In *Tenth International Workshop on Qualitative Reasoning (QR '96)*
*Fallen Leaf Lake, CA; May, 1996.*

# Temporal Constraints on Trajectories 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 new method for specifying temporal constraints on trajectories of dynamical systems and enforcing them during qualitative simulation. Such constraints are otherwise inexpressible using standard qualitative reasoning techniques. Trajectory constraints can be used to restrict the simulation to a region of the state space and to inject discontinuities. This capability can be used to focus the simulation for larger, more complex simulations, 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. It combines the expressive power of qualitative differential equations with temporal logic by interleaving temporal logic model checking with the simulation to constrain and refine the resulting predicted behaviors and to inject discontinuous changes into the simulation.

The paper discusses the applicability of temporal constraints in tasks ranging from simulation to monitoring and control of continuous dynamical systems. We present a real–world control problem in the domain of water supply. Finally, the basic algorithm and theoretical results (soundness and completeness) are described.

## Introduction

State space equations that constrain the values of related variables within individual states are often used in models of continuous dynamical systems such as ordinary differential equations. These models do not allow the representation of non–local information constraining the behavior of the system across time except through the application of continuity. Qualitative simulation (Kuipers, 1994; Forbus, 1984) uses an abstraction of ordinary differential equations to specify structural equations based on a state space description. The discretization of trajectories into abstract qualitative states, however, makes the representation used by qualitative simulation amenable to the application of temporal formalisms to specify non–local trajectory constraints. In general, trajectory information can be used to restrict the simulation to a region of the state space. This capability can be used to focus the simulation for larger, more complex simulations, simulate non–autonomous systems, reason about boundary condition problems and incorporate observations into the simulation.

TeQSIM (Temporally Constrained QSIM, pronounced *tek'sim*) allows the modeler to specify both continuous and discontinuous behavioral information via *trajectory constraints*[1] that restrict the simulation. Trajectory constraints are formulated using a combination of temporal logic expressions, a specification of discontinuous changes and a declaration of external events not predictable from the model.

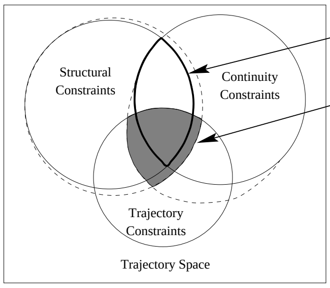
This paper provides a detailed description of the TeQSIM algorithm along with the presentation of an example demonstrating how trajectory constraints can be used to restrict a simulation. In addition, we present theoretical results showing that TeQSIM is sound and complete with respect to the trajectory constraints. Finally, an evaluation of this research is provided along with a discussion of some of the tasks that TeQSIM can be used to address.

## Conceptual Framework

Qualitative simulation uses a discretization of a continuous real valued trajectory space into qualitative states to represent a set of behaviors consistent with the structural constraints included in the qualitative differential equation (QDE) and continuity constraints applied during the simulation. Each qualitative behavior corresponds to a set of real–valued trajectories. Trajectory information can be used to further constrain the qualitative simulation and reduce ambiguity within the behavioral description. Figure 1 describes

[*]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 $\tau$ 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 variables within the model; implicit **continuity constraints** restrict the relationship between variable values across time to ensure the continuity of each variable; and **trajectory constraints** are specified via temporal logic expressions restricting the behavior of individual variables and the interactions between the behaviors of related 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.

the relationship between different sources of constraining power.

A billiards shot provides a very simple example of how trajectory information can be useful when reasoning about the behavior of a dynamical system. We are interested in deriving quantitative bounds on the velocity required to hit a successful billiards shot given an initial description of the table. (i.e. How hard and in which direction does the white ball need to be hit so that it strikes a designated ball and propels it into a particular pocket?) A standard qualitative simulation would derive a description of all potential behaviors including behaviors in which the shot is not successful. The modeler would then need to identify those behaviors that are consistent with the desired shot and extract the required information. TeQSIM allows the modeler to use trajectory constraints to specify boundary conditions on the desired behaviors to restrict the simulation to the region of the state space in which the cue ball strikes the target ball and the target ball hits the desired pocket. This example demonstrates only some of the benefits provided by trajectory constraints within TeQSIM.

Three different types of expressions are used to specify trajectory constraints: temporal logic expressions, discontinuous change specifications, and external event declarations. Temporal logic expressions are the primary method of specifying trajectory constraints. Temporal logic model checking is interleaved with the simulation process to ensure that only behaviors satisfying the set of temporal logic expressions are included within the resulting description. The temporal logic used is a variation of a propositional linear–time temporal logic (PLTL) (Emerson, 1990). It combines state formulae 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. For example, the expression `(until (qvalue X (nil inc)) (qvalue Y (Y* nil)))` states that the qdir of $X$ is `inc` (i.e. $X$ increases) until $Y$ reaches $Y*$.

Our logic extends work done by Shults and Kuipers (Kuipers and Shults, 1994; Shults and Kuipers, 1996) to query behavioral descriptions. Two extensions are required to constrain a simulation. 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 the description. 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 insufficiently determined to evaluate the truth of a temporal logic expression.

Discontinuous change expressions define when a particular discontinuity can occur and specify its immediate effects (i.e. new values for the variables that change discontinuously). This information is propagated through the model to determine the variables affected by the discontinuous change. Thus, the expression `(disc-change (qvalue X (X* std)) ((inflow (if* inf) :range (400 440))))` states that when $X$ reaches $X*$ and is `std` that `inflow` will instantaneously change into the interval `(if* inf)` and that the value will be within the range `(400 440)`.

Finally, the declaration of external events allows the modeler to provide a quantitatively bounded temporal correlation between the occurrence of these events and distinctions predicted by the model. References to external events included in both temporal logic and discontinuous change expressions allow the modeler to temporally constrain the information contained within these expressions. For example, the discontinuous change expression `(disc-change (event open) ((inflow if*)))` states that the inflow changes to `if*` when the `open` event occurs. Quantitative bounds on when the `open` event can occur are included in the

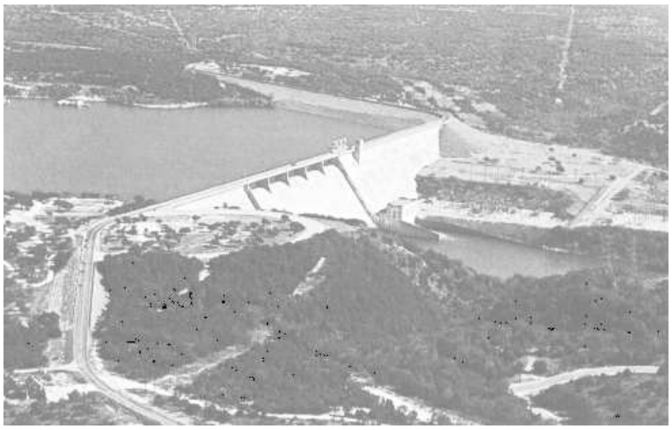


Figure 2: Lake Travis and Mansfield Dam, Austin, TX.

event declaration statement.

## A control problem

TeQSIM has been applied to a variety of problems to address a range of tasks. This section provides a simple example to demonstrate how trajectory information can be used to constrain a qualitative simulation in a realistic setting. Operators of a dam restraining a lake face the control problem of determining how to react (in terms of operations on gates and turbines) to a forecasted perturbation to the flow of water into the lake. We assume that operators are involved in risk assessment decision making instead of optimal resource management. A simple model of a lake, consisting of a reservoir, an incoming river and an outgoing river is used. The lake level and the outflow are regulated through a dam which includes a single floodgate. The model implemented uses quantitative information concerning Lake Travis, near Austin (see figure 2) obtained from the Lower Colorado River Authority. Information has been provided in 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 3 describes 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 sea mean level, the gate *opening*, and the gate *discharge rate*. A similar table correlates the lake stage with its volume.

The steady–state description of the simulated scenario consists of the lake having an initial stage of 676.25 feet, an inflow (called `Colorado-up`) in the range [70, 75] millions of cubic feet per day (mcfd) (or [810, 868] cfs) and a 1 foot opening of a single gate that guarantees a steady outflow in the downstream leg of the Colorado River(`Colorado-dn`). It is forecasted that in 2–3 days the inflow will increase up to [1500, 1800] mcfd (*i.e.* approx. [17000, 21000] cfs) and that for the subsequent 15–21 days there will be no substantial change. The task is to determine if there is any risk

| Stage (ft) | Opening (ft) | Discharge rate (cfs) |
|---|---|---|
| 665.00 | 1.00 | 638.82 |
| 665.00 | 2.00 | 1277.65 |
| ... | | |
| 665.00 | 8.50 | 5430.00 |
| 670.00 | 1.00 | 721.82 |
| ... | | |
| 720.00 | 8.00 | 5867.65 |
| 720.00 | 8.50 | 6200.00 |

Figure 3: Rating table for floodgates of Lake Travis.

of overflowing the dam and, if so, which actions could be taken to prevent this.

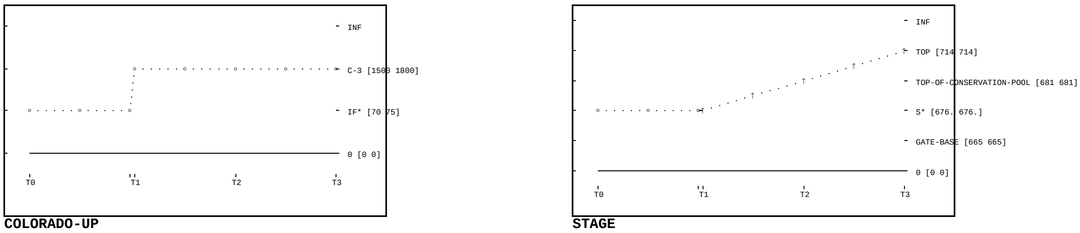
We use TeQSIM to specify trajectory constraints on input variables: input flow rate and gate opening. The following trajectory constraints specify the perturbation to the flow rate.

```
(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*)))
```

A simulation using these trajectory constraints shows that an overflow of the lake is possible if no intervening action is taken (figure 4). To guarantee that an overflow does not occur, an 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-conservation-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 between (`op* max`) constrained to be greater than or equal to 4) and the ordering with respect to the threshold. By restricting the simulation to behaviors that lead to an overflow condition, TeQSIM determines a lower bound for the temporal occurrence of actions leading to an overflow.

```
(event open)
(disc-change (event open)
             ((g1.opening (op* max)
                          :range (4 NIL))))
(before (qvalue stage (top NIL)) (event open))
(eventually (qvalue stage (top NIL)))
```

The simulation determines that an overflow is prevented if the gate is opened to at least 4 feet within 15.5 days (figure 5). Using the results from this simulation as an input, a third simulation derives an upper bound of 6 ft for the size of the opening given a restriction on the outflow rate expressed via the temporal logic expression `(always (value-<= colorado-dn 350))`.




A TeQSIM simulation of the lake model with the specified perturbation and no control action results in three behaviors. In the behavior shown above, `Stage` starts from its initial value (`S*`) and reaches the top of the dam (`Top`) before the end of the perturbation. The other two predicted behaviors describe the overflow occurring at the same time as the end of the perturbation and the situation where no overflow occurs.

Figure 4: Lake simulation with no action.

In this example, trajectory constraints are used to control an exogenous variable via discontinuous change expressions and the scope of the simulation is restricted via temporal logic constraints. Due to the simplicity of this example, only a small number of behaviors are filtered out via the temporal logic constraints. In an example such as the billiard problem presented previously, the number of behaviors filtered by the trajectory constraints can be quite large.

Both components of the TeQSIM algorithm, qualitative simulation and trajectory specification, are crucial when solving a problem of this nature. Qualitative simulation provides a discretization of trajectories essential for supporting a search mechanism. In addition, each qualitative behavior is refined via forward and backward propagation of quantitative information contained within the constraints. Trajectory constraints are used to drive the system by specifying the behavior of the exogenous variable (i.e. input flow rate) and by specifying performance requirements. This information cannot be represented within the QDE.

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, the specification of input trajectories is uncertain (*i.e.* ranges for times and values). Nevertheless with three simple simulations a reasonably useful result has been determined.

## TeQSIM Architecture and Theory

TeQSIM can be divided into two main components: the *preprocessor* modifies the QDE model and decomposes the trajectory specification into temporal logic and discontinuous change expressions; and the *simulation and model checking* component integrates temporal logic model checking into the simulation process by filtering and refining qualitative behaviors according to a set of temporal logic expressions and injects discontinuous changes into the simulation. Figure 6 provides an overview of the system architecture.

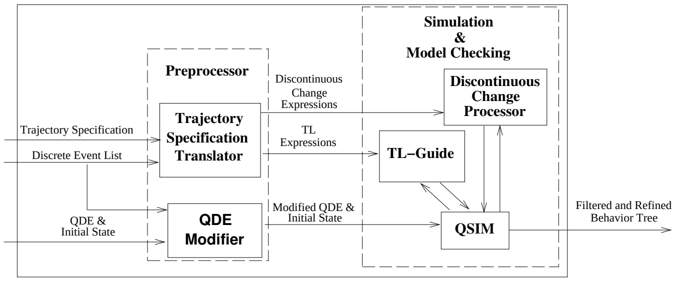
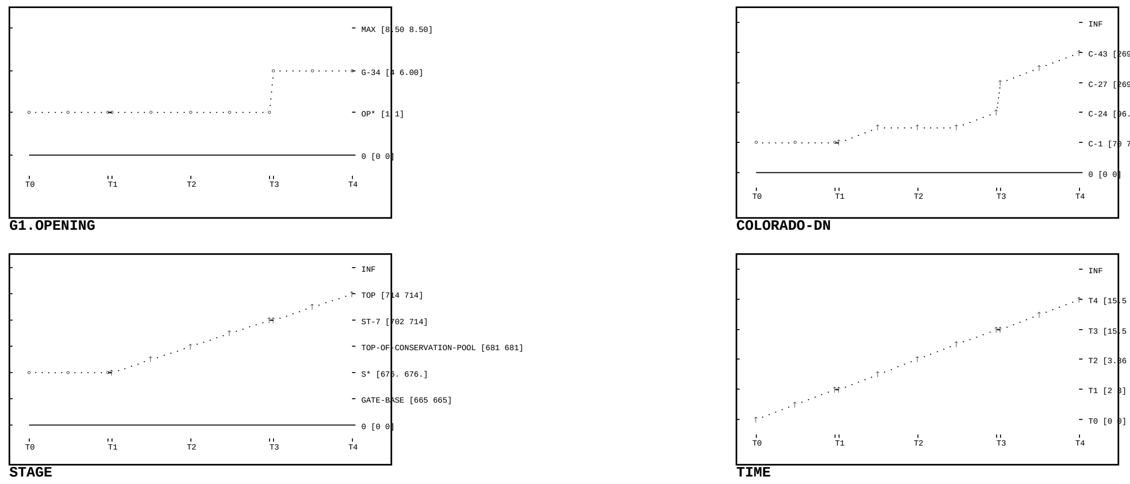



Figure 6: TeQSIM architecture.

The user provides trajectory constraints to TeQSIM in the form of a trajectory specification. The specification consists of an external event list and a set of extended temporal logic and discontinuous change expressions. An *external event* represents a distinguished time–point not predictable by the model. The external event list is a totally ordered sequence of named, quantitatively bound external events. Events are incorporated into the simulation by the addition of an auxiliary variable representing "real time" to the model with a landmark corresponding to each event. The addition of this variable causes QSIM to branch on different orderings between external events and internal qualitative events identified during the simulation. The occurrence of the external events is restricted by their quantitative bounds and the trajectory constraints specified by the modeler.

Temporal logic and discontinuous change expressions are extended within the trajectory specification by allowing direct references to events within the event list. These references are replaced by the appropriate formula containing a reference to the real–time variable and the landmark corresponding to the event. The addition of the real–time variable incorporates external events into the simulation in a seamless manner that does not require special handling during the simulation and model checking component of the algorithm.

This section provides a description of how tempo-




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 5: Lake simulation with opening actions leading to overflow.

ral logic and discontinuous change expressions are processed within TeQSIM. In addition, a summary of the formal framework used when describing the model checking algorithm is included. A more detailed treatment of syntax and semantics of the language and main theorems is given in (Brajnik and Clancy, 1996b). Proofs of these and other theorems along with additional lemmas and corollaries are included in (Brajnik and Clancy, 1996a).

## Guiding and refining behaviors using trajectory constraints

Model checking and behavior refinement is performed by the *Temporal Logic Guide* (TL–Guide). Each time QSIM extends a behavior by the addition of a new state, the behavior is passed to the TL–Guide. The behavior is filtered if there is sufficient information within the partially formed behavior to determine that all completions of the behavior fail to satisfy the set of TL expressions. If the behavior can potentially model the set of TL expressions, then it is refined by incorporating relevant quantitative information contained within the TL expressions. Otherwise the behavior is retained unchanged.

The following is an example of a state formula of the trajectory specification language:

`(qvalue` $v$ `(`*qmag qdir*`))` where $v$ is a QDE variable, *qmag* is a landmark or open interval defined by a pair of landmarks in the quantity space associated with $v$, 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 formulae are defined recursively as either a state formula or a combination of path formulae using temporal and boolean operators. A state formula is true of a path if it is true for the first state in the path. The path formula (`until` $p$ $q$), where both $p$ and $q$ are path formulae, 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 the behavior contains at least two states and $p$ holds in the path starting at the second state. We extended the temporal operators defined by Kuipers and Shults (Shults and Kuipers, 1996) to provide a more abstract language to simplify specification of assertions. These abbreviations are described in figure 7.

Temporal logic formulae 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 QSIM 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. Such objects are needed because the language provides a set of propositions which refer to quantitative ranges and whose truth value cannot always be determined. When ambiguity occurs for a formula, then the interpretation is required to provide necessary and sufficient refinement conditions on quantitative ranges of states to disambiguate the truth value of the formula. A *refinement condition* is an inequality between the partially known numeric value of a variable in a state and an extended real number, or a boolean combination of conditions. The trajectory specification language contains two potentially

| | | |
|---|---|---|
| (releases $p$ $q$) | $\equiv$ | (not (until (not $p$) (not $q$))) |
| (before $p$ $q$) | $\equiv$ | (not (until (not $p$) $q$)) |
| (eventually $p$) | $\equiv$ | (until true $p$) |
| (always $p$) | $\equiv$ | (releases false $p$) |
| (never $p$) | $\equiv$ | (always (not $p$)) |
| (starts $p$ $q$) | $\equiv$ | (releases $p$ (implies $p$ (always $q$))) |
| (follows $p$ $q$) | $\equiv$ | (releases $p$ (implies $p$ (strong-next (always $q$)))) |
| (occurs-at $p$ $q$) | $\equiv$ | (releases $p$ (implies $p$ $q$)) |
| (between $p$ $q$ $r$) | $\equiv$ | (releases $p$ (implies $p$ (strong-next (until $r$ $q$)))) |

The intuitive meaning for some of these forms is:

| | |
|---|---|
| `(releases` $p$ $q$`)` | $q$ is true up until and including the first state in which $p$ is true, if any. |
| `(starts` $p$ $q$`)` | $q$ holds from the first occurrence of $p$. |
| `(follows` $p$ $q$`)` | similar to *starts*, but $q$ should hold just after the first occurrence of $p$. |
| `(occurs-at` $p$ $q$`)` | $q$ occurs at the first occurrence of $p$. |
| `(between` $p$ $q$ $r$`)` | $r$ holds in the open time interval between the first occurrence of $p$ and the subsequent first of $q$. |

Figure 7: Temporal abbreviations.

ambiguous state formulae (in the following, $\mathcal{R}(v, s)$ denotes the range of possible numeric values for variable $v$ in state $s$, $v_s$ the unknown value of $v$ in $s$, and $n, n_i \in \Re \cup \{-\infty, +\infty\}$):

`(value-<=` $v$ $n$`)` is true iff $\forall x \in \mathcal{R}(v, s) : x \leq n$; it is false iff $\forall x \in \mathcal{R}(v, s) : n < x$; it is conditionally true otherwise. 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.* $v_s \leq n$).

`(value-in` $v$ $(n_1\ n_2)$`)` is true iff $\mathcal{R}(v, s) \subseteq [n_1, n_2]$; it is false iff $\mathcal{R}(v, s) \cap [n_1, n_2] = \emptyset$. It is conditionally true otherwise, and the refinement condition is that the greatest lower bound is equal to $n_1$ and the least upper bound is equal to $n_2$ (*i.e.* $n_1 \leq v_s \wedge v_s \leq n_2$).

Applying a refinement condition to a state yields a new, more refined state. For example, the formula `(value-<= X .3)` may generate 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 syntactic property, but rather it depends on state information. For example, `(value-<= X .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 formulae. The first one, called *models*, characterizes non–ambiguous true formulae while the second one, called *conditionally models* characterizes ambiguous formulae.

To simplify the analysis of the refinement process, the usage of ambiguous formulae must be restricted. The problem is that an arbitrary formula may yield several alternative refinement conditions. A disjunction of refinement conditions cannot be applied to states without requiring the introduction of a new behavior which is qualitatively identical to the original behavior in the tree. 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 in which $X_{s'}$ is restricted to be $\leq 0.5$ and a state where $Y_{s''} \leq 15$. The set of *admissible formulae* is a syntactic restriction which excludes formulae which may result in disjunctive conditions. This restriction complies with the general principle that all important distinctions in trajectories are made explicit in qualitative information. If such a principle is followed then the restriction to admissible formulae does not reduce the applicability of TeQSIM.

## Discontinuous Changes

The injection of discontinuous changes into the qualitative simulation process 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. The following expression is used to specify a discontinuous change:

`(disc-change` *preconds effects*`)` where *preconds* is a boolean combination of `qvalue` propositions and *effects* is a list of expressions of the form `(`*variable qmag* `[:range` *range*`])`.

This expression is translated into the temporal logic path formula

`(occurs-at` *preconds* `(strong-next` *effects'*`))` where *effects'* is a conjunction of formulae `(qvalue` *variable* `(`*qmag NIL*`))` and `(value-in` *variable range*`)` derived from *effects*. This expression is true for a behavior iff *effects'* is true for the state immediately following the first state in which *preconds* is true. This formula is added to the list of temporal logic formulae used to guide and refine behaviors.

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 effect 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 inserted as transition successors of $s$. Each discontinuous change expression can only be applied once within each behavior.

Qualitative reasoning uses continuity constraints to restrict the possible changes that can occur within a system. In order to predict the effects of discontinuous changes, however, continuity constraints have to be relaxed leading to a combinatorial explosion of possible outcomes. *Continuity relaxation* (Brajnik, 1995) is a technique that propagates the effects of a discontinuous change through the model by identifying variables that are necessarily continuous and variables that are potentially discontinuous. If only one variable within a non–differential constraint is not known to be continuous then it is inferred to be necessarily continuous. Such a technique assumes that state variables (*i.e.* those being integrals of model variables) and input variables, unless mentioned in the discontinuous change expression, are necessarily continuous.

Continuity relaxation is proven to be correct, but has not been proved to be complete. No counterexamples have been found, however, where continuity relaxation is too a weak method.

## Model checking

The temporal logic model checking algorithm is designed to evaluate a QSIM behavior with respect to a set of temporal logic formulae as the behavior is incrementally developed. This allows behaviors to be filtered and refined as early as possible during the simulation. Our algorithm is derived from the one described in (Kuipers and Shults, 1994); however, it has been modified to deal with conditionally true formulae and to cope with behaviors which are not closed, *i.e.* which are still to be extended by the simulator and it may be still undetermined whether or not they satisfy a given temporal logic formula. The algorithm, described in (Shults and Kuipers, 1996), computes the function $\tau\colon \textit{Formulae} \times \textit{Behaviors} \rightarrow \{\mathtt{T},\mathtt{F},\mathtt{U}\} \times \textit{Conditions}$, where *Conditions* 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 not true for any suffix of $b$, since $p$ may become true in the future.

A behavior $b$ is considered to be *sufficiently determined* with respect to a temporal logic formula $\varphi$ (written $b \rhd \varphi$) whenever there is enough information within the behavior to determine a single truth value for all completions of the behavior. If a behavior is not sufficiently determined for a formula, then `U` is returned by the algorithm[2]. The definition of *sufficiently determined* (omitted because of its length) 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 $\varphi$. 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.

When given a behavior $b$ and an admissible formula $\varphi$, TL–guide computes $\tau(\varphi, b) = (v, C)$. The behavior $b$ is refuted iff $v = \mathtt{F}$; it is retained unmodified iff $v \neq \mathtt{F}$ and $C = \mathtt{TRUE}$; and it is refined using $C$ iff $v \in \{\mathtt{T},\mathtt{U}\}$ and $C \neq \mathtt{TRUE}$.

The following theorem justifies 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 $\varphi$ then TL–guide:*

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

**Proof.** By induction on the formula $\varphi$.

# Problem Solving with Trajectory Information

The ability to specify trajectory information, discontinuous changes and external events provides a powerful extension to qualitative simulation broadening the range of relevant dynamical system theory problems. Trajectory specifications may refer to dependent or independent variables of a piecewise continuous dynamical system described by ordinary differential equations. The following classes of dynamical system theory problems are distinguished by the role that the constrained variables play within the model. Both classes can be addressed by incorporating trajectory information into a qualitative simulation.

**Simulation of non–autonomous systems:** Trajectory specifications are provided for the input variables describing their time–varying behavior and driving the simulation. The specification of discontinuous changes in the input variables enables the user to model external actions occurring at a faster time–scale than the simulation. This is useful when modeling

- situations where limited knowledge is available regarding the transient period during which the action occurs;
- situations where the transient is by itself uninteresting;

[2] Inferences within the model checking algorithm are limited to the information contained within the behavior. Thus, even though the constraint `(constant X)` is included within a model, the model checking algorithm will not be able to determine that the formula `(eventually (qvalue X (nil inc)))` is `F` for all completions of a non–closed behavior.

- a discrete event system (e.g. a digital controller) acting upon an otherwise continuous plant.

**Solution of boundary condition problems:** Trajectory constraints can be used to specify information about the behavior of dependent variables at various time points restricting the space of solutions of the differential equation. Available boundary condition information may include

- knowledge about intermediate or final states of the system, or
- knowledge about observed variables.

Trajectory information combined with qualitative simulation unifies these two classes allowing for the solution of problems that combine features of both classes. Table 1 describes some of the tasks that have been addressed using trajectory information.

## Related Work

While the incorporation of trajectory information into a qualitative simulation has been investigated in the literature, it has not been extensively explored. DeCoste (DeCoste, 1994) introduces *sufficient discriminatory envisionments* to determine whether a goal region is possible, impossible or inevitable from each state of the space. This is performed by generating the simplest state description that is sufficient for inferring these discriminations. Discrete actions are handled using the *action–augmented envisionment* method reviewed below. Though similar in spirit, our work is: (i) more general, because TeQSIM enables the user to address a wide category of problems, not limited to determining reachability of a state; (ii) semantically well–founded, based on temporal logic and (iii) formally proven to provide guaranteed results if applied under certain conditions. Washio and Kitamura (Washio and Kitamura, 1995) present a technique that uses temporal logic to perform a *history oriented envisionment* to filter predictions. TeQSIM, within a more rigorously formalized framework, provides a more expressive language, it refines behaviors as opposed to just filtering them, and it incorporates discontinuous changes into behaviors.

The integration of temporal logic model checking and qualitative simulation was initially investigated by Kuipers and Shults (Kuipers and Shults, 1994). They use a branching–time temporal logic to prove properties about continuous systems by testing the entire behavioral description against a temporal logic expression. The appropriate truth value is returned depending upon whether or not the description satisfies the expression. Our work focuses on constraining the simulation as opposed to testing a simulation after it is completed.

Forbus (Forbus, 1989) explicitly introduces the concept of discrete action, with pre and post–conditions in the *action–augmented envisionment*. The purely qualitative total envisionment that is generated includes all possible instantiation of known actions. Forbus allows only instantaneous actions and adopts heuristic criteria (based on minimality of the change in the description) to handle discontinuities. No provision is made to handle quantitative information, nor to prove correctness of the discontinuity handling mechanism. Discontinuities have been investigated also by Nishida and Doshita (Nishida and Doshita, 1987), who describe two methods for handling discontinuities caused by external agents or generated autonomously within a system (*e.g.* change in operating regime): (i) approximating a discontinuous change by a quick continuous change and (ii) introducing "mythical states" to describe how a system is supposed to evolve during a discontinuous change. The former requires a complex machinery to compute the limit of the quick change, whereas the latter is based on heuristic criteria for selecting appropriate states. Both methods are interesting, but their effectiveness and formal properties are difficult to ascertain. Iwasaki and colleagues (Iwasaki *et al.*, 1995) discuss the semantics of discontinuous changes that is more appropriate when dealing with hybrid systems. Their work leads to the adoption of a complex non–standard analysis semantics for reals and the development of a mechanism similar to continuity relaxation but requiring user–supplied frame axioms.

The trajectory specification language can be compared to other formalisms for specifying temporal constraints. Our language is similar in expressiveness to both Allen's *interval algebra* (Allen, 1984) and Dechter, Meiri and Pearl's *temporal constraint networks* (Dechter *et al.*, 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.

## Discussion and Future Work

TeQSIM supports a general methodology for incorporating arbitrary trajectory information into the qualitative simulation process. We are validating this methodology in several control problems, including some in the domain of risk assessment in water supply control, where models of lakes, rivers, and dams are simulated with the purpose of evaluating potential actions (e.g. changing the power requested to a turbine, or the opening of a floodgate).

The following extensions are being investigated to increase the expressiveness of the trajectory specification language.

**Limited first order expressiveness** – The temporal logic used is limited to propositional expressions and is unable to quantify over attributes of states. Certain trajectory constraints require the ability to refer to values in other states within the behavior.

| | | |
|---|---|---|
| **Goal oriented simulation** | Input:<br>Output:<br>Relation: | A description of the desired (or undesired) behavior.<br>Behaviors that are "consistent" with the goal specification.<br>By specifying the desired behavior a modeler can gain an understanding of these behaviors. Specifying undesired behaviors allows the modeler to determine *if* these behaviors exist and to decide how they can be avoided when they do exist. |
| **Analysis of discrete actions** | Input:<br>Output:<br>Relation: | A list of discontinuous changes resulting from actions, and trajectory constraints specifying when the actions can be performed.<br>The behaviors resulting from the actions.<br>Multiple simulations can be performed to evaluate alternative courses of actions. |
| **Analysis of control responses** | Input:<br>Output:<br>Relation: | Trajectory constraints specifying the desired closed–loop behavior of the system and an environmental perturbation.<br>A description of a controller response to the perturbations that is necessary for achieving the closed–loop behavior specified as input.<br>All potential control responses satisfying the closed–loop behavior are included within the behavioral description. If there is no consistent behavior, then the desired closed–loop behavior cannot be achieved by any controller in response to the perturbation. |
| **Parameter identification** | Input:<br>Output:<br>Relation: | A model containing a partial description of a controller and trajectory constraints describing the desired behavior of the controller and a perturbation.<br>Behaviors containing quantitative bounds on the unknown controller parameters.<br>The range of potential values for unknown controller parameters that are necessary to achieve the desired behavior. |
| **Analysis of feedback control laws** | Input:<br>Output:<br>Relation: | Specification of a control law using trajectory constraints.<br>A description of the resulting closed–loop behaviors.<br>The control law is specified by relating the value of the monitored variable with the required value, or trend, of the control variable. |

Table 1: Tasks to which TeQSIM has been applied.

For example, the description of a decreasing oscillation requires the ability to compare the magnitude of a variable across states. A limited form of first order temporal logic may provide a sufficiently expressive language while still giving satisfactory performance with respect to complexity.

**Metric temporal logic** – Due to the introduction of landmarks during the simulation process, TeQSIM behaviors are potentially infinite structures. Getting a definite answer for formulae such as (`eventually` $p$) is not always possible when potentially infinite behaviors are encountered since it is always possible for $p$ to occur in the future. Metric temporal logic (Alur and Henzinger, 1993) allows the definition of a horizon for a temporal logic expression. This would allow statements such as "within 50 seconds the tanks level reaches 70 inches." These statements are only expressible within our logic using an externally defined event. Extending the logic would allow the modeler more flexibility to express relevant constraints.

**Discontinuous change specification** – In the current version of TeQSIM we provide very simple means for representing and reasoning about discontinuous changes. While sufficient for certain kinds of problems (*e.g.* driving the simulation by controlling the behavior of an exogenous variable), other problems cannot be addressed (*e.g.* identifying repetitive actions performed by a controller). Two possible extensions are being considered:

- supporting more complex relationships between the precondition and the effect using additional temporal logic operators. For example, the modeler may want to identify a sequence of states over which the action can be performed or express information about the *possibility* of a discontinuous change occurring.
- allowing preconditions to be specified using an arbitrary temporal logic expression. This would extend the range of addressable feed–forward control problems.

**Partially ordered events** – External events are currently restricted to a single, totally ordered set. Relaxing this restriction to allow for a partially ordered set of events would significantly increase the expressiveness of the language. This extension in conjunction with an increase in the expressiveness of the discontinuous change expressions would provide a powerful mechanism for simulating hybrid discrete–continuous systems and for planning control actions.

Finally, the current algorithm for incremental model checking is inefficient if compared to the *on–the–fly* model checker algorithm developed by Bhat and colleagues (Bhat *et al.*, 1995).[3] We are planning to incorporate it within TeQSIM.

## Conclusions

Qualitative simulation and temporal logic are two alternative formalisms for reasoning about change across time. TeQSIM integrates these two paradigms. Trajectory information is specified using temporal logic, external events, and discontinuous changes and integrated into the qualitative simulation process. Trajectory constraints can be used to qualitatively restrict

[3] In all the examples we have run the practical time–complexity of a TeQSIM simulation is dominated by quantitative inferences as opposed to model checking. The total run time of the simulations presented in the example section is around 20 seconds on a Sun–20 workstation running Lucid Lisp.

the simulation to a region of the state space and to refine individual behaviors quantitatively. Behaviors that do not model the set of temporal logic expressions are filtered during simulation. Numeric information specified within the temporal logic expressions can be integrated into the simulation to provide a more precise numerical description for the behaviors which model these expressions. Semantically well–defined discontinuities are integrated into trajectories and a sound and general method is used to relax continuity constraints and automatically determine variables affected by the change.

The correctness of the TL–guide algorithm and of the discontinuous change processor along with the correctness of QSIM guarantee that all possible trajectories of the modeled system compatible with the QDE, 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.

## Acknowledgments

We would like to thank Benjamin Shults for letting us use part of his program to implement TeQSIM and and the Qualitative Reasoning Group for many fruitful discussions.

QSIM and TeQSIM are available for research purposes via anonymous `ftp` at `ftp.cs.utexas.edu` in the directory `/pub/qsim`. These and other results of the Qualitative Reasoning Group are accessible by World-Wide Web via `http://www.cs.utexas.edu/users/qr`.

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.

## References

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

R. Alur and T. Henzinger. Real–time logics: complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.

G. Bhat, R. Cleaveland, and O. Grumberg. Efficient on–the–fly model checking for CTL*. In *Proc. of Conference on Logic in Computer Science (LICS–95)*, 1995.

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

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

G. Brajnik. Introducing boundary conditions in semi–quantitative simulation. In *Ninth International Workshop on Qualitative Reasoning*, pages 22–31, Amsterdam, May 1995.

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

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

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

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

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

Y. Iwasaki, A. Farquhar, V. Saraswat, D. Bobrow, and V. Gupta. Modeling time in hybrid systems: how fast is "instantaneous"? In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1773–1780, Montréal, Canada, 1995. Morgan Kaufmann Publishers, Inc.

B.J. Kuipers and B. Shults. Reasoning in logic about continuous change. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Principles of Knowledge Representation and Reasoning*, San Mateo, CA, 1994. Fourth International Conference (KR–94), Morgan Kaufmann Publishers, Inc.

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

T. Nishida and S. Doshita. Reasoning about discontinuous change. In *AAAI87*, pages 643–648, 1987.

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

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