In *Proc. of the Third International Workshop on Temporal Representation and Reasoning (TIME'96)*
May 19-20, 1996, Key West, Florida.
IEEE Computer Society Press, pp. 144-151.

Guiding and Refining Simulation using Temporal Logic

Giorgio Brajnik*

Dip. di Matematica e Informatica
Università di Udine
33100 Udine, ITALY

Daniel J. Clancy

Department of Computer Sciences
University of Texas at Austin
Austin, TEXAS 78712

Abstract

We illustrate TeQSIM, a qualitative simulator for continuous dynamical systems. It combines the expressive power of qualitative differential equations with temporal logic by interleaving simulation with model checking to constrain and refine the resulting predicted behaviors. Temporal logic expressions are used to specify constraints that restrict the simulation to a region of the state space and to specify trajectories for input variables. A propositional linear–time temporal logic is adopted, which is extended to a three valued logic that allows a formula to be conditionally entailed when quantitative information specified in the formula can be applied to a behavior to refine it. We present a formalization of the logic with theoretical results concerning the adopted model checking algorithm (correctness and completeness). We show also an example of the simulation of a non–autonomous dynamical system and illustrate possible application tasks, ranging from simulation to monitoring and control of continuous dynamical systems, where TeQSIM can be applied.

1 Introduction

Reasoning about change across time is a common problem within artificial intelligence and computer science in general. For systems with discrete state spaces, temporal logic (TL) provides a broad class of formalisms that have been used for such tasks as reasoning about the effect of actions, specifying and verifying correctness of reactive computer programs and synthesizing or analyzing discrete event systems [13, 6, 2, 8]. Qualitative reasoning [10] has been used to reason about continuous change within the field of dynamical systems in the presence of incomplete knowledge. The expressiveness of the models used within qualitative simulation, however, is often limited to structural equations constraining the potential values for related variables. The modeler is unable to express behavioral information about the trajectory of a variable or relationships between the trajectories of interconnected variables. Such an information allows the modeler to restrict the simulation to a region of the state space and to specify trajectories for input variables.

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

The Temporally Constrained QSIM (TeQSIM, pronounced *tek'sim*) algorithm combines the expressive power of these two paradigms by interleaving temporal logic model checking with the qualitative simulation process. Temporal logic is used to specify qualitative and quantitative trajectory information that is incorporated into the simulation to constrain and refine the resulting behaviors.

Qualitative simulation constructs a set of possible behaviors consistent with a model of a dynamical system represented by a qualitative differential equation (QDE). The QSIM algorithm [10] represents the behavior of a dynamical system by an incrementally generated tree of qualitative states. Each state describes the system at either a time–point or over a time–interval between two points by a tuple of qualitative values for the variables specified within the QDE. Each qualitative value is described by a magnitude and a direction of change: the direction of change represents the sign of the variable's time derivative while the magnitude is defined upon a totally ordered set of distinctive landmark values and is either a landmark value or an interval between two landmark values. The simulator uses the constraints specified within the QDE along with continuity to derive a branching–time behavioral description. Each path within the tree represents a potential behavior of the system: branches result from inherent ambiguity within the qualitative description. Semi–quantitative simulation incorporates quantitative information into the qualitative simulation in the form of numeric ranges for landmark values and bounding envelopes for functions.

TeQSIM interleaves temporal logic model checking with the qualitative simulation process to obtain two major benefits. *Behavior filtering* tests each partial behavior against the set of temporal logic expressions representing trajectory constraints as the set of behaviors is incrementally generated. A behavior is eliminated from the simulation when it can be shown that all of its possible completions fail to model the set of temporal logic expressions. Thus, the space of the behavioral description is restricted to include only behaviors that can satisfy the temporal logic expressions. *Behavior refinement* integrates numeric information contained within the temporal logic expressions into the qualitative simulation to provide a more precise numerical description. This process restricts an individual behavior to include only those real valued inter-

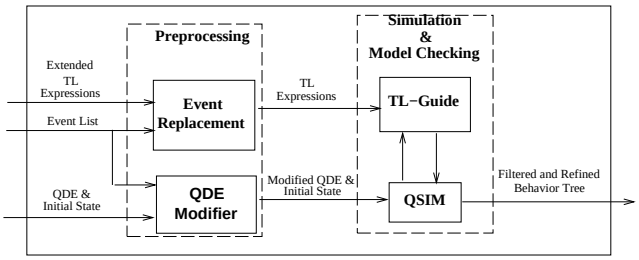

Figure 1: TeQSIM architecture.

pretations which model the set of temporal logic expressions and therefore improves the precision of the prediction.

The integration of temporal logic model checking and qualitative simulation was initially investigated by Kuipers and Shults [11]. 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 models the expression. Our work focuses on constraining the simulation as opposed to testing a simulation after it is completed.

The next section provides an overview of the TeQSIM algorithm. Section 3 provides an example along with a discussion of some of the applications of this technique while section 4 describes the formal syntax and semantics of our temporal logic. Soundness and completeness theorems are presented for the TL–guide algorithm. Finally, section 5 provides a discussion of some of the extensions that will be investigated in the future.

2 TeQSIM Overview

TeQSIM has been designed to provide the user with a mechanism for specifying trajectory constraints to guide and refine the qualitative simulation process. By *trajectory* of a set of variables over a time interval $(a, b) \subseteq \Re^*$ we mean a mapping from (a, b) to variable values ($\subseteq \Re^*$). Trajectory constraints on a set of variables restrict the possible trajectories for those variables.

Figure 1 provides an overview of the system architecture. In general, the algorithm can be divided into two main components: a preprocessing stage that combines the trajectory information provided by the modeler into the qualitative model and generates the appropriate TL expressions; and a simulation and model checking stage that integrates model checking into the simulation process by filtering and refining qualitative behaviors according to a set of temporal logic expressions.

Trajectory information is specified in the form of an *event list* and a set of *extended temporal logic* statements. An *event* is a time–point distinguished within the simulation. The event list is a sequence of named, quantitatively bounded events not represented within the QDE that are incorporated into the simulation. An extended temporal logic statement is simply a temporal logic formula which may include direct references to events occurring within the event list. These references are replaced by the appropriate formulae. The simulation provides a complete temporal ordering between these externally defined events and other events defined within the model.

Model checking and behavior refinement is performed by TL–guide. Each time QSIM extends a behavior by the addition of a new state, the behavior is passed to 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 model the set of TL expressions. If the behavior can potentially model the set of TL expressions, then it is refined by the incorporation of any relevant quantitative information contained within the TL expressions. Otherwise the behavior is retained unchanged.

3 Problem solving with TeQSIM

Incorporating temporal logic model checking into the qualitative simulation process allows the modeler to control the simulation by restricting the behavior of input and dependent variables. Trajectory constraints can be used for a variety of tasks, including simulation of non–autonomous systems, incorporating observations into simulation, analysis of continuous control laws and performing goal oriented simulation. The TeQSIM algorithm has been tested on a range of examples demonstrating each of the tasks above. The following example demonstrates the use of TeQSIM to simulate a non–autonomous system incorporating information obtained via observation. The system being simulated is a very simple one, which generalizes to the real–world problem of water supply control in the domain of lakes, rivers and dams [7]. Consider an open tank, with an uncontrolled inflow u, a regulated outflow y, valve opening v and amount of liquid A in the tank. The system is modeled by the differential equation $\dot{A} = u - y; y = f(A, v)$ where $f(A, v)$ is an unknown monotonically increasing function on both arguments, numerically bounded. This model can be straightforwardly specified using the QSIM QDE language.

Figure 2 shows the trajectory constraints used by TeQSIM for simulating the regulated tank. Part (a) of the figure shows a totally ordered event list that provides quantitative temporal bounds for external events corresponding to two opening actions on the outflow valve. Four events are defined corresponding to the beginning and the end of each of these actions. Part (b) contains the trajectory constraints that specify the behavior of the outflow valve along with constraints on inflow and level (up to the end of the first opening action; the second action, not shown, is similarly specified):

i. the variable `Inflow` is constant and its value is within the range [200, 220] cm^3/s,

ii. the valve opening is constant at 0.5 until the beginning of the first opening action,

iii. between events `b-open1` and `e-open1` (*i.e.* the duration of the first opening action) the valve

```
(event b-open1 :time (30 30))   ; sec
(event e-open1 :time (35 36))
(event b-open2 :time (150 150))
(event e-open2 :time (153 155))
```

(a) External event declaration.

```
i)   (always (and (qvalue InFlow (nil std))
                  (value-in InFlow (200 220))))
ii)  (until (and (value-in Valve (0.5 0.5))
                 (qvalue Valve (nil std)))
            (event b-open1))
iii) (between (event b-open1) (event e-open1)
              (and (qvalue Valve (nil inc))
                   (qvalue Level (nil inc))))
iv)  (occurs-at (event e-open1)
                (value-in Valve (0.65 0.7)))
```

(b) Trajectory constraints (first opening action only).

Figure 2: Input to TeQSIM.

opening is increasing and `Level` is observed to increase, and

iv. valve reaches a value in [0.65, 0.7] at the end of the first opening action.

Figure 3 shows the result of the simulation. TeQSIM yields a single behavior where `Level` increases and reaches a new equilibrium value, well below the `High` threshold. Correctness properties of TeQSIM guarantee that this is the only possible outcome of any real plant that is validly described by the QDE model, the initial state and the trajectory constraints.

This example is very simple but it shows one possible use of trajectory constraints to extend the scope of qualitative simulators (like QSIM, that are limited to simulation of initial value problems only). A brief description of other tasks to which TeQSIM can be applied along with a discussion of how each task can be demonstrated on the model described above is contained in figure 4. Additional examples are contained in [5].

4 Guiding and refining simulation

TeQSIM guides and refines the simulation based upon a specification formulated using a variation of propositional linear time logic (PLTL). PLTL combines state formulae, that express information about an individual state, with temporal operators such as *until*, *always*, and *eventually* to extend these state formulae across time and represent change. We have extended PLTL by using a three valued logic that allows an expression to be *conditionally entailed* when quantitative information contained within the expression can be applied to a behavior to refine the description. A *refinement condition* specifies numerical bounds extracted from the TL expressions. Application of these conditions to the behavior eliminates the region of the state space that extended beyond the quantitative information specified in the TL expression.

In addition, the TL–guide 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 TL expression.

4.1 TL specification language

This section provides a formal description of the syntax and semantics for the TL language. The syntax and semantics are derived from work done by Emerson [6] and Kuipers and Shults [12]. A discussion of the TL–guide algorithm along with soundness and completeness theorems are presented in the following subsections. Proofs of these theorems along with additional lemmas and corollaries can be found in [5].

Syntax. The syntax and semantics for the state formulae are described in figure 5(a).

Path formulae are derived by applying the temporal operators `until` and `strong-next` along with boolean operators to state formulae. Path formulae $\mathcal{P}$ are formally defined as $\mathcal{P} ::= \mathcal{S} | (\texttt{and}\ \mathcal{P}\ \mathcal{P}) | (\texttt{not}\ \mathcal{P}) | (\texttt{strong-next}\ \mathcal{P}) | (\texttt{until}\ \mathcal{P}\ \mathcal{P})$, where $\mathcal{S}$ is the set of state formulae.

Informally, (`until` p q) is true for a path if p holds in all states preceding the first one where q holds, while (`strong-next` p) is true for a path if p holds in the second state of the path which must exist. Figure 5(b) gives a set of useful abbreviations.

We require that formulae are in *positive normal form*, *i.e.* (i) `until`, `releases` and `strong-next` are the only temporal operators in the formula, (ii) for every `not` in the formula, its scope is an atomic proposition, and (iii) such a scope does not include any proposition constructed using `value-<=`, `value->=` or `value-in`.
The first two requirements do not restrict the expressiveness of the language since the abbreviations shown above can be used to transform a formula to satisfy these conditions. The latter requirement is due to the specific representation of numeric information in QSIM, which does not allow open numeric intervals.

Semantics. Temporal logic formulae are given meaning with respect to the interpretation structures defined below. These structures are extended from their typical definition (*e.g.* [6]) in order to accommodate the refinement process.

Path formulae are interpreted against an *interpretation structure* $M = <S, \sigma, \Gamma, \mathcal{I}, \mathcal{C}, \mathcal{M}>$ where:

- S is a set of states;
- $\sigma: S \rightarrow S$ is a partial function, mapping states to their successors (we are defining a linear–time logic, hence each state has at most one successor);
- $\mathcal{I}: \mathcal{S} \times S \rightarrow \{t, f, \Gamma\}$ is an assignment of truth values to propositions and states (Γ denotes the "unknown" truth value);

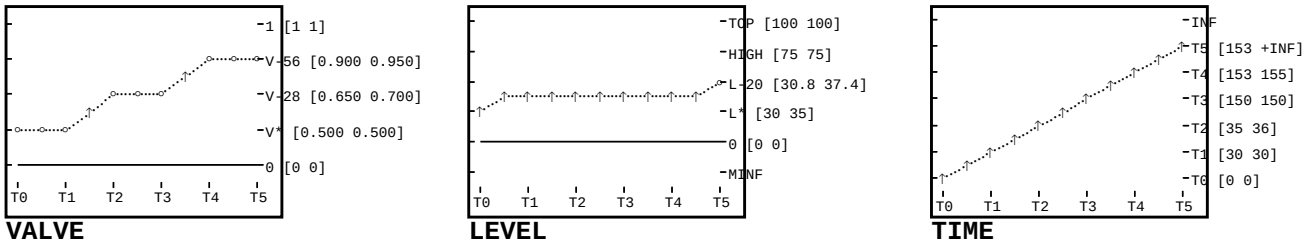

Figure 3: Output of TeQSIM.

- Γ is a set of *refinement conditions*. Γ is closed with respect to the standard boolean operators $\{\wedge, \neg\}$ and contains the distinguished item `TRUE`;
- $\mathcal{C}$:$\mathcal{S} \times S \rightarrow \Gamma$ is a function (*condition generator*) that maps state formulae and states into refinement conditions that disambiguate a formula truth value; we require that $\mathcal{C}(\varphi, s) =$ `TRUE` iff $\mathcal{I}(\varphi, s) = t$ and $\mathcal{C}(\varphi, s)$ to be defined when $\mathcal{I}(\varphi, s) = \Gamma$.
- $\mathcal{M}$:$\Gamma \times S \rightarrow S$ is a function (*state modifier*) that maps a condition and a state into a refined state. For any state s, $\mathcal{M}($`TRUE`$, s) = s$ and $\mathcal{M}(\neg$`TRUE`$, s) = -$.

We require that if φ is an atomic proposition then refinement conditions are necessary and sufficient for resolving the ambiguity, *i.e.* if $C = \mathcal{C}(\varphi, s)$ then $\mathcal{I}(\varphi, \mathcal{M}(C, s)) = t$ and $\mathcal{I}(\varphi, \mathcal{M}(\neg C, s)) = f$ (unless $C =$ `TRUE`, in which case $\mathcal{M}(\neg C, s) = -$).
As customary, a *path* x is a sequence of states $x = <s_0, s_1, \ldots>$ such that for any pair of consecutive states (s_i, s_{i+1}) we have that $\sigma(s_i) = s_{i+1}$. The length of a path is denoted by $|x|$, which for infinite paths is ∞. For all non negative integers $i < |x|$, x^i denotes the sub–path $<s_i, \ldots>$ and $x(i)$ denotes s_i. A *full–path extension* of a finite path x, denoted with $\widehat{x}$, is an infinite path formed by concatenating x with an infinite sequence of states. Finally, $\mathcal{M}$ is naturally extended to paths in order to refine paths.

In the specific case of QSIM, $\mathcal{I}$ may give Γ only for propositions including `value-<=`, `value->=` and `value-in` (as illustrated in fig. 5(a)). 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 condition that the QDE variable X in state s has to be less than 5 is written "$X_s < 5$".

Notice that ambiguity is not purely a syntactic property, but it depends on state information. For example, `(value-<= X .3)` will be (unconditionally) true on a state s where $\mathcal{R}(X, s) = [0, 0.25]$, but only conditionally true on s' where $\mathcal{R}(X, s') = [0, 1.0]$. Because of ambiguity, to define the semantics of formulae we need to introduce two entailment relations. The first one, called *models* ($\models$), will be used to characterize non–ambiguous true formulae; the second one, called *conditionally–models* ($\models^?$) will characterize formulae that are ambiguous. We will say that an interpretation M and a state s *falsify* ($\not\models$) a formula φ when neither $s \models \varphi$ nor $s \models^? \varphi$ hold. Figure 5(c,d) gives the semantics of the language.

To simplify the analysis of the refinement process, the usage of ambiguous formulae must be restricted. The problem is that an arbitrary ambiguous formula may yield several alternative refinement conditions. A disjunction of refinement conditions cannot be applied to states without requiring a change in the successor function σ and the introduction of a new behavior which is qualitatively identical to the original behavior in the tree. Two different types of disjunction can result from certain ambiguous formula. A *state disjunction* results from a disjunction of ambiguous state formulae. For example, when interpreted against a particular state `(or (value-<= X 0.5) (value->= Y 15))` may yield the condition $(X_s \leq 0.5 \vee Y_s \geq 15)$. When applying such a condition to a state, $\mathcal{M}(C, s)$ yields two states – s' in which $X_{s'}$ is restricted to be ≤ 0.5 and s'' where $Y_{s''} \geq 15$. A *path disjunction*, on the other hand, occurs when an ambiguous formula is included in a path formula in such a manner that a sub–formula can be conditionally true for more than one sub–path. For example, in the path formula `(until p (value-<= X 0.5))` a disjunction occurs across sub–paths regarding when `(value-<= X 0.5)` should be conditionally true.

The following definitions restrict the syntax to formulae that are well–behaved. A *potentially ambiguous formula* is any TL formula that (i) is an atomic proposition constructed using one of the following operators `value-<=`, `value->=` or `value-in`, or (ii) is a path formula which contains a potentially ambiguous sub–formula. *Admissible formulae* are those formulae φ that satisfy the following conditions:

1. φ is in positive normal form, and
2. if $\varphi \equiv$ (`until` p q) then q is not potentially ambiguous, and
3. if $\varphi \equiv$ (`releases` p q) then p is not potentially ambiguous, and
4. if $\varphi \equiv$ (`or` p q) then at most one of p and q is potentially ambiguous.

It can be proved that for all admissible formulae φ

Continuous feedback control	A control law is expressed in terms of a set of formulae relating the value of the monitored variable (say `Level`) with the required value, or trend, of the control variable (say `Valve`). The resulting closed–loop behaviors can then be analyzed with respect to the controller's goal.
	The following trajectory constraint provides a partial specification of a control law to avoid overflowing the tank. It states that the valve opening must be increasing whenever the magnitude of `Level` is greater than `high` and the valve hasn't yet reached its maximum opening `max`.
	
	`(always (between (qvalue Level ((high nil) nil)) (qvalue Level (high dec))`
	`(implies (qvalue Valve ((min max) nil))`
	`(qvalue Valve (nil inc)))))`

Continuous feed–forward control	The control law is expressed in terms of a set of formulae relating a predicted value of the monitored variable to the current value or trend of the control variable.
	The following trajectory constraint specifies that if the tank can potentially overflow then the valve opening should be increased until it reaches its maximum value or `level` becomes smaller than `high`.
	
	`(always (implies (eventually (qvalue Level (top nil)))`
	`(until (qvalue Valve (nil inc))`
	`(or (qvalue Level (high dec)) (qvalue Valve (max nil))))))`

Goal Oriented Simulation	The statements reported below can be used to check whether the tank will overflow within a specified time frame. Since TeQSIM is sound, if no behaviors are produced then the modeled system can not violate these constraints (assuming that the QDE model is valid).
	The following trajectory constraint limits the simulation to behaviors in which the tank `Level` reaches `high` within 150 seconds.
	
	`(and (event horizon :time 150)`
	`(before (qvalue Level (high nil)) (event horizon)))`

Figure 4: Applying TeQSIM to other tasks using the regulated tank model.

and for any interpretation M and path x, if $x \not\models^2 \varphi$ then any necessary and sufficient condition C for refining x into a model for φ (*i.e.* $\mathcal{M}(C,x) \models \varphi$ and $\mathcal{M}(\neg C,x) \not\models \varphi$) is either a single condition or a conjunction of conditions.

Even though the restriction to admissible formulae reduces expressiveness, such a restriction does not hinder the practical applicability of TeQSIM. As long as important distinctions are qualitatively represented (using landmarks or events), most trajectory constraints can be cast into admissible formulae. For example, the constraint that "until level goes above 50 the input flow rate has to be below 200" could be expressed with the following non–admissible formula `(until (value-<= InFlow 200) (value->= Level 50))` where the two distinctions (200 and 50) do not correspond to qualitative landmarks. By adding a landmark to the quantity space of `Level` corresponding to the value 50, the formula can be rewritten in an admissible form (*i.e.* `(until (value-<= InFlow 200) (qvalue Level (lm-50 nil))))`.

QSIM computes in finite time a set of behaviors, each representing a class of trajectories of the system being simulated. Although a QSIM behavior is a finite structure, it may represent infinite trajectories of the simulated system. In fact, quiescent states are finite descriptions of fixed–point trajectories[1]. Formally, a *QSIM behavior* b is a finite sequence of non repeating states $<s_0, \ldots, s_n>$ such that $\forall i, 0 \leq i < n$: (s_i, s_{i+1}) belongs to the QSIM relations `successor` or `transition`. A behavior b is *closed* iff QSIM detected that s_n is a quiescent state or that s_n is a transition state that has no possible successor.

4.2 Model checking

The model checking algorithm is designed to evaluate a behavior with respect to a set of admissible 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 [11]; however, it has been modified to deal with conditionally true formulae and to cope with behaviors which are not closed. A detailed discussion of the algorithm is provided in [5].

The algorithm computes the function $\tau{:}\mathcal{P} \times \mathit{Behaviors} \rightarrow \{\mathtt{T},\mathtt{F},\mathtt{U}\} \times \mathit{Conditions}$. 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, for all interpretations M, will *not* be sufficiently determined with respect to the formula $\varphi \equiv$ `(eventually` p`)` if p is not true for any suffix of b, since p may become true in the future. A behavior is considered to be *sufficiently determined* with respect to a formula whenever there is enough information within the behavior to determine

[1] QSIM may identify cyclic behaviors as well and represent them through cycles in a directed graph. The use of quantitative information, however, only makes sense if time is not cyclic; furthermore the refinement of behaviors requires that they do not share sub–behaviors. For these reasons cycle detection is disabled within TeQSIM.

(a) Syntax and semantics of state formulae.

In the definitions of the most important state formulae given below, v denotes a QSIM variable, $\mathcal{R}(v,s)$ the range of potential values for v in state s, and v_s the unknown value of v in s. n, n_i denote extended real numbers.

`(qvalue` v `(`$qmag$ $qdir$`))` where $qmag$ 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 exactly when the qualitative value of v in the state s matches the description `(`$qmag$ $qdir$`)`.

`(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 unknown otherwise. In such a case the refinement condition is that the least upper bound of the possible real values of v is equal to n (*i.e.* $v_s \leq n$). `(value->=` v n`)` is similar.

`(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 unknown 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.* $n_1 \leq v_s \wedge v_s \leq n_2$).

Non–atomic propositions are defined using standard boolean operators (`and`, `not`); standard propositional abbreviations are also allowed (`true`, `false`, `or`, `implies`, `iff`).

(b) Path formulae abbreviations.

`(or` p q`)`	$\equiv$	`(not (and (not` p`) (not` q`)))`
`(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.

`(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 is true 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.

(c) Entailment relations for state formulae.

(a ranges over atomic propositions, p and q over $\mathcal{S}$):

$s \models a$	iff	$\mathcal{I}(a,s) = t$
$s \not\models a$	iff	$\mathcal{I}(a,s) = ?$
$s \models$ (`and` p q)	iff	$s \models p$ and $s \models q$
$s \not\models$ (`and` p q)	iff	$s \not\models p$ and $s \models q$, or $s \models p$ and $s \not\models q$, or $s \not\models p$ and $s \not\models q$
$s \models$ (`not` p)	iff	$s \not\models p$
$s \not\models$ (`not` p)	iff	$s \not\models p$

(d) Entailment relations for path formulae.

($p \in \mathcal{S}$ and $\varphi, \psi \in \mathcal{P}$):

$x \models p$	iff	$x(0) \models p$
$x \not\models p$	iff	$x(0) \not\models p$
$x \models$ (`strong-next` φ)	iff	$\lvert x\rvert > 1$ and $x^1 \models \varphi$
$x \not\models$ (`strong-next` φ)	iff	$\lvert x\rvert > 1$ and $x^1 \not\models \varphi$
$x \models$ (`until` φ ψ)	iff	$\exists i \geq 0 : x^i \models \psi$ and $\forall j < i : x^j \models \varphi$
$x \not\models$ (`until` φ ψ)	iff	$\exists i \geq 0 : (x^i \models \psi$ or $x^i \not\models \psi)$ and $\forall j < i : (x^j \models \varphi$ or $x^j \not\models \varphi)$ and $\not\models$ occurs at least once

The semantics of (`and` φ ψ) and (`not` φ) is similar to the propositional case.

Figure 5: Syntax and semantics of the language.

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 τ and the behavior is not filtered out by TL–guide. Notice that indeterminacy is a property independent from ambiguity: the former is related to incomplete paths, while the latter deals with ambiguous information present in states of a path.

The following recursive definition characterizes determinacy. Given an interpretation M, a path x is *sufficiently determined* for a positive normal formula φ (written $x \rhd \varphi$) iff one of the following conditions is met:

1. x corresponds to a closed behavior or φ is a proposition or $x \models \varphi$ or $x \not\models \varphi$
2. $\varphi \equiv$ (`strong-next` p) and $|x| > 1$ and $x^1 \rhd p$
3. $\varphi \equiv$ (`until` p q) and $\exists i : i < |x|$ and $x^i \not\models p$ and $x^i \rhd p$ and $\forall j \leq i : x^j \not\models q$ and $x^j \rhd q$
4. $\varphi \equiv$ (`releases` p q) and $\exists i : i < |x|$ and $x^i \not\models q$ and $x^i \rhd q$ and $\forall j \leq i : x^j \not\models p$ and $x^j \rhd p$
5. $\varphi \equiv$ (`and` p_1 p_2) and $\exists i : x \not\models p_i$ and $x \rhd p_i$
6. $\varphi \equiv$ (`or` p_1 p_2) and $\forall i : x \not\models p_i$ and $x \rhd p_i$

We will write $x \not\rhd \varphi$ to signify that x is not sufficiently determined for φ.

We are now ready to state the main correctness and completeness theorem on model checking.

Theorem 1 (τ is sound and complete) *For any admissible formula φ and for any QSIM behavior b the following statements hold:*

1. $\tau(\varphi, b) = (\mathtt{T}, \mathtt{TRUE}) \iff$ *there exists an interpretation* M *such that* $b \models \varphi$.
2. $\tau(\varphi, b) = (\mathtt{T}, C)$ *and* $C \neq \mathtt{TRUE} \iff$ *there exists an interpretation* M *such that* $b \not\models^? \varphi$ *and if* b', b'' *exist such that* $b' = \mathcal{M}(C, b)$ *and* $b'' = \mathcal{M}(\neg C, b)$ *then* $b' \models \varphi$ *and for all full–path extensions* $\widehat{b''}$ *of* $b'' : \widehat{b''} \not\models \varphi$.
3. $\tau_1(\varphi, b) = \mathtt{F} \iff$ *for all interpretations* M *and for all full–path extensions* $\widehat{b} : \widehat{b} \not\models \varphi$ *and* $b \rhd \varphi$.
4. $\tau(\varphi, b) = (\mathtt{U}, C) \iff$ *for all interpretations* M, $b \not\models \varphi$ *and if* $b' = \mathcal{M}(\neg C, b)$ *exists then for all full–path extensions* $\widehat{b'} : \widehat{b'} \not\models \varphi$.

Proof is based on induction on formula length. It follows by a case–by–case analysis of the algorithm.

4.3 Guiding and refining the simulation

When given a behavior b and an admissible formula φ, 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 into $\mathcal{M}(C, b)$ 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 2 (TL–guide is sound and complete) *Given a QSIM behavior* b *and an admissible formula* φ *then TL–guide:*

1. *refutes* b *iff for all interpretations* M *and for all full–path extensions* $\widehat{b} : \widehat{b} \not\models \varphi$ *and* $b \rhd \varphi$.
2. *retains* b *without modifying it iff*
   (a) *there exists an interpretation* M *such that* $b \models \varphi$; *or*
   (b) *for all interpretations* M, $b \not\models \varphi$ *and there is no necessary condition* C *such that if* $b' = \mathcal{M}(\neg C, b)$ *exists then for all full–path extensions* $\widehat{b'} : \widehat{b'} \not\models \varphi$.
3. *replaces* b *with* b' *iff*
   (a) *there exists an interpretation* M *such that* $b \not\models^? \varphi$ *and exists* C *such that* $b' = \mathcal{M}(C, b) \models \varphi$ *and* C *is necessary* (i.e. *exists* b'' *such that* $b'' = \mathcal{M}(\neg C, b)$ *and for all full–path extensions* $\widehat{b''} : \widehat{b''} \not\models \varphi$); *or*
   (b) *for all interpretations* M, $b \not\models \varphi$ *and there is a necessary condition* C *such that* $b' = \mathcal{M}(\neg C, b)$ *and for all full–path extensions* $\widehat{b'} : \widehat{b'} \not\models \varphi$.

Proof follows almost directly from the previous theorem.

5 Discussion and future work

TeQSIM is designed to provide a general methodology for incorporating trajectory constraints into the qualitative simulation process. The current trajectory specification language is, however, insufficient to express certain constraints relevant to dynamical systems (*e.g.* stability requirements for controllers). Three different extensions to the language are currently being investigated.

Limited first order expressiveness - The temporal logic used is limited to PLTL and is unable to quantify over attributes of states. Certain trajectory constraints require the ability to refer to values across states within the behavior. For example, the specification of a decreasing oscillation requires the ability to compare the magnitude of a variable across states. A limited form of first order 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, QSIM 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 [1] 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." This statements is only expressible within our logic using an external predefined event. Such an extension offers the modeler more flexibility to express relevant constraints.

Functional envelopes - Semi–quantitative reasoning [3] within TeQSIM uses interval bounds and static functional envelopes for monotonic functions to derive quantitative information about a behavior. NSIM [9] derives dynamic envelopes describing a variable's behavior with respect to time. Currently, only interval information can be specified within TeQSIM trajectory constraints. Extending the language to include information about bounding envelopes with respect to time would increase the precision of solutions computed by TeQSIM.

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 [4].[2] We plan to incorporate it within TeQSIM.

6 Conclusions

Qualitative simulation and temporal logic provide two alternative formalisms for reasoning about change

[2] In all the examples we have run so far, the practical time–complexity of a TeQSIM simulation is definitely dominated by other operations (like quantitative inferences) rather than model checking.

across time. TeQSIM integrates these two paradigms by incorporating trajectory information specified via temporal logic into the qualitative simulation process. Behaviors that do not model the set of temporal logic expressions are filtered during simulation. Numeric information specified within the TL expressions can be integrated into the simulation to provide a more precise numerical description for the behaviors which model these expressions.

The correctness of the TL–guide algorithm 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 to the Qualitative Reasoning Group for many fruitful discussions. Thanks also to a careful anonymous referee.

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

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

[2] M. Barbeau, F. Kabanza, and R. St-Denis. Synthesizing plant controllers using real-time goals. In *Proc. of IJCAI–95*, pages 791–798. IJCAI, Morgan Kaufman, August 1995.

[3] D. Berleant and B.J. Kuipers. Using incomplete quantitative knowledge in qualitative reasoning. In *Proc. of the Sixth National Conference on Artificial Intelligence*, pages 324–329, 1988.

[4] 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.

[5] 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.

[6] 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.

[7] A. Farquhar and G. Brajnik. A semi–quantitative physics compiler. In *Tenth International Conference on Applications of Artificial Intelligence in Engineering*, Udine, Italy, July 1995. Presented also at the Eighth International Workshop on Qualitative Reasoning on Physical Systems, 1994, Nara, Japan.

[8] D. Jonescu and J.Y. Lin. Optimal supervision of discrete event systems in a temporal logic framework. *IEEE Transactions on Systems, Man and Cybernetics*, 25(12):1595–1605, Dec. 1995.

[9] H. Kay and B.J. Kuipers. Numerical behavior envelopes for qualitative models. In *Proc. of the Eleventh National Conference on Artificial Intelligence*. AAAI Press/MIT Press, 1993.

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

[11] 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.

[12] 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.

[13] J.G. Thistle and W.M. Wonham. Control problems in a temporal logic framework. *International Journal on Control*, 44(4):943–976, 1986.