, 15]$. A similar, more severe problem occurs with path formulæ.

The set of *admissible formulæ* is a syntactic restriction that excludes formulæ that may result in disjunctive conditions. Even though such a restriction reduces the expressiveness of the language, it does not have an important impact from a practical point. If the modeler adheres to the general principle that all important distinctions are made explicit in the qualitative model (*i.e.* introduces appropriate landmarks with associated numerical bounds instead of using quantitative bounds), then the restriction to admissible formulæ does not reduce the applicability of TeQSIM.

Discontinuous Changes

The injection of discontinuous changes into qualitative simulation consists of identifying when the change occurs and then propagating its effects through the model to determine which variables inherit their values across the change and which don't.

A discontinuous change is specified by `(disc-change` *precond effect*), where *precond* is a boolean combination of `qvalue` propositions and *effect* is a list of expressions of the form (*variable qmag* [`:range` *range*]). This expression is translated into the temporal logic path formula (`occurs-at` *precond* (`strong-next` *effect'*)) where *effect'* is a conjunction of `qvalue`, `value-<=` and `value->=` formulæ derived from *effect*. This formula is true for a behavior iff *effect'* is true for the state immediately following the first state in which *precond* is true.

The Discontinuous Change Processor monitors states as they are created and tests them against the preconditions of applicable discontinuous change expressions. A new state is inserted into the simulation following state s if the preconditions are satisfied and a discontinuous change is required to assert the effects in the successor states. A new, possibly incomplete state s' is created by asserting the qualitative values specified within the effects and inheriting values from s for variables not affected by the discontinuous change via continuity relaxation (see below). All consistent completions of s' are computed and installed as successors of s.

Continuity relaxation propagates the effects of a discontinuous change through the model by identifying potentially affected variables. The following assumptions are made: (i) *state* variables (variables whose time derivative is included in the model) are *piecewise–*C^1 (*i.e.* continuous everywhere, and differentiable everywhere except at isolated points); (ii) *non–state* variables are at least *piecewise*-C^0 (*i.e.* continuous everywhere except at isolated points); (iii) all discontinuous changes in input variables are explicitly specified; and (iv) the model is valid during the transient caused by a discontinuous change. These assumptions suffice to support an effective criterion, proven to be sound, for automatically identifying all the variables that are potentially affected by the simultaneous discontinuity of variables in a set $\Delta = \{V_1 \dots V_n\}$.

Given a qualitative differential model $\mathcal{M}$, a variable Z is *totally dependent* on a set of variables $\mathcal{A}$ (written $\mathcal{A} \rightsquigarrow Z$) iff $\mathcal{M}$ includes a non–differential, continuous relation $R(X_1 \dots X_i, Z, X_{i+1} \dots X_n)$ with $n \geq 1$ such that $\forall i$: $X_i \in \mathcal{A}$ or $\mathcal{A} \rightsquigarrow X_i$. For example, if $\mathcal{M}$ includes the constraint (`add` X Y Z) then $\{Y, Z\} \rightsquigarrow X$. Furthermore, let $TD(\mathcal{A})$ represent the set of variables totally dependent on $\mathcal{A}$ (i.e. $TD(\mathcal{A}) = \{X | \mathcal{A} \rightsquigarrow X\}$).

Let $\mathcal{E}$ be the set of input variables and $\mathcal{S}$ the set of state variables of $\mathcal{M}$. Then define $\mathcal{PD}_\Delta$ (the set of variables that are potentially affected by the discontinuity of variables in Δ) as the maximum set of variables of $\mathcal{M}$ that satisfies:

1. $\Delta \subseteq \mathcal{PD}_\Delta$ (by definition of Δ);
2. $\mathcal{S} \cap \mathcal{PD}_\Delta = \emptyset$ (by assumption (i));
3. $\mathcal{E} \cap \mathcal{PD}_\Delta = \Delta$ (by assumption (iii));
4. $TD(\mathcal{S} \cup \mathcal{E} - \Delta) \cap \mathcal{PD}_\Delta = \emptyset$ (by definition of total dependency, if Z totally depends on a set of continuous variables, then Z must be continuous too and cannot belong to $\mathcal{PD}_\Delta$).

Continuity relaxation handles discontinuous changes of variables in Δ by computing the set $\mathcal{PD}_\Delta$ so that, during a transient, variables in $\mathcal{PD}_\Delta$ are unconstrained and can change arbitrarily, whereas those not in $\mathcal{PD}_\Delta$ retain their previous qualitative magnitude. The direction of change (*i.e. qdir*) for all variables is assumed to be potentially discontinuous.

In the simulation shown in figure 3, the discontinuity occurring to `Opening` at `T3` cannot affect `Stage` because the latter is totally dependent on the state variable `Volume`. On the other hand, the discontinuity affects the magnitude of `Discharge-rate` (not shown in the figure) because none of the conditions above apply. Notice also how the discontinuities affect the *qdir* of variables.

Model Checking

The temporal logic model checking algorithm is designed to evaluate a QSIM behavior with respect to a set of temporal logic formulæ as the behavior is incrementally developed. This allows behaviors to be filtered and refined as early as possible *during* the simulation. Kuipers and Shults (1994) developed a model checking algorithm to prove properties about continuous dynamical systems by testing a *completed* simulation against temporal logic expressions. We have extended this work to deal with conditionally true formulæ and with behaviors that are not closed, *i.e.* still being extended by the simulator.

The model checking algorithm, described in (Brajnik & Clancy 1996b), computes the function τ: *Formulæ* $\times$ *Behaviors* $\rightarrow \{\mathtt{T}, \mathtt{F}, \mathtt{U}\} \times \mathcal{C}$, where $\mathcal{C}$ is the set of all possible refinement conditions, including the trivial condition `TRUE`. A definite answer (*i.e.* `T` or `F`) is provided when the behavior contains sufficient information to determine the truth value of the formula. For example, a non–closed behavior b will *not* be sufficiently determined with respect to the formula (`eventually` p) if p is false for all suffixes of b, since p may become true anytime in the future.

A behavior b is *sufficiently determined* with respect to a temporal logic formula φ (written $b \rhd \varphi$) whenever there is enough information within the behavior to determine a single truth value for all of its completions. If a behavior is not sufficiently determined for a formula, then `U` is returned by the algorithm. The definition of *sufficiently determined* (omitted due to space restrictions) is given recursively on the basis of the syntactic structure of the formula. We will write $b \not\rhd \varphi$ to signify that b is not sufficiently determined for φ.

Notice that indeterminacy is a property independent from ambiguity: the former is related to incomplete behaviors, whereas the latter deals with ambiguous information present in states of a behavior.

The following theorem supports our use of temporal logic model checking for guiding and refining the simulation.

Theorem 1 (TL–guide is sound and complete) *Given a QSIM behavior b and an admissible formula φ then TL–guide:*

1. *refutes b iff $b \rhd \varphi$ and there is no way to extend b to make it a model for φ.*
2. *retains b without modifying it iff*
   (a) *$b \rhd \varphi$ and b is a model for φ; or*
   (b) *$b \not\rhd \varphi$ and there is no necessary refinement condition C for refining b into a model for φ.*
3. *replaces b with b' iff*
   (a) *$b \rhd \varphi$ and b conditionally models φ and there exists C that is necessary and sufficient for refining b into a model for φ; or*
   (b) *$b \not\rhd \varphi$ and there is a necessary condition C for refining b into a model for φ.*

Proof. By induction on the length of φ; see (Brajnik & Clancy 1996b).

Discussion and Conclusions

We are currently exploring several directions to extend the expressiveness of the trajectory specification language. Enabling the comparison of magnitudes of variables across states (*e.g.* to specify a decreasing oscillation) requires a move from a propositional logic to some sort of first order logic. Expressing the *possibility* of a discontinuous change requires a more complex relationship between preconditions and effects of a discontinuous change. Addressing discontinuous feed–forward control problems requires that preconditions are specified using arbitrary temporal logic expressions, not simply state formulæ. Simulating hybrid discrete–continuous systems calls for a more flexible specification of partially ordered external events.

While the practical time–complexity of a TeQSIM simulation is dominated by quantitative inferences performed by QSIM, we are still investigating improvements to our algorithm with respect to complexity.

The incorporation of trajectory information into a qualitative simulation has been explored by DeCoste (1994), who introduces *sufficient discriminatory envisionments* to determine whether a goal region is possible, impossible or inevitable from each state of the space. Washio and Kitamura (1995) also present a technique that uses temporal logic to perform a *history oriented envisionment* to filter predictions. TeQSIM, within a rigorously formalized framework, provides a more expressive language not limited to reachability problems, refines behaviors as opposed to just filtering them, and incorporates discontinuous changes into behaviors.

Discontinuities have been investigated by Nishida and Doshita (1987), Forbus (1989), Iwasaki and colleagues (1995), and others. The continuity relaxation method adopted in TeQSIM is conceptually simpler, sound, widely applicable and practically effective.

Our trajectory specification language is similar in expressiveness to both Allen's *interval algebra* (Allen 1984) and Dechter, Meiri and Pearl's *temporal constraint networks* (Dechter, Meiri, & Pearl 1991). The usage of the language in TeQSIM, however, is quite different from these two formalisms. Instead of asserting temporal constraints in a database of assertions and querying if certain combinations of facts are consistent, TeQSIM checks that a database of temporally related facts generated by QSIM satisfy a set of temporal logic constraints.

TeQSIM supports a general methodology for incorporating otherwise inexpressible trajectory information into the qualitative simulation process. The correctness of TL–Guide, of the Discontinuous Change Processor, and of QSIM guarantee that all possible trajectories of the modeled system that are compatible with the model, the initial state and the trajectory constraints are included in the generated behaviors. In addition, the completeness of TL–Guide ensures that all behaviors generated by TeQSIM are potential models of the trajectory constraints specified by the modeler. For these reasons, and its limited complexity, TeQSIM can be applied to problems where QSIM alone would not be appropriate.

Acknowledgments

We thank Benjamin Shults for letting us use his TL program to implement TeQSIM. This work has taken place in the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Qualitative Reasoning Group is supported in part by NSF grants IRI–9216584 and IRI–9504138, by NASA grants NCC 2–760 and NAG 2–994, and by the Texas Advanced Research Program under grant no. 003658–242.

QSIM and TeQSIM are available for research purposes via `http://www.cs.utexas.edu/users/qr`.

References

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

Brajnik, G., and Clancy, D. J. 1996a. Guiding and refining simulation using temporal logic. In *Proc. of the Third International Workshop on Temporal Representation and Reasoning (TIME'96)*. Key West, Florida: IEEE Computer Society Press. To appear.

Brajnik, G., and Clancy, D. J. 1996b. Temporal constraints on trajectories in qualitative simulation. Technical Report UDMI–RT–01–96, Dip. di Matematica e Informatica, University of Udine, Udine, Italy.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.

DeCoste, D. 1994. Goal–directed qualitative reasoning with partial states. Technical Report 57, The Institute for the Learning Sciences, University of Illinois at Urbana–Champaign.

Emerson, E. 1990. Temporal and modal logic. In van Leeuwen, J., ed., *Handbook of Theoretical Computer Science*. Elsevier Science Publishers/MIT Press. 995–1072. Chap. 16.

Forbus, K. 1984. Qualitative process theory. *Artificial Intelligence* 24:85–168.

Forbus, K. 1989. Introducing actions into qualitative simulation. In *IJCAI–89*, 1273–1278.

Iwasaki, Y.; Farquhar, A.; Saraswat, V.; Bobrow, D.; and Gupta, V. 1995. Modeling time in hybrid systems: how fast is "instantaneous"? In *IJCAI–95*, 1773–1780. Montréal, Canada: Morgan Kaufmann Publishers, Inc.

Kuipers, B., and Shults, B. 1994. Reasoning in logic about continuous change. In *Principles of Knowledge Representation and Reasoning (KR–94)*. Morgan Kaufmann Publishers, Inc.

Kuipers, B. 1994. *Qualitative Reasoning: modeling and simulation with incomplete knowledge*. Cambridge, Massachusetts: MIT Press.

Nishida, T., and Doshita, S. 1987. Reasoning about discontinuous change. In *AAAI–87*, 643–648.

Shults, B., and Kuipers, B. J. 1996. Qualitative simulation and temporal logic: proving properties of continuous systems. Technical Report TR AI96–244, University of Texas at Austin, Dept. of Computer Sciences.

Washio, T., and Kitamura, M. 1995. A fast history–oriented envisioning method introducing temporal logic. In *Ninth International Workshop on Qualitative Reasoning (QR–95)*, 279–288.