# Qualitative Simulation and Temporal Logic: Proving Properties of Continuous Systems*

**Benjamin Shults**
Department of Mathematics
University of Texas at Austin
Austin, TX 78712
`bshults@math.utexas.edu`

**Benjamin J. Kuipers**
Computer Science Department
University of Texas at Austin
Austin, TX 78712
`kuipers@cs.utexas.edu`

January 17, 1996

**Abstract**

We demonstrate an automated method for proving statements in temporal logic about continuous systems, even in the face of incomplete knowledge. The method combines an implemented, on-the-fly, model-checking algorithm for statements in the temporal logic CTL* [3, 4, 5] with the output of the qualitative simulation algorithm QSIM [10, 12]. Based on the QSIM Guaranteed Coverage Theorem, we prove that for certain CTL* statements, $\Phi$, if $\Phi$ is true for the qualitative behavior tree produced by QSIM, then a corresponding theorem holds for the solution of any ordinary differential equation consistent with the qualitative differential equation that generated the QSIM behavior tree. We then show that for other CTL* formulas, if the QSIM output satisfies a certain completeness restriction, then conclusions can again be drawn about real-valued functions satisfying the given incomplete description. We also show how the methods can be applied to proving properties of continuous systems in which numeric information is available.

*This work has been supported in part by the National Science Foundation (grant IRI-9216584) and by the Electric Power Research Institute. A preliminary report on this work appeared as [14].

# 1 Introduction

In many applications in which ordinary differential equations (ODEs) are used, information about initial conditions or the specific relationship between a pair of quantities is not completely known. In some cases constants are only known to lie in a certain range or the relationship between quantities is only known to be monotonic. Qualitative reasoning allows this information to be used to generate descriptions of solutions to any ODE which abstracts to the known information. We call such an abstract ODE a qualitative differential equation (QDE). In many such applications we want to draw conclusions about the solution to *any* ODE consistent with the limited information we have about a system.

Furthermore, there are a number of applications of model-based reasoning that can profit from reliable inference about time-ordered events over the set of possible behaviors of a continuous system. Since applications such as control, monitoring, diagnosis and design must often cope with conditions of incomplete knowledge, the ability to do temporal reasoning over the possible behaviors of a system described by a qualitative or semi-quantitative model is particularly valuable. Our program, TL, makes a formal connection between solutions to real differential equations and temporal-logic model checking.

A qualitative simulator, such as QSIM, constructs a tree-like structure whose branches represent the possible behaviors consistent with the qualitative differential equation (QDE) and initial state input to the QSIM algorithm [10, 13]. This set of behaviors is expressed as a finite structure of qualitative state descriptions. In the case of QSIM, this structure is guaranteed to contain a branch which describes any "reasonable" extended real valued function which is a solution of an ordinary differential equation which abstracts to the QDE under circumstances to be described. We call this property the "soundness" of QSIM, and this property is the content of the Guaranteed Coverage Theorem.

Since the output of the QSIM algorithm is a structure whose paths describe reasonable, extended real valued functions, we would like to be able to formulate temporal questions about the system it describes and have those questions answered. This is accomplished using temporal-logic model-checking. A model-checking algorithm takes as input a temporal-logic formula and a tree-like structure and determines whether the structure is a model (in the logical sense of the word) for the formula. Temporal logic augments propositional logic with temporal operators on time-varying truth-values, such as *always*, *eventually*, and *until.* Modal logic adds operators for truth-values in alternate possible worlds (i.e., alternate behaviors or paths), such as *necessarily* and *possibly.*

We have chosen to use the branching-time temporal logic CTL* which is described by Emerson and Clarke [4, 5].

Because QSIM is sound, for any CTL* statement $\Phi$ which is "universal" in a sense we will define, if $\Phi$ is modeled by the structure produced by QSIM, then a corresponding theorem holds for the solution of any ordinary differential equation consistent with the QDE that generated the QSIM structure. Therefore, at least for universals, statements in temporal logic about continuous systems can be proved by qualitative simulation. This allows a hybrid reasoning system to prove common-sense statements and to do expert reasoning about dynamical systems.

In case all paths in the structure output by QSIM describe reasonable, extended real valued functions which are solutions to differential equations consistent with the QDE input to QSIM, then even CTL* formulas which are not universal may be used to prove properties of the system.

The propositional part of the temporal language includes propositions which allow the construction of formulas containing numerical information. This can be used in conjunction with the numerical extensions to QSIM—Q2 [11], Q3 [2] and NSIM [9]—in order to prove numerical properties of physical systems.

In Section 2 we describe and define the language CTL* and present some basic definitions and facts which will be useful. In Section 3 we describe the QSIM framework. In Section 4 we explain how the output of the QSIM algorithm is used as a structure over which formulas in CTL* can be interpreted and how CTL* formulas relate to the reasonable functions which the paths in the structure represent. In Section 5 we formalize the main results and discuss some issues concerning the implementation. In Section 6 we describe some applications of the combination of temporal-logic model-checking with qualitative simulation.

# 2 CTL*

Computational Tree Logic (CTL) is a branching-time temporal logic. The theory of branching-time temporal logics is summarized by Emerson in *The Handbook of Theoretical Computer Science* [5]. The popular language, CTL, is a sub-language of CTL*. We will customize CTL* slightly in order to allow states with no successors because in continuous systems a state may have no successor (e.g. if time reaches infinity or if the value of some variable crosses a boundary of its range). In this section, we define the syntax and semantics of the CTL* language and, in Section 2.4, give some basic results and definitions which will be used by our main theorems.

Our implementation (TL) of a model-checking algorithm for CTL* is an "on-the-fly" model-checker based on the algorithm of Bhat, Cleaveland and Grumberg [3]. On-the-fly algorithms have the advantage over the more common "global" algorithms of being able to terminate with the correct result before constructing the entire exponentially-large structure. If the formula happens to be in the sublanguage CTL of CTL* then the complexity of this on-the-fly algorithm is the same as the best known algorithms for CTL model-checking. Our implementation is customized for expressing statements about continuous systems (see Section 4).

## 2.1 Terminology and Notation

We interpret a CTL* formula over a temporal structure $M = \langle S, X, L \rangle$ where

- $S$ is a set of states,
- $X$ is a set of fullpaths,
- $L : S \times AP \rightarrow \{T, F\}$ is an interpretation which takes a state $s \in S$ and an atomic proposition $\phi \in AP$ and assigns a Boolean truth value.

Here $AP$ is the set of atomic propositions. A *fullpath* is a path which is either infinite or terminates with a state which has no successor. In our applications, we will be working with sets $X$ which are suffix closed but not necessarily fusion closed. A set $X$ is fusion closed if, whenever $x_1 s y_1, x_2 s y_2 \in X$, then $x_1 s y_2 \in X$. See Emerson's survey article for a review of these issues [5].

We use the notation $\langle s_0, s_1, s_2, \ldots \rangle$ to denote an infinite or finite totally-ordered set. We let $\Lambda(x)$ denote the cardinality of a finite, totally-ordered set $x$. If $x$ is an infinite, totally-ordered set, then by $i < \Lambda(x)$ we mean $i$ is any nonnegative integer. Here we use totally ordered sets to represent paths and fullpaths. Notice that the last state in a finite fullpath $x = \langle s_0, s_1, s_2, \ldots \rangle$ is $s_{\Lambda(x)-1}$.

We now describe the path quantifiers and the basic temporal operators on propositions. We prefer to give a rough description before the formal syntax and semantics are defined. Suppose some state $s$ and path $x$ starting at $s$ are given and that $p$ is a formula. The two path quantifiers are

(`necessarily` $p$), which is true at $s$ if $p$ is true of *every* fullpath starting with $s$, and

(`possibly` $p$), which is true at $s$ if $p$ is true of *some* fullpath starting at $s$.

The elementary temporal operators are (`next` $p$) and (`until` $p$ $q$) where $p$ and $q$ are formulas.

(`next` $p$) is true of the path $x$ if $\Lambda(x) = 1$ or $p$ is true of the path obtained from $x$ by deleting its first state, and

(`until` $p$ $q$) is true of $x$ if $q$ is true of some state in $x$ and $p$ is true of every state preceding the first state in which $q$ is true. This operator is sometimes called `strong-until`, to distinguish it from `weak-until` to be defined below.

The precise syntax and semantics of `until` and `next` will be defined in the following sections. We will use the following abbreviations to define other operators in terms of `until` and `next`:

(`releases` $p$ $q$) $\equiv$ (`not` (`until` (`not` $p$) (`not` $q$)))

$$
\begin{array}{rcl}
\texttt{(before } p\ q\texttt{)} & \equiv & \texttt{(not (until (not } p\texttt{) } q\texttt{))} \\
\texttt{(strong-next } p\texttt{)} & \equiv & \texttt{(not (next (not } p\texttt{)))} \\
\texttt{(eventually } p\texttt{)} & \equiv & \texttt{(until true } p\texttt{)} \\
\texttt{(always } p\texttt{)} & \equiv & \texttt{(not (eventually (not } p\texttt{)))} \\
\texttt{(never } p\texttt{)} & \equiv & \texttt{(always (not } p\texttt{))} \\
\texttt{(weak-until } p\ q\texttt{)} & \equiv & \texttt{(before } q \texttt{ (and (not } q\texttt{) (not } p\texttt{)))} \\
\texttt{(infinitely-often } p\texttt{)} & \equiv & \texttt{(always (eventually } p\texttt{))} \\
\texttt{(almost-everywhere } p\texttt{)} & \equiv & \texttt{(eventually (always } p\texttt{))}
\end{array}
$$

The formula `(releases` $p$ $q$`)` is true of a path if $q$ is always true or if $q$ is true through the first state in which $p$ is true. The statement `(before` $p$ $q$`)` is true of a path if $p$ is true in some state previous to the first state in which $q$ is true (though $q$ does not necessarily ever become true). The formula `(weak-until` $p$ $q$`)` is true of a path if $p$ is true in every state or in every state before the first state in which $q$ is true.

Because we are applying CTL* to structures which may have finite fullpaths, the temporal operator `next` may seem ambiguous. Therefore, we must distinguish between `strong-next` and `weak-next`. The statement `(weak-next` $p$`)` is true of a path if the path has no next state or if the path has a second state and $p$ is true of it. The statement `(strong-next` $p$`)` is true of a path if the path has a second state and $p$ is true of that state. In our discussion, we consider `next` alone to mean `weak-next`.

In the following two subsections we give the formal definitions for the temporal operators and path quantifiers of CTL*.

## 2.2 Syntax

A state formula is a formula which is true or false of a state and a path formula is a formula which is true or false of a path. State formulas in CTL* are generated by rules (S1–S3) below. The path formulas in CTL* are generated by rules (B1–B3) below. Although the semantics of `releases`, `strong-next` and `or` can be derived from their definitions as abbreviations, we include the definitions here so that the proofs later will be easier to follow.

**Definition 1** *The syntax of CTL* is defined as follows.*

**(S1)** *Each atomic proposition $\phi$ is a state formula,*

**(S2)** *if $p_1, \ldots, p_n$ are state formulas then so are* `(and` $p_1 \cdots p_n$`)`, `(or` $p_1 \cdots p_n$`)` *and* `(not` $p_1$`)`,

**(S3)** *if $p$ is a path formula then* `(possibly` $p$`)` *and* `(necessarily` $p$`)` *are state formulas,*

**(B1)** *every state formula is a path formula,*

**(B2)** *if* $p_1, \ldots, p_n$ *are path formulas then so are* (and $p_1 \cdots p_n$), (or $p_1 \cdots p_n$) *and* (not $p_1$),

**(B3)** *if* $p$, $q$ *are path formulas then so are* (next $p$), (strong-next $p$), (releases $p$ $q$) *and* (until $p$ $q$).

We also allow the standard boolean abbreviation for `implies`.

## 2.3 Semantics

The following notation is needed before the semantics of our logic can be defined. Given a path $x = \langle s_0, s_1, s_2, \ldots \rangle$, for every nonnegative integer $i < \Lambda(x)$ we let $x^i$ denote the path $\langle s_i, s_{i+1}, s_{i+2}, \ldots \rangle$, which is the suffix of $x$ starting at $s_i$. Thus, for any nonnegative integer $i < \Lambda(x)$, $x^i$ is the path obtained from $x$ by deleting from $x$ the first $i$ states. Notice that if $x$ is finite, then $x^i$ is not defined for $i \geq \Lambda(x)$ and $\Lambda(x^i) = \Lambda(x) - i$.

Now we are ready to give the semantics for the language. We write $M, s_0 \models \Phi$ (respectively $M, x \models \Phi$) to mean that the state formula $\Phi$ (respectively path formula $\Phi$) is true in the temporal structure $M$ at the state $s_0$ (respectively of the path $x$). Each item below gives the interpretation of the corresponding item in the syntax above.

**Definition 2** *If* $s_0$ *is a state in* $M$ *and* $x = \langle s_0, s_1, \ldots \rangle$ *is a nonempty fullpath in* $M$ *starting at* $s_0$*, then we inductively define* $\models$ *as follows:*

**(S1)** $M, s_0 \models \phi$ *where* $\phi$ *is an atomic proposition if and only if* $L(s_0, \phi) = \mathrm{T}$,

**(S2)** $M, s_0 \models$(and $p_1 \cdots p_n$) *if and only if* $M, s_0 \models p_i$ *for all* $i, 1 \leq i \leq n$,
$M, s_0 \models$(or $p_1 \cdots p_n$) *if and only if* $M, s_0 \models p_i$ *for some* $i, 1 \leq i \leq n$,
$M, s_0 \models$(not $p$) *if and only if it is not the case that* $M, s_0 \models p$,

**(S3)** $M, s_0 \models$(possibly $p$) *if and only if there is a fullpath* $y$ *in* $M$ *starting at* $s_0$*, such that* $M, y \models p$,
$M, s_0 \models$(necessarily $p$) *if and only if for every fullpath* $y$ *in* $M$ *starting at* $s_0$, $M, y \models p$.

**(B1)** $M, x \models p$ *where* $p$ *is a state formula if and only if* $M, s_0 \models p$,

**(B2)** $M, x \models$(and $p_1 \cdots p_n$) *if and only if* $M, x \models p_i$ *for all* $i, 1 \leq i \leq n$,
$M, x \models$(or $p_1 \cdots p_n$) *if and only if* $M, x \models p_i$ *for some* $i, 1 \leq i \leq n$,
$M, x \models$(not $p$) *if and only if it is not the case that* $M, x \models p$,

**(B3)** $M, x \models$(until $p$ $q$) *if and only if there is a nonnegative integer* $i < \Lambda(x)$*, such that* $M, x^i \models q$ *and for every nonnegative integer* $j < i$, $M, x^j \models p$,
$M, x \models$(releases $p$ $q$) *if and only if for every nonnegative integer* $i <$

*$\Lambda(x)$, $M, x^i \models q$ or there is a nonnegative integer $i < \Lambda(x)$ such that $M, x^i \models p$ and for every $j \leq i$, $M, x^j \models q$,*
*$M, x \models$*`(next` *$p$*`)` *if and only if $\Lambda(x) = 1$ or $M, x^1 \models p$,*
*$M, x \models$*`(strong-next` *$p$*`)` *if and only if $\Lambda(x) > 1$ and $M, x^1 \models p$.*

## 2.4 Basic Results

The proofs of our main theorems will use the fact that any formula can be written in the following form.

**Definition 3 (Positive Normal Form)** *A CTL* formula is in* positive normal form *if* `until`, `releases`, `next` *and* `strong-next` *are the only temporal operators in the formula and for every* `not` *in the formula, its scope is an atomic proposition.*

Here we require that `implies` first be rewritten in terms of `not` and `and` or `or`. Every CTL* formula is equivalent to a formula in positive normal form because all temporal operators can be written in terms of those mentioned above and `not`s can be propagated inward to propositions [3].

**Definition 4 (Universal Formula)** *A CTL* expression $\Phi$ is said to be* universal *if, when the formula is written in positive normal form, there are no occurrences of the path quantifier* `possibly`.

We call a path formula a *perfect path formula* if it contains no path quantifiers. These are exactly the formulas which correspond to formulas in Propositional Linear Time Logic (PLTL). If $\Phi$ is a formula in CTL*, then $\Phi'$ denotes the perfect path formula obtained from $\Phi$ by deleting all occurrences of the path quantifiers. For example, if $p$ and $q$ are propositions and

$\Phi =$ `(necessarily (until` $p$ `(necessarily` $q$`)))`,

then

$\Phi' =$ `(until` $p$ $q$`)`.

The following lemma is needed in the proof of Lemma 2 which is used in the proof of one of the main theorems.

**Lemma 1** *If $\Phi$ is a universal formula and $x$ is a fullpath in $M$ such that $M, x \models \Phi$, then $M, x \models \Phi'$.*

**Proof:** Without loss of generality, we assume that $\Phi$ is in positive normal form. The proof of the lemma is by induction on the length of $\Phi$.

In case $\Phi$ is an atomic proposition, $\Phi = \Phi'$ and hence $M, x \models \Phi$ if and only if $M, x \models \Phi'$. This justifies the base case.

Suppose that every universal formula of length less than $k$ makes the theorem true and that $\Phi$ has length $k$.

If $\Phi =$ `(necessarily` $p$`)`, then for every fullpath $y$ starting at the first state in $x$, $M, y \models p$. Therefore, by induction, for every fullpath $y$ starting at the first state in $x$, $M, y \models p'$. In particular, $M, x \models p'$ and hence $M, x \models \Phi'$.

If $\Phi =$ `(and` $p_1 \cdots p_n$`)`, then $M, x \models p_i$ for each $i, 1 \leq i \leq n$. We need to show that, $M, x \models p'_i$ for each $i$. This follows by induction. Thus, $M, x \models \Phi'$.

If $\Phi =$ `(or` $p_1 \cdots p_n$`)`, then $M, s \models p_i$ for some $i, 1 \leq i \leq n$. We need to show that, $M, x \models p'_i$ for some $i$. This follows by induction. Therefore, $M, x \models \Phi'$.

If $\Phi =$ `(not` $p$`)`, then $\Phi$ is an atomic proposition since $\Phi$ is in positive normal form and so $p = p'$.

If $\Phi =$ `(until` $p$ $q$`)`, then there is a nonnegative integer $i < \Lambda(x)$ (we choose the smallest) such that $M, x^i \models q$ and for every nonnegative integer $j < i$, $M, x^j \models p$. Therefore, $M, x^i \models q'$ and for every nonnegative integer $j < i$, $M, x^j \models p'$ by induction. Therefore, $M, x \models$ `(until` $p'$ $q'$`)`.

If $\Phi =$ `(releases` $p$ $q$`)`, then for every nonnegative integer $i < \Lambda(x)$ such that $M, x^i \not\models q$ there is a nonnegative integer $j < i$ such that $M, x^j \models p$.

First suppose that for every nonnegative integer $i < \Lambda(x)$, $M, x^i \models q$. Then $M, x^i \models q'$ for every nonnegative integer $i < \Lambda(x)$ by induction. Therefore, $M, x \models \Phi'$.

Now suppose that there is a nonnegative integer $i < \Lambda(x)$ such that $M, x \not\models q$. We select $i$ to be the smallest such nonnegative integer. Thus, there is a nonnegative integer $j < i$ such that $M, x^j \models p$. For all such $j$ we also have $M, x^j \models p'$ by induction. For every nonnegative integer $k < i$, we have $M, x^k \models q$ by the choice of $i$ and so $M, x^k \models q'$ by induction. Therefore, we get that for every nonnegative integer $l < \Lambda(x)$ such that $M, x^l \not\models q'$ there is a nonnegative integer $j < l$ such that $M, x^j \models p'$. That is to say, $M, x \models$ `(releases` $p'$ $q'$`)`.

If $\Phi =$ `(next` $p$`)`, then if $\Lambda(x) > 0$ then $M, x^1 \models p$ and we are done by induction. If $\Lambda(x) = 0$ then $M, x \models$ `(next` $p'$`)` as well.

If $\Phi =$ `(strong-next` $p$`)`, then $\Lambda(x) > 0$ and $M, x^1 \models p$ so we are done by induction.

□

**Lemma 2** *For every universal CTL* state formula $\Phi$, and every temporal structure $M$ and state $s$ in $M$, if $M, s \models \Phi$ then for every fullpath $x$ in $M$ starting at $s$, $M, x \models \Phi'$.*

**Proof:** The proof follows easily by induction on the length of $\Phi$ by using Lemma 1.

□

# 3 QSIM

In Section 3.1 we briefly describe the QSIM framework. We refer the reader to Kuipers' full description of the QSIM framework [13] and to Shults' reformalization of QSIM [18] for details on the new definition of a reasonable function and other reformalizations of concepts related to the Guaranteed Coverage Theorem.

## 3.1 The QSIM Framework

A *qualitative differential equation* (QDE) consists of a finite set of *variables* (each of which is associated with a *quantity space* which is a totally-ordered set of *landmarks*), and a set of constraints on the values of the variables. A QDE is a structural abstraction of a class of ordinary differential equations (ODEs). The QDE codifies the QSIM user's incomplete knowledge of a physical system.

Starting with a QDE, $C$, and an initial state, $I$, qualitative simulation with QSIM produces a finite tree, $M$, of *qualitative states*, linked by the QSIM *successor* relation, $R$. The finite tree which it produces is called a *QSIM behavior tree* in the literature. We let $M = \langle S, B \rangle$ represent the behavior tree where $S$ is the set of states, $B$ is the set of fullpaths (represented as finite, totally-ordered sets). We will say that this QSIM behavior tree was *generated by* the pair $\langle C, I \rangle$ of the QDE and the initial state. A *QSIM behavior* is a path in the behavior tree, starting at the root and terminating at a leaf of the tree, i.e. $B$ is the set of QSIM behaviors in $M$. Each state describes the *qualitative value* of each *variable* appearing in the QDE model. Each variable will represent a function of time. The qualitative value of a variable $v$ over a state $s$ is of the form $\langle qmag, qdir \rangle$, where *qmag* describes the magnitude of $v$ as equal to a landmark or in an open interval defined by two landmarks, and *qdir* is the sign of the derivative of $v$. By considering the qualitative values of the variables at a state, and the constraints in the QDE, QSIM is able to derive a number of properties of the states and behaviors, including quiescence, stability and cycles.

Given an ODE, $F$, there is associated with it a set, $F'$, of simultaneous equations which is derived from $F$. See Shults' reformalization for more details [18]. $F'$ is useful because it can be easily *abstracted* to a QDE and a solution to $F'$ can be converted into a solution to $F$. We will let $U$ denote the solution to $F'$. That is to say that $U$ is a set of functions which make each of the equations in $F'$ true. If $M$ is a QSIM tree produced by the QDE, $C$, abstracted from $F'$, we let $\psi_{U,M}$ be the bijection from $U$ to the set of variables in $C$, the abstraction of $F'$. The bijection is simply the relationship between the names of the variables in $F'$ and the names of the variables in $C$.

A QSIM state is called a *transition state* if it has no $R$-successors due to the fact that the value of one of its variables crosses a boundary of the QDE description. QSIM allows the user to produce *transition relations* between a transition state in one tree and the root of a tree generated by another QDE. This allows the user to produce a tree which has different models for its behavior in different ranges. The theorems here could be extended to take transition relations into account, however, the extension is tedious and unenlightening so, in the theorems in this paper, we assume that transition states have no successors.

**Closed Trees** Ideally, given a QDE, the QSIM algorithm will terminate, not because it runs out of memory or other resources, but because it has finished simulating all possible behaviors. When the QSIM algorithm terminates in this "natural" way, we call the tree it produces *closed.* In this case, every behavior in the behavior tree returned by QSIM terminates with a state which is a transition state, a cycle state or a quiescent state. There are cases, however, in which QSIM does not return a closed tree regardless of how long it is allowed to run. In cases where QSIM returns a tree which is not closed, the hypotheses of the Guaranteed Coverage Theorem do not hold. If the behavior tree $M$ is not closed then it is possible that an actual behavior of the system is not represented by any rooted fullpath in the represented structure $\widehat{M}$.

The normal QSIM simulation style creates new landmarks for critical values, applies a strong cycle-match criterion (all variables must have identical landmark values), and does not allow *cross-edge* cycles (i.e. considers cycle matches only within the same QSIM behavior.) Under this simulation style, certain systems such as the damped spring never close. However, by applying the *envisionment* simulation style (no new landmarks, weak cycle-match criterion, and cycle matches anywhere in the behavior tree), *every* qualitative model has a finite closed behavior tree. (See Chapter 5 of Kuipers' *Qualitative Reasoning* [13].)

**The Represented QSIM Structure** QSIM may use various matching criteria when it detects cycles. The *strong-match* criterion requires that the value of each variable in the states to be matched is a landmark (rather than an interval) and that those landmarks match the values of the variables in the previously-existing state (the qualitative derivatives have to match regardless of the match criterion). The *weak-match* criterion allows a match when the values are intervals or landmarks. The QSIM user may also dictate whether cycles are detected across QSIM behaviors (cross-edge cycles) or only on the same QSIM behavior. The type of cycle detection chosen makes a difference in the interpretation of the tree. We want to do temporal reasoning about paths which pass through these cycle states. Therefore we will define what we call the *QSIM structure,* $\widehat{M} = \langle S, X \rangle$, *represented by* $M$, where $X$ is the set of fullpaths represented in the behavior tree $M$. A *fullpath* in the QSIM structure is a path which is infinite or terminates in a state which is not a cycle state and has no $R$-successor. See Shults' reformalization of QSIM [18] to see exactly how $\widehat{M}$ is constructed from $M$. First we define the set $X^r$ of *rooted fullpaths.* A rooted fullpath is a path starting at the root of the QSIM tree which continues through cycle states in a semantically sensible way. To obtain the set $X^r$ we do not necessarily add the cycles to the QSIM successor relation, $R$. We only include fullpaths which are sensible. $X$ is the suffix closure of $X^r$.

If we are using strong-match cycle detection and no cross-edge cycles, then when a cycle is detected, the system has returned to a state where it has been before; by the uniqueness theorem for differential equations, it must continue on the exact path which it followed previously. Therefore, a cycle behavior represents a single fullpath. As a result, there is a bijection $z : X^r \to B$. Thus, $X$ is not necessarily fusion closed and hence not $R$-generable. (A set $X$ is $R$-generable if there exists a total binary relation $R$ on $S$ such that a path $x = \langle s_0, s_1, s_2, \ldots \rangle \in X$ if and only if for every nonnegative integer $i$, $\langle s_i, s_{i+1} \rangle \in R$.) Since, in this case, any fullpath in $X$ is a suffix of only one rooted fullpath, we can extend $z$ so that for any $x \in X, z(x) = z(x_r)$ where $x_r$ is the rooted fullpath of which $x$ is a suffix. (Thus, $z$ is no longer a bijection.) We think of $z$ as a map from a fullpath to its associated QSIM behavior. This assignment will be useful when we prove properties of systems about which we have some quantitative information.

**Quantitative Information** When strong-cycle matching is used and cross-edge cycles are not allowed, then the QDE and the initial state may be augmented with quantitative information such as numerical interval bounds on the real values denoted by landmarks and other symbolic terms in the behavior prediction [2, 9, 11, 13]. In this case, QSIM propogates this quantitative information and uses it to prune branches of the tree which are inconsistent with the information. The most important quantitative information for the purposes of this paper is the information which QSIM derives about the landmarks.

A landmark in a quantity space of a QSIM variable is intended to name some real number. The quantitative extensions to QSIM are able to restrict the possible values of a landmark to some closed, extended-real interval. This quantitative information may be different on each QSIM behavior. That is to say that on different QSIM behaviors, a given landmark may name different real numbers. Therefore, if weak-cycle matching is used or cross-edge cycles are detected, then the numeric information loses its sense [18].

## 3.2 Qualitative Description

For the statement of the main theorems, as well as the Guaranteed Coverage Theorem, we need to have a formal way of relating fullpaths in the QSIM structure with the underlying reasonable, extended real valued functions which they are meant to be describing. For this purpose, we will abstract the functions by means of a *splitting* and we will "instantiate" the qualitative fullpath by means of a *specification.*

**Splittings** We are given a set of reasonable, extended real valued functions $U = \{u_i : 1 \leq i \leq n\}$. Since we are thinking of $U$ as a solution to a set of simultaneous equations which was derived from an ODE, we will assume that each of the functions in $U$ shares the domain, $A$, some interval in the extended reals. We define a *splitting* of $U$ as follows. Let $\{t_i\}$ be a strictly increasing sequence of points (indexed from 0) in $A$ satisfying the following conditions: 1) $\{t :$ for some $k, t$ is a critical point of $u_k\} \subset \{t_i\}$, 2) $\{t_i\}$ has no finite limit point and 3) $\{t_i\}$ converges to $\infty$ only if $\infty \notin A$. According to the definition of a reasonable function [18], such a set exists, and may be infinite only if $\infty$ is the supremum of $A$ and $A$ is open on the right. We will call the pair $\langle\{t_i\}, U\rangle$ a *splitting* of $U$. Since the critical points of the functions all must be in $\{t_i\}$, a splitting is distinguished from another splitting by the choice of non-critical points in $\{t_i\}$. If the set $\{t_i\}$ is finite, then we will let $t_J$ be the greatest element so that $J + 1$ is the cardinality of $\{t_i\}$.

Associated with any splitting $\langle\{t_i\}, U\rangle$ there is a natural partition $\{D_k\}$ of the interval $A$. Each $D_k$ is either a singleton containing one of the points in $\{t_i\}$ or an open interval whose endpoints are two consecutive points in $\{t_i\}$. The indexes on the sets in the partition follow the order of the indexes in $\{t_i\}$. That is, if $A$ is closed on the left then $D_0 = \{t_0\}, D_1 = (t_0, t_1)$, and so on. If $A$ is open on the left then $D_0 = (a, t_0), D_1 = \{t_0\}, D_2 = (t_0, t_1)$, and so on, where $a$ is the infimum of $A$. If $A$ is closed on the right then $D_{2J} = \{t_J\}$ (or $D_{2J+1} = \{t_J\}$ if $A$ is open on the left). If $A$ is open on the right, then as we have said, there may be infinitely many sets in $\{D_k\}$.

When dealing with a splitting, $\langle\{t_i\}, U\rangle$, of $U$, we will use the following abbreviations:

$$\begin{aligned} U|_l &= \{u_i|_{D_l} : 1 \leq i \leq n\} \\ U|_{l+} &= \{u_i|_{\bigcup_{l \leq k}\{D_k\}} : 1 \leq i \leq n\} \end{aligned}$$

That is to say that $U|_l$ is the set of functions in $U$ each restricted to the domain $D_l$ and $U|_{l+}$ is the set of functions in $U$ each restricted to the domain $\bigcup_{l \leq k}\{D_k\}$.

**Specification** In order to define the notion of a splitting being *described* by a fullpath, we will need to relate the landmarks in the fullpath to specific extended real numbers. We call a mapping of landmarks to extended real numbers satisfying certain sensible conditions, a *specification* of the landmarks. We will want the extended real numbers to which the specification assigns landmarks to be consistent with the order in which the landmarks occur in the quantity spaces of the variables of which they are values. Also if quantitative information is assigned on a fullpath, we want the mapping to be consistent with that information.

A function, $c$, with domain, some partially ordered set, $V$, of landmarks and range, $R^*$, is called a *specification* of $V$ if $c$ preserves the partial order and

$c(\mathtt{minf}) = -\infty, c(\mathtt{inf}) = \infty$ and $c(0) = 0$.

Given a fullpath $x$ and a specification, $c$, of the landmarks of the variables in $x$ under the partial order determined by the quantity spaces of the variables in $x$, we call the pair $\langle x, c\rangle$ a *specification of $x$* if $c$ is also consistent with any numeric information which might be associated with $z(x)$. (Recall, $z(x)$ is the QSIM behavior associated with $x$ when such can be determined uniquely.) In particular, if, on the behavior $z(x)$, the qualitative landmark, X1, has been determined, by a quantitative extension to QSIM, to refer to a number in the numeric range $[n_1, n_2]$, then $c(\mathtt{X1}) \in [n_1, n_2]$.

**Definition of Qualitatively Describes** Given a specification $\langle x, c\rangle$ of a fullpath $x = \langle s_0, s_1, \ldots\rangle$ in a QSIM directed graph $\widehat{M}$ and a splitting $\langle\{t_i\}, U\rangle$ of the set of reasonable, extended real valued functions $U = \{u_i : 1 \leq i \leq n\}$ with common domain, $A$, we say that $\langle x, c\rangle$ *qualitatively describes* the splitting if the specification corresponds to the splitting as described in detail in the remainder of this subsection.

The intention is that the partition, $\{D_k\}$, of the domain, $A$, determined by the splitting will correspond to the range of the time variable in the fullpath $x$ in such a way that each element, $D_k$, of the partition will correspond to the value of the time variable in the state $s_k$ and thus $s_k$ will correspond to $U|_k$.

The function $\psi_{U,M}$ referred to below is the bijection described in Section 3.1 which relates the variables in $M$ with the variables in $U$.

The cardinality of $\{t_i\}$ must equal the number of time-point states in $x$. Therefore, since the states in $x$ alternate between time-point states and time-interval states, $\langle x, c\rangle$ qualitatively describes $\langle\{t_i\}, U\rangle$ if and only if there is a correspondence between $x$ and $U$ in which $s_k$ corresponds to $U|_k$ as follows:

- Given any landmark X1 of a QSIM variable $\psi_{U,M}(u_i)$ and any nonnegative integer $k$, the qualitative value of $\psi_{U,M}(u_i)$ at the state $s_k$ is X1 if and only if $u_i|_{D_k} = c(\mathtt{X1})$.
- $u_i'|_{D_k} > 0$ if and only if the qualitative derivative of the variable $\psi_{U,M}(u_i)$ in $s_k$ is inc.
- $u_i'|_{D_k} = 0$ if and only if the qualitative derivative of the variable $\psi_{U,M}(u_i)$ in $s_k$ is std.
- $u_i'|_{D_k} < 0$ if and only if the qualitative derivative of the variable $\psi_{U,M}(u_i)$ in $s_k$ is dec.

**Lemma 3** *The specification $\langle x, c\rangle$ of $x$ qualitatively describes $\langle\{t_i\}, U\rangle$ if and only if $\langle x^h, c\rangle$ qualitatively describes $\langle\{t_i : h \leq i\}, U|_{h+}\rangle$ for every nonnegative integer $h < \Lambda(x)$.*

**Proof:** The proof comes straight from the definition of what it means for a specification of a fullpath to describe qualitatively a splitting of a set of functions. This definition involves a correspondence between $s_h$ with $D_h$ for every nonnegative integer $h < \Lambda(x)$.

□

It follows from this definition and the definition of the QSIM algorithm that if $\langle x, c\rangle$ qualitatively describes a splitting $\langle \{t_i\}, U\rangle$ of a set of reasonable, extended real valued functions, then given any pair of landmarks, `X0` and `X1`, of a QSIM variable $\psi_{U,M}(u_i)$ and any nonnegative integer $k$, if the qualitative value of $\psi_{U,M}(u_i)$ at the state $s_k$ is the interval `(X0 X1)` then $u_i(D_k) \subset (c(\mathtt{X0}), c(\mathtt{X1}))$. It also follows that the state $s_k$ has been determined by QSIM to be quiescent if and only if $u_i'|_{D_k} = 0$ for each $1 \leq i \leq n$ and that the fullpath $x$ is finite and QSIM has labeled the last state in $x$ with `t=inf` if and only if $t_J = \infty$.

### 3.3 The Guaranteed Coverage Theorem

Suppose $M$ is a closed tree generated from $\langle C, I\rangle$ and the ODE, $F$, abstracts to $C$ and $F'$ has solution $U$. The QSIM algorithm is carefully crafted to guarantee that $U$ is described by some rooted fullpath in $\widehat{M}$. This is the idea behind the Guaranteed Coverage Theorem.

**Theorem 1 (Guaranteed Coverage)** *Under the conditions above, there is a rooted fullpath $x$ in $\widehat{M}$ and a specification $\langle x, c\rangle$ of $x$ such that $\langle x, c\rangle$ qualitatively describes some splitting $\langle \{t_i\}, U\rangle$ of $U$.*

The proof is given by Kuipers [10, 13]. Because most of the QSIM framework—the algorithm itself, for example—is beyond the scope of this paper, we will not detail the proof here.

## 4 QSIM and the Logic

Given a formula $\Phi$ in CTL* and a temporal structure $M$, a model-checking program can determine whether a given state or fullpath in the structure $M$ is a model for $\Phi$. Here we consider how this process is applied to QSIM. We show how QSIM structures are treated as temporal structures and how a CTL* formula relates to a solution to a differential equation.

### 4.1 QSIM Structures for CTL*

The *temporal structure* $\widehat{M}_{TL} = \langle S, X, L\rangle$, *represented by* a QSIM behavior tree $M = \langle S, B\rangle$, is obtained from $\widehat{M} = \langle S, X\rangle$ by the interpretation $L$ of the propositions given in Figure 1 in which $s$ represents the state over which the propositions are being interpreted.

This temporal structure $\widehat{M}_{TL}$ is the structure over which we will interpret CTL* formulas.

Figure 1: The propositional level of the language

`(qval` $v$ `(`$qmag$ $qdir$`))` where $v$ is a variable of the state $s$, $qmag$ 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`}. Such a proposition is true when the qualitative derivative of $v$ in $s$ is $qdir$ and the qualitative magnitude of $v$ in $s$ is $qmag$ or if $qmag$ is an interval and the qualitative magnitude of $v$ is a subset of $qmag$.

`(status quiescent)` is true exactly when the qualitative derivative of each variable in the state is `std`.

`t=inf` is true at a state if QSIM was able to determine that the time in this state must be infinite.

`t<inf` is true at a state if QSIM was able to determine that the time in this state must be finite.

`(in-range` $v$ `(`$n_1$ $n_2$`))` where $v$ is a variable in the state $s$ and $n_1$ and $n_2$ are extended real numbers. If the value of $var$ in $s$ is a landmark then this proposition is true if and only if the number represented by that landmark in $s$ is known to lie in an interval which is a subset of $[n_1, n_2]$. If the value of $var$ in $s$ is an interval `(X1 X2)`, then this proposition is true if and only if the interval $[n_1, n_2]$ contains both of the intervals in which QSIM has determined the numbers named by `X1` and `X2` to lie.

Our implementation, TL, of the model checker includes propositions in the language which are not mentioned in Figure 1 but are useful in practice. Since they add clutter to the statements of definitions and theorems in this paper, we will describe some of these operators in Appendix A and explain what adjustments need to be made to definitions and proofs in order to retain our theorems.

The propositions `t=inf` and `t<inf` allow the user to express the difference between, for example, "eventually in a possibly asymptotic sense" and "eventually in finite time". Alone, `eventually` really means "eventually in a possibly asymptotic sense". In order to express "eventually in finite time", use the propositions `t=inf` and `t<inf`. For example we may say `(eventually (and` $p$ `t<inf))` to mean that $p$ becomes true in finite time.

The proposition `in-range` is sensible only in the states of behavior trees which have had some type of quantitative extension to QSIM (e.g. Q2 [11], Q3 [2] or NSIM [9]) run on them. The numbers referred to in these expressions are extended real numbers: they may be `-inf` or `+inf` as well as real values. The use of the numeric propositions and quantitative information derived by QSIM from the numeric information given in the QDE, allows TL to prove time-related properties of physical systems.

The expressiveness of the application of CTL* to QSIM can easily be increased without adding to the complexity of model-checking by augmenting the propositional part of the language. See Appendix A for some such extensions.

## 4.2 Temporal Description

Here we define what it means for a CTL* path formula $\Phi$ to describe a set of functions. First we need to define a specification of a formula. Let $V$ be the set of landmarks of the variables mentioned in the formula $\Phi$ and suppose $V$ is partially ordered. If $c$ is a function from $V$ to $R^*$ which preserves the partial order, then $\langle\Phi, c\rangle$ is a specification of $\Phi$.

Let $\langle\Phi, c\rangle$ be a specification of a perfect path formula $\Phi$ and let $\langle\{t_i\}, U\rangle$ be a splitting of a set of reasonable, extended real valued functions $U = \{u_i : 1 \leq i \leq n\}$ on a common domain $A$ and let $\{D_k\}$ denote the cover associated with $\{t_i\}$. Let $\psi$ be a bijection from some subset $U_1$ of $U$ to the set of variables mentioned in the formula $\Phi$. We recursively define what it means to say that $\langle\Phi, c\rangle$ *temporally describes* the splitting $\langle\{t_i\}, U\rangle$ *via* $\psi$.

- The following propositions must correspond to the splitting as follows:
  - $\Phi =$ `(qval` $\psi(u_i)$ `(`*qmag qdir*`))` if and only if
    - if *qmag* is the landmark value `X0` of $\psi(u_i)$ then $u_i|_{D_0} = c(\texttt{X0})$,
    - if *qmag* is an interval `(X1 X2)` in the quantity space of $\psi(u_i)$ whose endpoints are landmark values of $\psi(u_i)$ then $u_i(D_0) \subset (c(\texttt{X1}), c(\texttt{X2}))$,
    - *qdir* = `inc` if and only if $u_i'|_{D_0} > 0$,
    - *qdir* = `std` if and only if $u_i'|_{D_0} = 0$ and
    - *qdir* = `dec` if and only if $u_i'|_{D_0} < 0$,
  - $\Phi =$ `(status quiescent)` if and only if $u_i'|_{D_0} = 0$ for each $1 \leq i \leq n$,
  - $\Phi =$ `t=inf` if and only if $D_0 = \{\infty\}$,
  - $\Phi =$ `t<inf` if and only if every element of $D_0$ is a real number,
  - $\Phi =$ `(in-range` $\psi(u_i)$ `(`$n_1$ $n_2$`))` if and only if $u_i(D_0) \subset [n_1, n_2]$,

- $\Phi =$ `(and` $p_1 \cdots p_m$`)` if and only if $p_k$ temporally describes $\langle\{t_i\}, U\rangle$ via $\psi$ for each $1 \leq k \leq m$,

- $\Phi =$ `(or` $p_1 \cdots p_m$`)` if and only if $p_k$ temporally describes $\langle\{t_i\}, U\rangle$ via $\psi$ for some $1 \leq k \leq m$,

- $\Phi =$ `(not` $p$`)` if and only if $p$ does not temporally describe $\langle\{t_i\}, U\rangle$ via $\psi$,

- $\Phi =$ `(until` $p$ $q$`)` if and only if for some nonnegative integer $h, q$ temporally describes $\langle\{t_i : h \le i\}, U|_{h+}\rangle$ via $\psi$ and for every nonnegative integer $l < h$, $p$ temporally describes $\langle\{t_i : l \le i\}, U|_{l+}\rangle$ via $\psi$,
- $\Phi =$ `(releases` $p$ $q$`)` if and only if for every nonnegative integer $h$ such that $q$ does not temporally describe $\langle\{t_i : h \le i\}, U|_{h+}\rangle$ via $\psi$, there is a nonnegative integer $l < h$ such that $p$ temporally describes $\langle\{t_i : l \le i\}, U|_{l+}\rangle$ via $\psi$,
- $\Phi =$ `(next` $p$`)` if and only if $A = D_0$ or $p$ temporally describes $\langle\{t_i : 1 \le i\}, U|_{1+}\rangle$ via $\psi$ and
- $\Phi =$ `(strong-next` $p$`)` if and only if $A \neq D_0$ and $p$ temporally describes $\langle\{t_i : 1 \le i\}, U|_{1+}\rangle$ via $\psi$.

# 5 The Main Results

Section 5.1 contains Theorem 2 which relates the two descriptions of a set of functions defined in Sections 3.2 and 4.2. The remainder of this section contains the theorems which are the most important of our theorems for applications.

## 5.1 Determined QSIM Trees

In this section, we want to prove that if $x$ is a fullpath in a closed QSIM structure $\widehat{M}$ and $\widehat{M}_{TL}, x \models \Phi$ where $\Phi$ is a perfect path formula and $\langle x, c\rangle$ is a specification of $x$ which qualitatively describes $\langle\{t_i\}, U\rangle$ then $\langle \Phi, c\rangle$ temporally describes $\langle\{t_i\}, U\rangle$ via $\psi_{U,M}$.

In order to prove this, we need to be certain that QSIM determines the information in propositions completely and correctly. Formally, we need to know that if $s$ is a state in a QSIM tree $M$ and $\widehat{M}_{TL}, s \models \phi$ where $\phi$ is an atomic proposition then for every fullpath, $y$, starting at $s$, $\langle y, c\rangle$ qualitatively describes $\langle\{t_i\}, U\rangle$ if and only if $\langle \phi, c\rangle$ temporally describes $\langle\{t_i\}, U\rangle$ via $\psi_{U,M}$. In an arbitrary QSIM tree, this may not be true. This may fail in a transition state at which it is impossible to determine whether `t=inf` or `t<inf`.

We define a QSIM state to be *determined* with respect to the propositions `t=inf` and `t<inf` if QSIM has determined one of `t=inf` or `t<inf`.

The user can have the TL program warn the user about any state which is not determined with respect to an atomic proposition being queried on that state. When a state is not determined, the TL program still operates but the hypotheses of the theorems relating the operation of TL with the reasonable, extended real valued functions are no longer satisfied.

**Lemma 4** *If $s$ is a state in a QSIM tree $M$ which is determined with respect to the proposition $\phi$ and $\widehat{M}_{TL}, s \models \phi$, then for every fullpath, $x$, starting at $s$,* *if $\langle x, c\rangle$ is a specification of $x$, then $\langle x, c\rangle$ qualitatively describes $\langle\{t_i\}, U\rangle$ if and only if $\langle\phi, c\rangle$ temporally describes $\langle\{t_i\}, U\rangle$ via $\psi_{U,M}$.*

**Proof:** Suppose that $s$ is a state in a QSIM tree $M$ and $\widehat{M}_{TL}, s \models \phi$ where $\phi$ is an atomic proposition with respect to which $s$ is determined. Let $x$ be a fullpath starting at $s$ and $\langle x, c\rangle$ be a specification of $x$. Let $U$ be a set of reasonable, extended real valued functions on some interval with splitting $\langle\{t_i\}, U\rangle$.

The result follows directly from the definition of the semantics of the proposition (given in Figure 1, the definition of the meaning of a specification of a formula temporally describing a splitting of a set of functions and the definition of the meaning of a specification of a fullpath qualitatively describing a splitting of a set of functions.

□

We say that a QSIM structure $\widehat{M}$ is determined with respect to a proposition if every state in $\widehat{M}$ is determined with respect to the proposition.

Theorem 2 relates the two ways of describing a set of reasonable, extended real valued functions and will be used in the proofs of the main theorems of this paper.

**Theorem 2** *Suppose $x$ is a fullpath in a QSIM structure $\widehat{M}$ which is determined with respect to all of the propositions in $\Phi$, a perfect path formula, and $\widehat{M}_{TL}, x \models \Phi$. If the specification $\langle x, c\rangle$ of $x$ qualitatively describes $\langle\{t_i\}, U\rangle$ then $\langle\Phi, c\rangle$ temporally describes $\langle\{t_i\}, U\rangle$ via $\psi_{U,M}$.*

**Proof:** Without loss of generality, we assume that $\Phi$ is in positive normal form. Suppose that $x$ is a fullpath in a QSIM structure $\widehat{M}$ which is determined with respect to all of the propositions in $\Phi$, a perfect path formula in positive normal form, and $\widehat{M}_{TL}, x \models \Phi$. We will apply induction on the length, $k$, of $\Phi$. In the base case, $\Phi$ is an atomic proposition. This case follows from Lemma 4.

Our induction hypothesis says that for any fullpath $y$ in a QSIM structure such that $\widehat{M}_{TL}, y \models \Psi$ where $\Psi$ is a perfect path formula in positive normal form of length less than $k$ and $M$ is determined with respect to every proposition in $\Psi$, then if $\langle y, c\rangle$ qualitatively describes a splitting $\langle\{r_i\}, V\rangle$ then $\langle\Psi, c\rangle$ temporally describes $\langle\{r_i\}, V\rangle$ via $\psi_{U,M}$.

If $\Phi = \texttt{(and } p_1 \cdots p_m\texttt{)}$, then we must show that if $\langle p_j, c\rangle$ temporally describes $\langle\{t_i\}, U\rangle$ via $\psi_{U,M}$ for each $j, 1 \le j \le m$ then $\widehat{M}_{TL}, x \models p_j$ for each $j, 1 \le j \le m$. This follows by the induction hypothesis.

If $\Phi = \texttt{(or } p_1 \cdots p_m\texttt{)}$, then we must show that if $\langle p_j, c\rangle$ temporally describes $\langle\{t_i\}, U\rangle$ via $\psi_{U,M}$ for some $j, 1 \le j \le m$ then $\widehat{M}_{TL}, x \models p_j$ for some $j, 1 \le j \le m$. Again, this follows by induction.

If $\Phi = \texttt{(not } p\texttt{)}$, then we must show that if $\langle p, c\rangle$ does not temporally describe $\langle\{t_i\}, U\rangle$ via $\psi_{U,M}$ then $\widehat{M}_{TL}, x \not\models p$. Since, by the definition of positive normal form, $p$ must be an atomic proposition, this follows from the induction hypothesis and Lemma 4.

If $\Phi =$ (until $p$ $q$), then we must show that if $\langle x, c\rangle$ qualitatively describes $\langle\{t_i\}, U\rangle$, then $\langle\Phi, c\rangle$ temporally describes $\langle\{t_i\}, U\rangle$ via $\psi_{U,M}$. By the semantics of until we know that there is a nonnegative integer $h < \Lambda(x)$ such that $\widehat{M}_{TL}, x^h \models q$ and for every nonnegative integer $l < h, \widehat{M}_{TL}, x^l \models p$. Since $\langle x, c\rangle$ qualitatively describes $\langle\{t_i\}, U\rangle$, $\langle x^h, c\rangle$ qualitatively describes $\langle\{t_i : h \leq i\}, U|_{h+}\rangle$ by Lemma 3. Thus $\langle q, c\rangle$ temporally describes $\langle\{t_i : h \leq i\}, U|_{h+}\rangle$ via $\psi_{U,M}$ by induction. For every nonnegative integer $l < h$, $\langle x^l, c\rangle$ qualitatively describes $\langle\{t_i : l \leq i\}, U|_{l+}\rangle$ by Lemma 3. Thus $\langle p, c\rangle$ temporally describes $\langle\{t_i : l \leq i\}, U|_{l+}\rangle$ via $\psi_{U,M}$ by induction. So $\langle\Phi, c\rangle$ temporally describes $\langle\{t_i\}, U\rangle$ via $\psi_{U,M}$.

If $\Phi =$ (releases $p$ $q$), then we must show that if $\langle x, c\rangle$ qualitatively describes $\langle\{t_i\}, U\rangle$ then $\langle\Phi, c\rangle$ temporally describes $\langle\{t_i\}, U\rangle$ via $\psi_{U,M}$. Since $\widehat{M}_{TL}, x \models \Phi$ we know that for every nonnegative integer $h < \Lambda(x)$ such that $\widehat{M}_{TL}, x^h \not\models q$ there is a nonnegative integer $l < h$ such that $\widehat{M}_{TL}, x^l \models p$. First suppose that $\widehat{M}_{TL}, x^h \models q$ for every nonnegative integer $h < \Lambda(x)$. We know that $\langle x^h, c\rangle$ qualitatively describes $\langle\{t_i : h \leq i\}, U|_{h+}\rangle$ for every nonnegative integer $h < \Lambda(x)$ by Lemma 3. Therefore, $\langle q, c\rangle$ temporally describes $\langle\{t_i : h \leq i\}, U|_{h+}\rangle$ via $\psi_{U,M}$ for every nonnegative integer $h < \Lambda(x)$ by induction and so $\langle x, c\rangle$ qualitatively describes $\langle\{t_i\}, U\rangle$.

Now suppose that there is a nonnegative integer $h < \Lambda(x)$ such that $\widehat{M}_{TL}, x^h \not\models q$ and it is the smallest such $h$. There is a nonnegative integer $l < h$ such that $\widehat{M}_{TL}, x^l \models p$. For every nonnegative integer $j < h, \widehat{M}_{TL}, x^j \models q$ and since $\langle x^j, c\rangle$ qualitatively describes $\langle\{t_i : j \leq i\}, U|_{j+}\rangle$, $\langle q, c\rangle$ temporally describes $\langle\{t_i : j \leq i\}, U|_{j+}\rangle$ via $\psi_{U,M}$ by induction. For one of these $j, \widehat{M}_{TL}, x^j \models p$ and hence $\langle p, c\rangle$ temporally describes $\langle\{t_i : j \leq i\}, U|_{j+}\rangle$ via $\psi_{U,M}$ by induction. So we have that if there is a nonnegative integer $h < \Lambda(x)$ such that $\widehat{M}_{TL}, x^h \not\models q$, there is a nonnegative integer $j < h$ such that $\langle p, c\rangle$ temporally describes $\langle\{t_i : j \leq i\}, U|_{j+}\rangle$ via $\psi_{U,M}$ and for every nonnegative integer $l \leq j, \langle q, c\rangle$ temporally describes $\langle\{t_i : l \leq i\}, U|_{l+}\rangle$ via $\psi_{U,M}$. Therefore, $\langle\Phi, c\rangle$ temporally describes $\langle\{t_i\}, U\rangle$ via $\psi_{U,M}$.

If $\Phi =$ (next $p$), then we must show that if $0 = 2J$ or $\langle p, c\rangle$ temporally describes $\langle\{t_i : 1 \leq i\}, U|_{1+}\rangle$ via $\psi_{U,M}$ then $\Lambda(x) = 1$ or $\langle x^1, c\rangle$ qualitatively describes $\langle\{t_i : 1 \leq i\}, U|_{1+}\rangle$. This follows by induction and the use of Lemma 3.

If $\Phi =$ (strong-next $p$), then we must show that if $0 < 2J$ and $\langle p, c\rangle$ temporally describes $\langle\{t_i : 1 \leq i\}, U|_{1+}\rangle$ via $\psi_{U,M}$ then $\Lambda(x) > 1$ and $\langle x^1, c\rangle$ qualitatively describes $\langle\{t_i : 1 \leq i\}, U|_{1+}\rangle$. Once again, this follows from the induction hypothesis and Lemma 3.

□

## 5.2 Main Theorem for Universal Formulas

This section contains the first theorem that is useful in applications. As a consequence of this theorem, the user of the TL and QSIM system may prove temporal statements about dynamical systems. First, the user constructs a QDE, $C$, and uses QSIM to generate a closed tree, $M$. Then the user may use TL to check if a universal formula, $\Phi$, is modeled by $\widehat{M}_{TL}$. If it is, then the user has proved that $\Phi'$ describes the solution to any differential equation which abstracts to $C$.

**Theorem 3** *Let $\Phi$ be a universal path formula in CTL*. Let $U$ be a solution to some ODE which abstracts to the QDE, $C$. Suppose QSIM generates the closed tree $M$ from $\langle C, I\rangle$. If $M$ is determined with respect to every proposition in $\Phi$ and $\widehat{M}_{TL}, s \models$* `(necessarily Φ)`*, then there is a specification $c$ of the landmarks in $\Phi$ such that $\langle \Phi', c\rangle$ temporally describes some splitting $\langle \{t_i\}, U\rangle$ of $U$ via $\psi_{U,M}$.*

**Proof:** Let $\Phi$ be a universal path formula and $M$ a closed QSIM behavior tree which is determined with respect to the propositions in $\Phi$. Suppose $\widehat{M}_{TL}, s \models$ `(necessarily Φ)`. Then for every fullpath $x$ starting at $s$, $\widehat{M}_{TL}, x \models \Phi'$ by Lemma 2. By the Guaranteed Coverage Theorem, we know that for every set $U$ which is the solution to an ODE which abstracts to $C$, there is a fullpath $y_U$ in $\widehat{M}_{TL}$ starting at $s$ and a specification, $\langle y_U, c\rangle$, of $y_U$ such that $\langle y_U, c\rangle$ qualitatively describes some splitting $\langle \{t_i\}, U\rangle$ of $U$. Since $M$ is determined with respect to the propositions in $\Phi$, and $\widehat{M}_{TL}, y_U \models \Phi'$, we conclude that $\langle \Phi', c\rangle$ temporally describes $\langle \{t_i\}, U\rangle$ via $\psi_{U,M}$ by Theorem 2.

□

This theorem allows the TL user to enter formulas with nested quantifiers. The conclusion of the theorem states that the perfect path formula related to the universal formula describes the solutions to the equations. Quantifier nesting is irrelevant, as far as the conclusions of this theorem are concerned. While there are situations in which nested quantifiers are useful, such as gaining insight into some detail of the QSIM structure, $\widehat{M}$, these applications do not rely on the main point of our theorems: the relation between the QSIM prediction and the underlying dynamical system.

The fact that nested quantifiers are not useful in applications involving proving temporal statements about dynamic systems is fortunate because of an important detail in the implementation of our model checker. For formulas with nested quantifiers, TL treats cycles as if they were weak-matches and thus follows all branches: i.e. it assumes that the structure is fusion closed. This is acceptable if nested quantifiers are only used when information is being sought about the underlying structure of the QSIM structure, $\widehat{M}$, or about the reasonable functions represented by trees generated using weak-match cycle detection.

Even in the latter case, the user who wishes to use this theorem may as well use a perfect path formula preceded by the `necessarily` quantifier.

There is a second important detail in the implementation which bears discussion at this point. It has been mentioned that the numeric information which QSIM derives about landmarks may vary across behaviors. QSIM keeps track of numeric information with respect to QSIM behaviors, not with respect to states. This fact makes a more restricted form of Theorem 3 desirable.

Since Theorem 3 mentions only a state and not a fullpath, the implementation cannot tell which of the original behaviors it is on when it is checking a state. Thus, Theorem 3 does not provide the theoretical backing for handling numeric information.

The issues of the proper handling of formulas without nested quantifiers on structures with strong-match cycles and the handling of propositions involving numeric information are addressed carefully in Section 5.3.

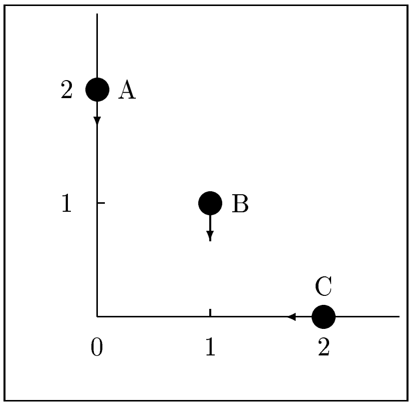
## 5.3 Numeric Queries

If there are nested quantifiers, then the implementation does not explicitly represent the fullpath which is currently being checked. To check a proposition involving numeric information (such as `in-range`) we must know which fullpath the state being checked in and that fullpath must be associated with a particular QSIM behavior in $M$ so that numeric information can be retrieved with respect to that QSIM behavior. The latter problem is solved by using the function $z$ defined in Section 3.1 since, in trees which contain numeric information, the cycle matching must be such that there are exactly as many rooted fullpaths as there are QSIM behaviors. The former problem is solved by restricting the syntax of the formulas in Theorem 3. This is exactly what the corollary in this section accomplishes.

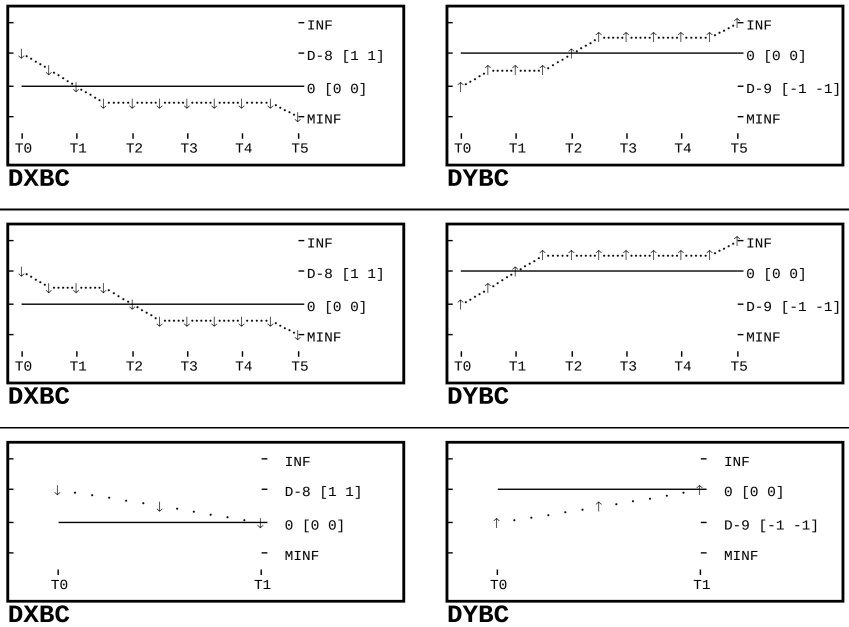
In order to illustrate the need for the corollary, we will construct a simple example with numeric information. Three billiard balls [17] start to move with constant velocities and initial positions shown in Figure 2(a). The QSIM QDE model for this scenario provides quantity spaces for position, velocity, and acceleration in the $x$ and $y$ directions, and constraints for constant-velocity motion. Collisions are detected when the differences in $x$ and $y$ positions of two balls are simultaneously zero.

When there is partial quantitative information about the speeds of the balls—`A` and `B` have velocity $-2$ units/second, and the velocity, `Cxp`, of `C` is some constant within the interval $[-3.5, -1.5]$ in units/second—QSIM predicts three possible behaviors, corresponding to `C` passing ahead of `B`, passing behind `B`, and colliding with `B` (Figure 2(b)). In case `C` collides with `B`, the collision takes place at $t = 0.5$ seconds. There is no possibility of `C` colliding with `A`.

We chose to deal with this amount of information because it illustrated our point without much complexity. Naturally, if the user had more or less




(a) Three balls on a billiard table, with initial positions and velocities.




(b) Given incomplete knowledge of the speed of ball C, QSIM predicts that C may pass ahead of B, behind B, or collide with B at $t = 0.5$.

Figure 2: Predicting behaviors of a real-time system.

knowledge about the conditions on the system, another QDE and initial state could be constructed.

Now, consider the value of horizontal velocity, `Cxp`, of `C` in the first state. We know that it is a real number between $-3.5$ and $-1.5$ and that it is constant. In the third state of the third behavior, we know that `Cxp` is equal to $-2$. In the third state of the second behavior, we know that `Cxp` is greater than $-2$. In the third state of the first behavior, we know that `Cxp` is less than $-2$. But since `Cxp` is a constant, its value over each behavior does not vary, therefore, its value at the first state (which all three behaviors share) depends on which behavior we are "in". QSIM must store the quantitative information about the values of landmarks not at each state but at each behavior.

In light of this discussion, the following corollary to Theorem 3 is seen to be necessary at least for situations in which numerical information is to be used.

**Corollary 1** *Let $U$ be a solution to some ODE which abstracts to the QDE, $C$. Suppose QSIM generates the closed tree $M$ from $\langle C, I\rangle$. Let $\Phi$ be a perfect path formula in CTL*. If $\widehat{M}_{TL}, I \models$* `(necessarily` $\Phi$`)`*, then for every rooted fullpath $x \in X^r$ there is a specification, $\langle x, c\rangle$, of $x$ such that $\langle \Phi, c\rangle$ temporally describes some splitting $\langle \{t_i\}, U\rangle$ of $U$ via $\psi_{U,M}$.*

This is a direct consequence of Theorem 3. The important difference between this and Theorem 3 is that this corollary can specifically mention the specification of the rooted fullpath since there are no nested quantifiers. In the case of QSIM structures involving numbers, each fullpath is associated with exactly one QSIM behavior, so quantitative information is available to the model checker. In the implementation, this allows us to consider numeric information on each rooted fullpath separately.

This corollary also provides the theoretical backing for the proper handling of structures with strong-match cycles even in the absence of quantitative information. Because QSIM structures with strong-match cycles are not necessarily fusion closed, the TL implementation requires that formulas querying such structures must have no nested quantifiers. This enables the checker to enumerate the rooted fullpaths so that it does not follow paths which are not represented by the QSIM tree. (See Shults' reformalization of the QSIM framework [18] for a discussion of this.)

## 5.4 Completeness Results

Given a closed QSIM structure, $\widehat{M}$, generated from $\langle C, I\rangle$, we sometimes would like to know whether there is any ODE, $F$, which abstracts to $C$ whose solution is described by some given temporal formula. In other words, we would like to have a similar theorem for formulas which are existential instead of universal. In order to do this, the QSIM tree must be closed, determined with respect to the propositions in $\Phi$ and satisfy the following completeness condition.

**Definition 5** *We call a closed QSIM behavior tree, $M$,* complete *if for every rooted fullpath, $x$ in $\widehat{M}$ there is an ODE, $F$ which abstracts to the input QDE and a splitting of the solution to $F'$ is qualitatively described by some specification of $x$.*

One way to check for the completeness of a tree is to prove, either mathematically or by numeric simulation, that there is a reasonable, extended real valued solution corresponding to each fullpath in the structure represented by the tree.

**Theorem 4** *Let $\Phi$ be a perfect path formula in CTL*. Let $M$ be a closed, complete QSIM behavior tree generated from the QDE and initial state $\langle C, I \rangle$ and determined with respect to the propositions in $\Phi$. If $\widehat{M}_{TL}, s \models$ `(possibly` $\Phi$`)`, then there is an ODE, $F$, such that $F'$ has solution $U$ and abstracts to $C$ and there is a specification of $\Phi$ which temporally describes some splitting of $U$ via $\psi_{U,M}$.*

**Proof:** Let $\Phi$ be a perfect path formula in CTL*, $M$ a closed, complete QSIM behavior tree generated by the QDE and initial state $\langle C, I \rangle$. Suppose that $M$ is determined with respect to the propositions in $\Phi$ and $\widehat{M}_{TL}, s \models$ `(possibly` $\Phi$`)`.

We want to show that there is an ODE, $F$, such that $F'$ has solution $U$ and abstracts to $C$ and there is a specification $\langle \Phi, c \rangle$ which temporally describes a splitting of $U$ via $\psi_{U,M}$.

We know from the semantics of CTL* that there is a fullpath $x$ in $\widehat{M}_{TL}$ such that $\widehat{M}_{TL}, x \models \Phi$. Because $M$ is complete, we know that there is a set of reasonable, extended real valued functions $U = \{u_i : 1 \leq i \leq n\}$ such that $U$ is a solution to some ODE which abstracts to $C$ and $\langle x, c \rangle$ qualitatively describes a splitting of $U$ for some specification $\langle x, c \rangle$ of $x$.

Therefore, $\langle \Phi, c \rangle$ temporally describes this fixed splitting of $U$ via $\psi_{U,M}$ by Theorem 2.

□

If the QSIM tree is not known to be complete, we still have the following.

**Corollary 2** *Let $\Phi$ be a perfect path formula in CTL*. Let $M$ be a closed, QSIM behavior tree determined with respect to the propositions in $\Phi$. If $\widehat{M}_{TL}, x \models \Phi$ and there is a set of reasonable, extended real valued functions, $U$, such that the specification $\langle x, c \rangle$ of $x$ qualitatively describes some splitting of $U$, then $\langle \Phi, c \rangle$ temporally describes that splitting via $\psi_{U,M}$.*

The $U$ mentioned in this corollary is not necessarily a solution to and ODE which abstracts to the QDE which was used to generate $M$.

# 6 Applications of CTL* and QSIM

TL is the name of a CTL* model-checker customized for use with QSIM. The current implementation replaces the experimental versions described and used in previous publications [12, 14]. The underlying model-checking algorithm is that of Bhat, Cleaveland and Grumberg [3]. Bhat, Cleaveland and Grumberg prove that this algorithm has the same complexity as the best known global algorithms for both CTL* and CTL. This algorithm has the added advantage of being "on-the-fly" rather than "global"; i.e. it is possible for the algorithm to halt with the correct answer without constructing the entire exponentially-large structure required to check some formulas in CTL*.

Temporal reasoning may be useful any time QSIM is used. QSIM has been used to simulate controllers, human organs and disease, abstract and real physical systems, electrical circuits, population dynamics, chemical reactions, etc. [13]

TL can be used to prove that a QSIM tree is closed with the following query

```
(TL R (necessarily
        (eventually (or (status quiescent)
                        (status cycle)
                        (status transition)))))
```

where `R` is the root of the tree. (See Appendix A for an explanation of arguments to the `status` proposition other than `quiescent`.)

TL automatically reports when an atomic proposition is checked on a state in which that proposition is not determined.

## 6.1 Examples

First, we demonstrate the use of TL to ask and answer questions about some simple models: the undamped oscillator, whose behavior tree (Figure 3) is rooted in the initial state `SS`; and the damped oscillator, whose behavior tree (Figure 4) is rooted in the state `DS`.

**Undamped Oscillator** The simple spring conserves energy, so all behaviors end in cycles, as shown by the behavior tree in Figure 3. Therefore, the closedness query would return T. The three behaviors differ according to whether the amplitude of the oscillation passes a predefined landmark value, `X*`. The queries shown demonstrate that the solution to any ODE consistent with the QDE in Figure 3 never becomes quiescent, always reaches a cycle state, and necessarily has an infinite sequence of events crossing $x = 0$ in opposite directions. (Since the variable `X` can have only one qualitative value in a state, the last two formulas below are equivalent.)

```
(TL SS (necessarily
         (always (not (status quiescent)))))
```

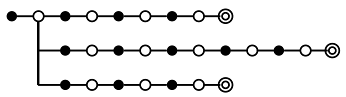
Figure 3: QSIM input and output for undamped spring

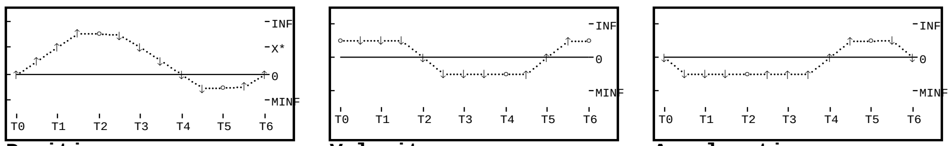
$$\ddot{x} + f(x) = 0$$

where $f \in M^+(x)$.

```
(define-QDE Spring-for-TL
  (quantity-spaces
   (A      (minf 0 inf)      "Acceleration")
   (V      (minf 0 inf)      "Velocity")
   (X      (minf 0 X* inf)  "Position"))
  (constraints
   ((d/dt X V))
   ((d/dt V A))
   ((M- A X)         (0 0) (minf inf) (inf minf))))

(defun spring-envisionment ()
  (setq SS (make-new-state :from-qde Spring-for-TL
                           :sim (make-sim :no-new-landmarks '(X V A)
                                          :cycle-detection :weak
                                          :state-limit 100)
                           :assert-values '((X (0 nil))
                                            (V ((0 inf) nil)))))
  (qsim SS)
  (qsim-display SS))
```






**Position** **Velocity** **Acceleration**

Each path in the behavior tree ends in a cycle state. Each cycle state matches the initial state. The three behaviors are distinguished by their amplitude. The second behavior is shown.

```
  => T

(TL SS (necessarily (eventually (status cycle))))
  => T

(TL SS (necessarily (and (infinitely-often (qval X (0 inc)))
                         (infinitely-often (qval X (0 dec))))))
  => T

(TL SS (necessarily
         (infinitely-often
            (before (qval X (0 dec))
                    (qval X (0 inc))))))
  => T
```

Since the simple spring tree is closed and determined, we have shown that every reasonable solution to an ODE which abstracts to the QDE in Figure 3 has a splitting which is temporally described by a specification of the perfect path formula associated with each of the formulas above.

We could rewrite the QDE in various ways to make the tree complete. Simply removing the extraneous landmark in X would suffice. This would produce a single behavior. The next example produces a complete tree.

**Damped Oscillator** The damped spring loses energy. The first behavior in the behavior tree in Figure 4 ends in a cycle representing a decreasing oscillation. The second two are partial cycles followed by "nodal" (i.e. over- or critically-damped) convergence to quiescent states at the origin. These qualitative behaviors have specifications which qualitatively describe real trajectories of nonlinear instances of the QDE. Since weak-match cycles were detected, this finite behavior tree represents a structure with infinitely many rooted fullpaths, oscillating a finite number of half-cycles around the origin before "nodal" convergence and a single rooted fullpath which never becomes quiescent. TL determines that each of the universal questions asked about the simple spring behavior tree above is false of the damped spring, but the corresponding existential statements are true.

```
(TL DS (possibly (always (not (status quiescent)))))
  => T

(TL DS (possibly (eventually (status cycle))))
  => T

(TL DS (possibly (eventually (status quiescent))))
  => T

(TL DS (possibly (and (infinitely-often (qval x (0 inc)))
```

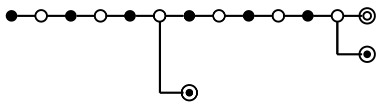
Figure 4: QSIM input and output for damped spring

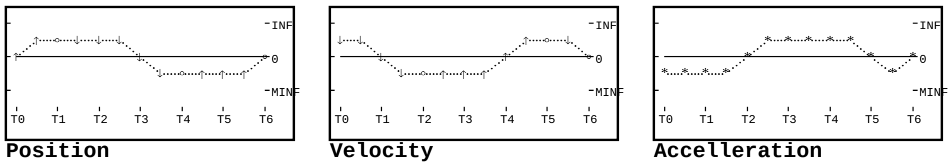
$$\ddot{x} + g(\dot{x}) + f(x) = 0$$

where $f$ and $g$ are in $M^{+}(x)$.

```
(define-QDE DSpring-for-TL
  (quantity-spaces
   (x     (minf 0 inf)   "Position")
   (v     (minf 0 inf)   "Velocity")
   (a     (minf 0 inf)   "Acceleration")
   (ff    (minf 0 inf)   "Fluid friction")
   (fs    (minf 0 inf)   "Spring force"))
  (constraints
   ((d/dt x v))
   ((d/dt v a))
   ((m- x fs)      (0 0)   (minf inf)  (inf minf))
   ((m- v ff)      (0 0)   (minf inf)  (inf minf))
   ((add fs ff a))))

(defun dspring-envisionment ()
  (setq DS (make-new-state
               :from-qde DSpring-for-TL
               :sim (make-sim :no-new-landmarks '(x v a ff fs)
                              :ignore-qdirs '(a)
                              :cycle-detection :weak
                              :state-limit 200)
               :assert-values '((x (0 nil)) (v ((0 inf) nil)))))
  (qsim DS)
  (qsim-display DS))
```






The first behavior in the tree ends in a cycle state which matches the root. The difference between the second two states is the direction from which they approach quiescence. The second behavior is shown.

```
                    (infinitely-often (qval x (0 dec))))))
  => T

(TL DS (possibly
         (infinitely-often
           (before (qval x (0 dec))
                   (qval x (0 inc))))))
  => T

(TL DS (necessarily
         (always (possibly (eventually (status quiescent))))))
  => T

(TL DS (necessarily
         (always (implies (not (status quiescent))
                   (possibly (always (not (status quiescent))))))))
  => T
```

The damped spring structure is complete, since there are nonlinear choices for the two monotonic functions in the model that give "spiral in" behavior away from the origin, followed by "nodal" behavior close to the origin. If both monotonic functions are linear, of course, the only possibilities are pure "nodal" and pure "spiral in" behavior. Therefore, we have proved that for each of the first five formulas above, there is a set of functions which is a solution to an ODE abstracted from the QDE given in Figure 4 and is temporally described via $\psi_{U,M}$ by a specification of the perfect path formula corresponding to the CTL* formula.

The last two formulas say that, no matter how many oscillations you've seen so far, it is always possible that (a) the behavior could terminate with nodal convergence to a quiescent state, and (b) the behavior could go on oscillating forever. Since the last two formulas are not universal, Theorem 3 gives no information. These two formulas are used for the purpose of discovering features of the QSIM structure, $\widehat{M}$, and not for proving properties of dynamical systems.

**A Quantitative Example** Now we reconsider the billiards example in order to show the use of quantitative information in QSIM QDEs and the resulting proofs that TL provides for time-critical systems. Refer to Figure 2 and the description of the system given in Section 5.3.

With only qualitative information about the balls' positions and speeds, QSIM gives a closed tree with 55 different behaviors, representing the different orders in which balls can collide, pass each other's positions, or reach infinity. With complete quantitative information, specifying identical speeds of 2 position units per second (in the indicated directions), QSIM predicts a single behavior in which balls `B` and `C` collide at $t = 0.5$ seconds.

The queries given below were checked on the tree with the information described in Section 5.3—A and B have velocity $-2$ units/second, and the velocity, Cxp, of C is some constant within the interval $[-3.5, -1.5]$ in units/second.

Because the predicted tree of behaviors is complete, we can draw conclusions from the answers to each of the following queries.

```
(TL SS (necessarily
         (always (implies (and (in-range dxBC (0 0))
                               (in-range dyBC (0 0)))
                   (in-range time (.5 .5))))))
  => T

(TL SS (possibly
         (eventually (and (in-range dxBC (0 0))
                          (in-range dyBC (0 0))
                          (in-range time (.5 .5))))))
  => T

(TL SS (necessarily
         (always (not (and (in-range dxAC (0 0))
                           (in-range dyAC (0 0)))))))
  => T
```

The first TL query proves that in the solution to any ODE which abstracts to the QDE if B and C collide then it happens at time 0.5. The second query proves (by Theorem 4) that there is a solution to an ODE consistent with the given QDE in which B and C do collide at time 0.5. The third query shows that A and C cannot collide in any ODE consistent with the given QDE. This is so because B blocks C in the behavior in which C might hit A.

## 6.2 TL as a Debugging Tool for QSIM Models

Because QSIM is not complete in general, a QSIM behavior tree may contain paths which do not correspond to real behaviors. Therefore, the truth of certain CTL* statements (e.g. those beginning with the quantifier possibly), do not imply the truth of the corresponding statement in an actual behavior. This provides an opportunity for a tool such as TL to be used to find such paths. If the QSIM user knows that a certain sequence of events cannot occur in a real behavior, he can use TL to find out if that sequence of events occurs in any of the paths in the QSIM behavior tree. The user can have TL print information which will isolate the path on which the spurious behavior occurs. Also, as in the damped spring example, nested quantifiers can be used to gain insight into some interesting structures of the represented QSIM structure.

The program can be and has been used on terminals which do not support the graphical display of QSIM behavior trees. In these circumstances, the user can learn everything he may need to know about a QSIM behavior tree by evaluating a few carefully chosen CTL* statements.

### 6.3 Proving Properties of Controllers

Kuipers & Åström [12] have used TL and QSIM to prove properties of heterogeneous control laws. A heterogeneous controller is a nonlinear controller created by the composition of local control laws appropriate to different, possibly overlapping, operating regions. Such a controller can be created in the presence of incomplete knowledge of the structure of the system, the boundaries of the operating regions, or even the control action to take. A heterogeneous control law can be analyzed, even in the presence of incomplete knowledge, by representing it as a qualitative differential equation and using qualitative simulation to predict the set of possible behaviors of the system. By expressing the desired guarantee as a statement in CTL*, the validity of the guarantee can be automatically checked against the set of possible behaviors. Kuipers & Åström [12] demonstrate the design of heterogeneous controllers, and prove certain useful properties, first for a simple level controller for a water tank, and second for a highly nonlinear chemical reactor.

Evi Gazi and Lyle Ungar also use TL to prove properties of models of chemical reaction controllers [6, 7].

There are three programs—Q2 [11], Q3 [2] and NSIM [9]—which extend QSIM to take advantage of numeric information, to prune spurious behaviors and to derive numeric bounds on landmark values and time-points. The program TL is easily applied to the behavior trees output by these QSIM extensions which use quantitative bounding information and produce quantitative bounds on the predictions. For these applications we use the propositional part of the language with the numeric propositions to include numerical information in the state propositions. These propositions allow TL to prove time-critical properties of models of a system, even in the face of incomplete knowledge.

## 7 Relation to other Work

The results described in this paper are related to other work done in the fields of temporal-logic model-checking and simulation and control.

Probably the most work in temporal-logic model-checking has been done in applications of CTL and CTL* to computer processes such as parallel computing [5, 15]. More closely related work has been done by Moon, *et al* [16] who checked statements in CTL against state transition graphs in discrete-time systems generated from programmable logic controller ladder diagrams. Their specific application was to chemical process control. TL makes it possible to apply a more complex temporal logic (CTL*) to continuous-time control systems, and indeed to dynamical systems in general.

Alur and Henzinger [1] use a logic called Metric Temporal Logic (MTL) to check properties of discrete event systems. Metric Temporal Logic is, strictly speaking, not as expressive as CTL*. However, it integrates time information at a higher level of the language, therefore it is easy to express some statements in MTL which are difficult to express in CTL*.

Jahanian [8] modeled real time systems in the Modechart language. Statements in Real Time Logic were checked against a Modechart model. Real Time Logic is undecidable in general but certain classes of statements are shown to be decidable. Model-checking CTL* is decidable [5]. However, Real Time Logic is especially suited for expressing statements which are useful in time-critical systems, whereas some such statements are more difficult to make in CTL*.

Other systems exist which allow temporal-logic sentences to be checked against a structure representing discrete event systems. TL makes a formal connection between continuous dynamical systems and time-critical temporal-logic model-checking.

# 8 Conclusion

TL implements a method for using modal- and temporal-logic formulas to prove properties of the behavior of a continuous physical system even with an incomplete, qualitative or semi-quantitative description. If the user can describe a physical system in terms of a set of qualitative constraints, then by using QSIM and TL, he can prove theorems about the behavior of any reasonable, extended real valued function consistent with those constraints. This applies even to systems with time-critical requirements. This provides a meaningful and sound interpretation for the phrase, "proof by simulation."

This link between logic-based and simulation-based inference methods will support a variety of hybrid reasoning techniques that could be of substantial value for the design and validation of continuous and piecewise-continuous systems.

## Acknowledgments

QSIM and TL are available for 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 report was prepared by the University of Texas at Austin as an account of work sponsored in part by the Electric Power Research Institute, Inc. (EPRI). Neither EPRI, members of EPRI, the University of Texas at Austin, nor any person acting on their behalf: (a) makes any warranty, express or implied, with respect to the use of any information, apparatus, method, or process disclosed in this report or that such use may not infringe privately owned rights; or (b) assumes any liabilities with respect to the use of, or for damages resulting from the use of, any information, apparatus, method, or process disclosed in this report.

We would like to thank Bhat, Grumberg and Cleaveland for their excellent article [3] and Rance Cleaveland for answering our questions about their algorithm. A difficult task in writing a paper which is intended for audiences in two somewhat disjoint disciplines, such as Qualitative Reasoning and Temporal Logic, is to use language which is comfortable to readers in both fields. Markus Kaltenbach and Richard Trefler from the Temporal Logic field gave helpful suggestions on the use of language and other useful information as well. We are also grateful to Dan Clancy, Bert Kay and Giorgio Brajnik from the Qualitative Reasoning field for their helpful discussions.

# A Extensions to the Propositional Language

The implementation, TL, of the language includes other propositions, some of which we describe in this appendix. In most cases, the added propositions are useful only to describe the predicted QSIM structure, and not to prove theorems about the underlying dynamical systems. Some of them can be included in the proof of the theorems but this inclusion would require distracting special treatment. This appendix discusses issues involved in adding new propositions to the language.

First we mention the "proposition" of the form `(funcall` $f$`)` where $f$ is a lisp function. This returns the value returned by the lisp function called with the state as its single argument. This is used mainly for side effects such as printing information about a state. Note that the proposition `funcall` cannot be considered as part of the logic when we talk about complexity, soundness or the main theorems. It is added for user-extensions and convenience and should be used with care.

The next proposition we mention illustrates the major issues involved in adding a proposition to the language. The proposition

$$\texttt{(contains-range } v \texttt{ (}n_1 \; n_2\texttt{))}$$

where $v$ is a variable name in the state $s$ and $n_1$ and $n_2$ are extended real numbers, is true when the numeric range, in which the number named by $v$ in the current behavior has been determined to lie, contains the interval $[n_1, n_2]$ as a subset.

If the theorems in the paper are going to be applied to a new proposition, then we must be able to include it in the proof of Lemma 4 in such a way that the induction step in the proof of Theorem 2 can be performed. Therefore, we must define what it means for a specification of this proposition to *describe temporally* a splitting of a set of reasonable, extended real valued functions and determine what is required for a state to be *determined* with respect to the proposition. Once Lemma 4 is proved for the proposition, the rest of the theorems will follow.

We say that a specification of the proposition

$$\texttt{(contains-range } \psi(u_i) \texttt{ (} n_1 \; n_2 \texttt{))}$$

temporally describes $\langle\{t_i\}, \{u_i : 1 \leq i \leq n\}\rangle$ via $\psi$ if and only if $[n_1, n_2] \subset u_i(D_0)$ where $\{D_k\}$ is the partition of the domain corresponding to the splitting.

In order for Lemma 4 to hold in general, for a QSIM state to be *determined* with respect to `(contains-range` $v$ `(`$n_1$ $n_2$`))`, the numeric range associated with a landmark of the variable $v$ in the state must be trivial (i.e. contain a single point).

Because the conditions on a state being determined with respect to `contains-range` are so restrictive, propositions such as `contains-range` are generally used more to gain information about the QSIM prediction than to prove theorems about the dynamical systems.

The `status` proposition can take any of the arguments {`quiescent`, `stable`, `unstable`, `transition`, `cycle`}. The proposition will be true when QSIM has determined the state to have the named property. A state has the `stable` property if it is quiescent and in stable equilibrium. A `transition` state is a terminal state in a path in which the value of one of the variables crosses a boundary of its range. A `cycle` state is a state which matches a previously generated state and whose successors are already represented in the tree. Other than `quiescent, stable` and `unstable` these properties have more to do with the QSIM interface than with the underlying functions being described. That is the reason we did not include a discussion of these properties in the discussion of the logic.

We define a QSIM state to be *determined* with respect to the propositions `(status stable)` and `(status unstable)` if it is not quiescent or if it has been determined to be stable. This is because QSIM may be incorrect when it determines a quiescent state to be `unstable`.

Additional propositional operators can also be added to allow the user to gain other information about QSIM states. For example, the newest release of QSIM produces "chatter-sink" states in order to express more succinctly the fact that certain variables may chatter indefinitely or, at some point, stop chattering. So, we could add the proposition `chatter-sink-p` to the language such that it is true of a state if and only if the state is a chatter-sink state. In this case, the proposition has no real meaning when translated to the domain of real-valued functions. Therefore, once again, it is used mainly to draw information about QSIM's output.

# References


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

[2] Daniel Berleant and Benjamin Kuipers. Combined qualitative and numerical simulation with Q3. In Boi Faltings and Peter Struss, editors, *Recent Advances in Qualitative Physics*. MIT Press, Cambridge, MA, 1992.

[3] Girish Bhat, Rance Cleaveland, and Orna Grumberg. Efficient on-the-fly model checking for CTL*. In *Proc. Conf. on Logic in Computer Science (LICS-95)*, 1995.

[4] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, August 1986.

[5] E. Allen Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 995–1072. Elsevier Science Pub. B. V./MIT Press, 1990.

[6] E. Gazi, W. D. Seider, L. H. Ungar, and B.J. Kuipers. Automatic analysis of monte-carlo simulations of dynamic chemical plants. In *Proceedings of ESCAPE 4*, 1995. (in press).

[7] E. Gazi, L. H. Ungar, and B. J. Kuipers. Temporal logic for summarizing monte-carlo simulation: an application to controller verification. In R. Shoureshi, editor, *Intelligent Control*. IEEE, 1996. (submitted).

[8] Farnam Jahanian and Douglas A. Stewart. A method for verifying properties of Modechart specifications. In *Proceedings of the Real-time Systems Symposium*, Huntsville, AL, December 1988.

[9] Herbert Kay and Benjamin Kuipers. Numerical behavior envelopes for qualitative models. In *Proc. 11th National Conf. on Artificial Intelligence*, pages 606–613, Cambridge, MA, 1993. AAAI/MIT Press.

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

[11] B. Kuipers and D. Berleant. Using incomplete quantitative knowledge in qualitative reasoning. *Proc. 7th National Conf. on Artificial Intelligence (AAAI-88)*, 1988.

[12] B. J. Kuipers and K. Åström. The composition and validation of heterogeneous control laws. *Automatica*, 30(2):233–249, 1994.

[13] Benjamin Kuipers. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. MIT Press, Cambridge, MA, 1994.

[14] Benjamin Kuipers and Benjamin Shults. Reasoning in logic about continuous systems. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR-94)*, San Mateo, CA, 1994. Morgan Kaufmann.

[15] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy thir linear specifications. In *Twelfth Annual ACM Symposium on Principles of Programming Languages*, pages 97–107, 1984.

[16] I. Moon, G. J. Powers, J. R. Burch, and E. M. Clarke. Automatic verification of sequential control systems using temporal logic. *AIChE Journal*, 38(1):67–75, 1992.

[17] Manny Rayner. On the applicability of nonmonotonic logic to formal reasoning in continuous time. *Artificial Intelligence*, 49:345–360, 1991.

[18] Benjamin Shults. Toward a reformalization of QSIM. University of Texas Artificial Intelligence Laboratory TR AI96-245, January 1996.