In J. Doyle, E. Sandewall, and P. Torasso, editors,
*Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR-94)*
Morgan Kaufmann, San Mateo, CA, 1994.

# Reasoning in Logic about Continuous Systems

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

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

## Abstract

An intelligent agent, reasoning symbolically in a continuous world, needs to infer properties of the behaviors of continuous systems. A qualitative simulator, such as QSIM, constructs a set of possible behaviors consistent with a qualitative differential equation (QDE) and initial state. This set of behaviors is expressed as a finite tree of qualitative state descriptions. In the case of QSIM, this set is guaranteed to contain the "actual" behavior under certain circumstances. We call this property the "soundness" of QSIM. The behavior tree can then be interpreted as a model for statements in a branching-time temporal logic such as Expressive Behavior Tree Logic (EBTL), which we introduce. Because QSIM is sound, validity of an EBTL proposition (`necessarily` $p$) implies the corresponding theorem about the dynamical system described by the QDE. 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 such commonsense statements as "what goes up (in a constant gravitational field) must come down", or to do such expert reasoning about dynamical systems as proving the stability of a nonlinear, heterogeneous controller.

## 1 INTRODUCTION

The world is infinite and continuous. A logical proof is finite and discrete. Nonetheless we want, and reasonably expect, to use logic to draw reliable conclusions about continuous behavior in the world.

A qualitative differential equation (QDE) is a symbolic description expressing a state of incomplete knowledge of the continuous world, and is thus an abstraction of an infinite set of ordinary differential equations. Qualitative simulation, using an algorithm such as QSIM [Kuipers, 86], predicts the set of possible behaviors consistent with a QDE and an initial state.

The QSIM algorithm generates a tree of qualitative states representing a branching-time description of the possible behaviors of the system being described. Qualitative simulation can be viewed as proving a theorem of a very specialized form:

$$QSIM \vdash QDE \wedge QState(t_0) \rightarrow or(QBeh_1, \ldots QBeh_n)$$

where $QDE$ is a qualitative differential equation, $QState(t_0)$ is a qualitative description of an initial state, and each $QBeh_i$ is a sequence of qualitative states. The QSIM Guaranteed Coverage Theorem states that this prediction describes all possible behaviors of all ordinary differential equations which are consistent with the given qualitative differential equation and initial state [Kuipers, 86]. The set of predictions may, however, include spurious predictions, those not corresponding to any real solution.

Building on the basic qualitative simulation algorithm, a variety of methods have been developed for filtering out additional classes of spurious behaviors, obtaining tractable predictions from a wider range of models while retaining the QSIM coverage guarantee. These methods include deeper types of mathematical analysis, application of partial quantitative information, appeal to carefully chosen additional assumptions, and change of the qualitative level of description [Kuipers, 93b].

Since the qualitative model and behavior tree are expressible in logic, we can show that a logical statement $\Phi$ follows from the model by showing that it follows from the behavior tree. We do this by showing that the behavior tree can serve as a logical model for $\Phi$.

Since the qualitative behavior tree is a branching-time description of temporal sequences, the appropriate language for such statements $\Phi$ is some form of modal temporal logic [Emerson, 90]. Temporal logic augments propositional logic with operators for temporal relations on time-varying truth-values, such as *sometimes*, *always*, *eventually*, and *until*. Modal logic adds operators for relations among truth-values in alternate possible worlds (i.e., alternate behaviors), such as *necessarily* and *possibly*.

We introduce Expressive Behavior Tree Logic (EBTL) as a tool for expressing statements about QSIM behavior trees, and hence about the continuous systems they describe. EBTL is a branching time temporal logic closely related to CTL and CTL* [Emerson, 90]. We describe an algorithm for checking the validity of an EBTL statement against a given QSIM behavior tree.

Based on the QSIM Guaranteed Coverage Theorem, we prove that for any EBTL statement $\Phi$ which is *universal* in a sense defined below, if $\Phi$ is true for the qualitative behavior tree predicted by QSIM, then the corresponding theorem holds for any ordinary differential equation consistent with the QDE that generated the QSIM behavior tree.

There are a number of applications of model-based reasoning that can profit from reliable inference over the set of all possible behaviors of a continuous system. Since applications – such as monitoring, diagnosis, and design – must often cope with conditions of incomplete knowledge, the ability to reason with all possible behaviors of a system described by a qualitative model is particularly valuable. A discussion of potential applications is provided in [Kuipers, 93a], and a specific application to the validation of heterogeneous controllers is provided in [Kuipers & Åström, 94] and briefly at the end of this paper.

# 2 BTL AND EBTL

Behavior Tree Logic (BTL) is a branching-time temporal logic. The theory of branching-time temporal logics is described in [Emerson, 90]. BTL is intended to be an extension and customization of Computational Tree Logic (CTL) to work with QSIM behavior trees. We are more interested in its more expressive extension, Expressive Behavior Tree Logic (EBTL), which is similar to CTL* [Emerson, 90]. Customization is necessary because CTL only applies to infinite temporal structures. A QSIM behavior tree is finite although it may be considered to represent an infinite tree. (In this paper, when we say "a QSIM behavior tree" we are referring to the actual output of the QSIM algorithm after finite time. Therefore, although the structure may grow without bound if QSIM were allowed to run indefinitely without memory constraints, a QSIM behavior tree in our discussion is necessarily finite. However, our theorems are applied to the often infinite trees represented by these finite structures.) Therefore, we have modified the logic so that it is applicable to finite QSIM behavior trees. Our definitions are only slight modifications (or complexifications) of Emerson's definitions of CTL and CTL*.

## 2.1 TERMINOLOGY AND NOTATION

In this section we define the structures related to the theory of Expressive Behavior Tree Logic. QSIM behavior trees are distinguished motivational examples of these structures but EBTL is applicable to a general class of behavior trees. A QSIM behavior tree [Kuipers, 86] can easily be compared to a temporal structure in the sense defined in [Emerson, 90]. This motivates the following general definition.

**Definition 1** *In (E)BTL, a* behavior tree *$M$ is an ordered triple $\langle S, R, L\rangle$ where*

*$S$ is a set of states,*

*$R$ is a binary relation on $S$, and*

*$L$ is a labeling which maps each state $s$ to an interpretation of all atomic proposition symbols in $s$.*

It is useful to view a behavior tree as a directed graph with node-set $S$ and arc-set $R$. Without loss of generality, we can assume that a behavior tree is a *tree* (thus the name), i.e. an acyclic directed graph in which each node has at most one predecessor and there is exactly one root. The root is the only node with the property that it has no predecessor and every node is accessible from it.

It may be helpful for the reader to beware of confusing the structures associated with the logic EBTL (of which QSIM behavior trees are examples) with QSIM structures. The logic EBTL may be applied to structures other than QSIM behavior trees. When we describe the application of EBTL to QSIM trees, many details such as the unwinding method for handling cycle pointers and the labeling of states will be made more explicit. We will try to make it clear when we are referring to the QSIM structures.

We let $\Lambda(x)$ denote the length of a finite ordered set $x$. A *behavior* $x = \langle s_0, s_1, s_2, \ldots\rangle$ in a behavior tree $M$ is any path in the behavior tree which either terminates at a state with no $R$-successors or is infinite. In case $x$ is of infinite length, we say $\Lambda(x) = \infty$. By a *path* $x = \langle s_0, s_1, s_2, \ldots\rangle$ we mean that for all $0 \leq i < \Lambda(x) - 1$, $\langle s_i, s_{i+1}\rangle \in R$. If $\Lambda(x) = \infty$ then by $i \leq \Lambda(x) - 1$ we mean $i$ is any nonnegative integer. Notice that the last state in a finite behavior $x = \langle s_0, s_1, s_2, \ldots\rangle$ is $s_{\Lambda(x)-1}$. In this paper we do not require $s_0$ to be the root of the behavior tree as is customary when referring to QSIM behaviors.

For simplicity we sometimes write $x \in M$ to mean that $x$ is a behavior in $M$. We say the behavior $x = \langle s_0, s_1, s_2, \ldots\rangle$ *starts at the state* $s_0$, and that $s_0$ *is the first state of* $x$. We will say that a behavior $x' \in M'$ *extends* a behavior $x = \langle s_0, s_1, \ldots, s_n\rangle$ in $M$ if the first $n+1$ states in $x'$ are $\langle s_0, s_1, \ldots, s_n\rangle$. When we speak of one tree $M$ being a subset of another tree $M'$ if every behavior in $M$ extends some behavior in $M'$. We call a behavior *rooted* if it starts at the root of its tree.

We now describe the behavior quantifiers and the basic temporal operators on propositions. We prefer to give the reader a rough description before the formal syntax and semantics are defined. Suppose some state $s_0$ and behavior $x$ starting at $s_0$ are given. The two behavior quantifiers are

> `(necessarily` $p$`)`, which is true if $p$ is true of *every* behavior starting with $s_0$, and
> `(possibly` $p$`)`, which is true if $p$ is true of *some* behavior starting at $s_0$.

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

> `(next` $p$`)` is true of the behavior $x$ if $p$ is true of the behavior 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. We may also call this relation `strong-until`, to distinguish it from `weak-until` to be defined below.

Let it be stressed that these descriptions are only given in order to give the reader a rough idea. The exact meaning of these operators comes from the formal definition of the syntax and semantics of the logic which are in subsequent sections. We use the following abbreviations:

`(eventually` $p$`)` $\equiv$
    `(strong-until true` $p$`)`
`(always` $p$`)` $\equiv$
    `(not (eventually (not` $p$`)))`
`(strict-precedes` $p$ $q$`)` $\equiv$
    `(and (not` $q$`)`
        `(strong-until (not (next` $q$`))` $p$`))`
`(weak-precedes` $p$ $q$`)` $\equiv$
    `(eventually (and` $p$ `(next (eventually` $q$`))))`
`(strong-precedes` $p$ $q$`)` $\equiv$
    `(and (strict-precedes` $p$ $q$`) (eventually` $q$`))`
`(weak-until` $p$ $q$`)` $\equiv$
    `(or (strong-until` $p$ $q$`) (always` $p$`))`
`(infinitely-often` $p$`)` $\equiv$
    `(always (eventually` $p$`))`
`(almost-everywhere` $p$`)` $\equiv$
    `(eventually (always` $p$`))`

These last two expressions seem to presume an infinite tree. The problem of reasoning about the infinite tree represented by a finite QSIM behavior tree is discussed later.

The statement `(strict-precedes` $p$ $q$`)` is true of a behavior if $p$ is true in some state in the behavior and $q$ is not true in any state previous to the first state in which $p$ is true. The statement `(weak-precedes` $p$ $q$`)` is true of a behavior if $q$ is true in some state in the behavior following some state in which $p$ is true. The statement `(strong-precedes` $p$ $q$`)` is true of a behavior if $q$ is true in some state in the behavior and $p$ is true in some state previous to the first state in which $q$ is true.

An expression in BTL is formed by an application of a behavior quantifier to a single one of the usual temporal operators: `always`, `strong-until`, `weak-until`, `next`, or `eventually`. EBTL is much more expressive because it allows boolean combinations and nestings of the behavior quantifiers and the usual temporal operators. Thus every statement in BTL is also a statement in EBTL, but "infinitely often" and "for all but finitely many" and other interesting statements can only be expressed in EBTL.

Our BTL is closely related to Emerson's CTL and our EBTL is closely related to Emerson's CTL*. The most noticeable difference is that BTL and EBTL are applicable to finite trees as well as infinite trees. Because (E)BTL is applicable to finite trees, the temporal operator `next` may seem ambiguous. This is so because some states do not have a successor. Therefore, we must distinguish between what is called `strong-next` and `weak-next`. The statement `(strong-next` $p$`)` is true of a behavior if the behavior has a second state and $p$ is true of that state. The statement `(weak-next` $p$`)` is true of a behavior if the behavior has no next state or if the behavior has a second state and $p$ is true of it. In our discussion, we consider `next` alone to mean `weak-next`. However, the language includes both terms and the user of our program may use both.

In the following two subsections we give the formal definitions of BTL and EBTL.

## 2.2 SYNTAX

The formal definitions of the syntax for the temporal operators and behavior quantifiers informally described above are given below. These definitions follow the treatment of CTL(*) in [Emerson, 90]. The definition of the syntax includes three state-formula generators, followed by one behavior-formula generator in the case of BTL, but followed by three behavior-formula generators in the case of EBTL. A state formula is a formula which is true or false of a state and a behavior formula is a formula which is true or false of a behavior. State formulæ in both BTL and EBTL are generated by rules (S1-S3) below. The behavior formulæ in BTL are generated by the rule (B0) below. The behavior formulæ in EBTL are generated by rules (B1-B3) below.

**Definition 2** *The syntax of EBTL is defined as follows.*

**(S1)** *Each atomic proposition* P *is a state formula,*

**(S2)** *if* $p$, $q$ *are state formulæ then so are* `(and` $p$ $q$`)` *and* `(not` $p$`)`*,*

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

**(B0)** *if* $p$, $q$ *are state formulæ then* `(next` $p$`)`, `(strong-next` $p$`)`, `(strong-until` $p$ $q$`)`, `(always` $p$`)`, `(weak-until` $p$ $q$`)` *and* `(eventually` $p$`)` *are behavior formulæ.*

**(B1)** *each state formula is also a behavior formula,*

**(B2)** *if* $p$, $q$ *are behavior formulæ then so are* `(and` $p$ $q$`)` *and* `(not` $p$`)`*,*

**(B3)** *if* $p$, $q$ *are behavior formulæ then so are* `(next` $p$`)`, `(strong-next` $p$`)`, `(strong-until` $p$ $q$`)`, `(always` $p$`)`, `(weak-until` $p$ $q$`)` *and* `(eventually` $p$`)`*.*

There are several things to notice here. First notice that (B0) is subsumed by (B1) and (B3). Therefore every expression in BTL is in EBTL. Also notice that the following formula is well-formed in both BTL and EBTL: `(strong-until (possibly (next` $p$`)) (necessarily (next` $p$`)))`. However, EBTL is strictly more expressive because, for example, `(necessarily (precedes` $p$ $q$`))` and `(possibly (not (weak-until` $p$ $q$`)))` are expressible in EBTL but not in BTL. We also allow the standard boolean abbreviations for `or` and `implies`.

### 2.3 SEMANTICS

The following notation is needed before the semantics of our logic can be defined. Given a behavior $x = \langle s_0, s_1, s_2, \ldots\rangle$, for $1 \le i \le \Lambda(x)-1$ we let $x^i$ denote the behavior $\langle s_i, s_{i+1}, s_{i+2}, \ldots\rangle$, which is the subbehavior of $x$ starting at $s_i$. I.e. it is the behavior obtained from $x$ by deleting from $x$ the first $i$ states.

Notice that if $\Lambda(x)$ is finite, then $x^i$ is not defined for $i > \Lambda(x) - 1$ and that $\Lambda(x^i) = \Lambda(x) - i$.

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 state formula $\Phi$ (respectively behavior formula $\Phi$) is true in the behavior tree $M$ at the state $s_0$ (respectively of the behavior $x$). Each item below gives the interpretation of the corresponding item in the syntax above.

**Definition 3** *If* $s_0$ *is a state in* $M$ *and* $x = \langle s_0, s_1, \ldots\rangle$ *is a behavior in* $M$ *starting at* $s_0$*, then we inductively define* $\models$ *as follows:*

**(S1)** $M, s_0 \models \mathrm{P}$ *if and only if* P *is true in* $L(s_0)$*,*

**(S2)** $M, s_0 \models$`(and` $p$ $q$`)` *if and only if* $M, s_0 \models p$ *and* $M, s_0 \models q$*,*
$M, s_0 \models$`(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 behavior* $y$ *in* $M$ *starting at* $s_0$*, such that* $M, y \models p$*,*
$M, s_0 \models$`(necessarily` $p$`)` *if and only if for every behavior* $y$ *in* $M$ *starting at* $s_0$*,* $M, y \models p$*.*

**(B1)** $M, x \models p$ *if and only if* $M, s_0 \models p$*,*

**(B2)** $M, x \models$`(and` $p$ $q$`)` *if and only if* $M, x \models p$ *and* $M, x \models q$*,*
$M, x \models$`(not` $p$`)` *if and only if it is not the case that* $M, x \models p$*,*

**(B3)** $M, x \models$`(strong-until` $p$ $q$`)` *if and only if there is a nonnegative integer* $i \le \Lambda(x) - 1$*, such that* $M, x^i \models q$ *and for every nonnegative integer* $j < i$*,* $M, x^j \models p$*,*
$M, x \models$`(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* $M, x^1 \models p$*,*
$M, x \models$`(weak-until` $p$ $q$`)` *if and only if for every nonnegative integer* $j \le \Lambda(x) - 1$*, if for every nonnegative integer* $k \le j$ *we have* $M, x^k \models$`(not` $q$`)`*, then* $M, x^j \models p$*,*
$M, x \models$`(always` $p$`)` *if and only if for every nonnegative integer* $j \le \Lambda(x) - 1$*,* $M, x^j \models p$*,*
$M, x \models$`(eventually` $p$`)` *if and only if there is a nonnegative integer* $j \le \Lambda(x) - 1$*, such that* $M, x^j \models p$*,*

The semantics of BTL formulæ are the same as those given above with (B3) giving the semantics of the formulæ given in (B0) of the definition of the syntax.

Now that the semantics are defined, the reader will notice that there are two definitions of the following operators: `weak-until`, `always` and `eventually`. We have given semantic definitions for these operators and we have also defined them as abbreviations of expressions involving `strong-until` and `next`. The proofs of the equivalence of these definitions are omitted because they are straight-forward but tedious manipulations of quantifiers, negation symbols, and boolean operators.

## 3 QSIM AND THE IMPLEMENTATION OF THE LOGIC

Here we consider how the logic is implemented and applied to QSIM. First, we define the relations that make finite QSIM behavior trees into possibly infinite trees. Second, we show exactly how QSIM behavior trees and the trees they represent are used as logical models for EBTL statements. Finally, we discuss the implementation of the program which checks the truth of statements in EBTL against a QSIM behavior tree. We call the program TL for "temporal logic".

### 3.1 QSIM AS A MODEL FOR (E)BTL

Qualitative simulation with QSIM produces a tree of *qualitative states*, linked by *successor* and *transition* relations.[1] A *QSIM behavior* is a path in the behavior tree, terminating at a leaf of the tree, but not necessarily starting at the root state. (This differs from normal usage.) Each state describes the *qualitative value* of each *variable* appearing in the QDE model. 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 value or in an open interval defined by two landmarks, and *qdir* is the sign of the derivative $v'$ of $v$. By considering the qualitative values of the variables at $s$, and the constraints in the QDE, QSIM is able to derive a number of properties of the state, including quiescence, stability, cycles, etc. Please see [Kuipers, 86, 94] for more detailed information on QSIM.

A QSIM behavior tree is made a logical model for statements in EBTL in the following way. A *QSIM behavior tree* $M$ is an ordered triple $\langle S, R, L\rangle$ where the set $S$ of states is the set of states in the output of the QSIM algorithm, the set $R$ is the union of the QSIM successor and transition relations, and the interpretation $L(s)$ is as follows.

For the sake of brevity, we consider only the atomic propositions associated with any QSIM state $s$ which are of one of the following forms:

`(status` *tag*`)` where *tag* is an element of $\{$`quiescent`, `stable`, `unstable`, `transition`, `cycle`$\}$. Such a proposition is true exactly when *tag* is a member of the QSIM structure `s.status` associated with the state $s$.

`(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`, `ign`$\}$. Such a proposition is true exactly when the value of $v$ in the state $s$ matches the description `(`*qmag qdir*`)`.

The expressiveness of the application of EBTL to QSIM could easily be increased without adding to the complexity by adding expressiveness to this propositional part of the language. In particular, we could allow propositional formulæ other than the two given above. For example, we could add the ability to compare the values of two variables, or to consider quantitative information about variable values.

By a *QSIM behavior* we mean a path in a QSIM behavior tree, not necessarily starting at the root state, such that the last node in the path has no $R$-successor. We call the state at which a behavior starts the *first state* of the behavior. In this paper, when we say "a QSIM behavior tree" we are referring to the finite output of the QSIM algorithm. That is to say, given a qualitative differential equation and allowed a finite amount of time to run, QSIM will return a finite tree. The finiteness of QSIM trees may seem to be a terrible limitation. For example, expressions such as "for all but finitely many" and "infinitely often" would apparently never be sensibly satisfied by a QSIM behavior tree. However, a QSIM behavior tree may *represent* an infinite behavior tree.

### 3.2 THE TREE REPRESENTED BY A QSIM BEHAVIOR TREE

QSIM has two ways of presenting a behavior over an infinite time-interval with a finite sequence of qualitative states. First, a fixed-point of a behavior is represented by a state with status `quiescent`. Second, repeated patterns in a behavior can be described by cycles. A *cycle state* in a QSIM behavior is one that matches a previously-generated state elsewhere in the behavior tree, so its successors are already represented by the successors of the previously-generated state. The user may select the state-matching criterion, and whether cycles must lie within a single behavior or may cross among behaviors. With respect to the tree $\widehat{M}$ *represented* by a QSIM behavior tree $M$, the expressions `(infinitely-often` $p$`)` and `(almost-everywhere` $p$`)` have exactly the desired meaning. The solution to the problem of reasoning about the infinite tree in finite time is discussed later.

**Definition 4** *The ordered pair $\langle s_i, s_j\rangle$ of states is in a* status-bound *relation if either of the following two conditions holds:*

*(1) The proposition* `(status quiescent)` *is true of $s_i$, and $s_i = s_j$ or*
*(2) the proposition* `(status cycle)` *is true of $s_i$, and $s_j$ is a successor of the previous state $s'$ in the tree such that $s' = s_i$.*

If the ordered pair $\langle s_1, s_2\rangle$ is an element of the set of status-bound relations, then we say that $\langle s_1, s_2\rangle$ is a *cycle relation* if $s_1 \neq s_2$.

**Definition 5 (Represented Tree)** *The possibly infinite tree $\widehat{M} = \langle \widehat{S}, \widehat{R}, \widehat{L}\rangle$,* represented by *a QSIM behavior tree $M = \langle S, R, L\rangle$, is the tree which results by*

[1] There is also a *completion* relation not discussed here, that holds between an incomplete state description and a complete one consistent with it. Handling this relation is a straight-forward extension of the methods discussed here.

*adding the status-bound relations to the set $R$. The set $\widehat{R}$ is the union of $R$ with the set of status-bound relations. The set $\widehat{S}$ is the union of $S$ and the new states which are generated first as second elements of status-bound relations and then by the unwinding process. (Cf. [Emerson, 90] for a precise definition of unwinding.) We will call the new states* copies *of the state in $S$ to which they correspond. Each new state inherits the interpretation $L(s)$ of its proposition symbols from the state of which it is a copy.*

### 3.3 CLOSED BEHAVIOR TREES

In the best case, every behavior in the tree returned by QSIM terminates with a quiescent or cycle state. We will call such a tree *closed*. 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 Guaranteed Coverage Theorem does not necessarily apply. If the behavior tree $M$ is not closed then it is possible that the actual behavior of the system is not represented in $\widehat{M}$.

Using the normal QSIM simulation style, creating new landmarks for critical values and applying a strong cycle-match criterion (all variables have identical landmark values), certain systems such as the damped spring have infinite behavior trees. In such cases, the QSIM algorithm cannot produce a closed behavior tree in finite time. However, by applying the *envisionment* simulation style (no new landmarks and weak cycle-match criterion), every qualitative model has a finite behavior tree. [Kuipers, 94] discusses this and a variety of methods for obtaining tractable behavior trees.

### 3.4 THE PARTIAL EXTENSION OF A BEHAVIOR TREE

Given an EBTL statement $\Phi$ to check against a QSIM behavior tree $M$, we can define a *partial extension* $\overline{M}(\Phi)$ of $M$,

$$M \subseteq \overline{M}(\Phi) \subseteq \widehat{M}$$

that is finite (where $\widehat{M}$ might not be) and enough larger than $M$ to make the truth value of $\widehat{M}, s_0 \models \Phi$ be the same as that of $\overline{M}(\Phi), s_0 \models \Phi$.

In section 3.6 we will prove that if the truth checker is given a QSIM behavior tree $M$ and a statement $\Phi$ in EBTL, then it returns the truth value of $\Phi$ regarding the generally larger tree $\widehat{M}$ represented by $M$. As we will see, this tree can be infinite and complex. We think of a statement in EBTL as a question which the user is asking about the given behavior tree. The user expects the program to respond with the truth value of $\widehat{M}, s_0 \models \Phi$, where $\widehat{M}$ is the possibly infinite tree represented by the QSIM tree $M$ and $s_0$ is the root of the tree. The program TL accomplishes this by constructing the partial extension of the given QSIM tree and checking the truth of the given expression on this larger yet still finite tree. The proof is accomplished by showing that the partial extension of the tree is large enough to decide the question $\Phi$.

The reader may find it helpful to examine Figure 1 for a motivation of the following definitions. The partial extension $\overline{M}(\Phi)$ of $M$ depends also on $\Phi$. It is constructed from $M$ and $\Phi$ by expanding $M$ according to the structure of nestings of `until` and `next` statements in $\Phi$.

Let us now give the needed definitions. Recall that a QSIM behavior tree $M$ is necessarily finite. The *until extent*, $x(\text{until})$, of a behavior $x$ in $M$ is a set of possibly truncated behaviors in $\widehat{M}$ which extend $x$. The addition of these longer behaviors enlarge $M$ exactly enough to answer properly any propositional `until` statement.

**Definition 6** *The* until extent $x(until)$ *of a finite behavior $x = \langle s_0, s_1, \ldots, s_n \rangle$ is the singleton set containing $x$ unless* `(status cycle)` *is true of $s_n$ in which case $x(until)$ is the set of paths $x'$ in $\widehat{M}$ extending $x$ but truncated at the first state $s \in x'$ at which the following property is satisfied:*

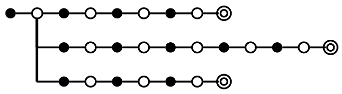
The Until Property: *$s$ is not in $x$ and either* `(status quiescent)` *is true of $s$ or* `(status cycle)` *is true of $s$ and $s$ is a copy of some previous state in $x'$.*

It is important to understand that the until extent of a behavior $x$ in a finite QSIM tree $M$ is a finite set of finite behaviors. To see this we need to recall two facts. First, QSIM behavior trees are finitely branching. Second, cycle states occur only at the terminal states of a QSIM behavior tree, thus there are only finitely many cycle states in a finite QSIM tree. If some $x' \in x(\text{until})$ were infinite, it would have to pass through infinitely many cycle states. Thus it would have to pass through one of them more than once, contradicting the Until Property. Since each behavior in $x(\text{until})$ is finite, and $M$ is finitely branching, $x(\text{until})$ must be finite.

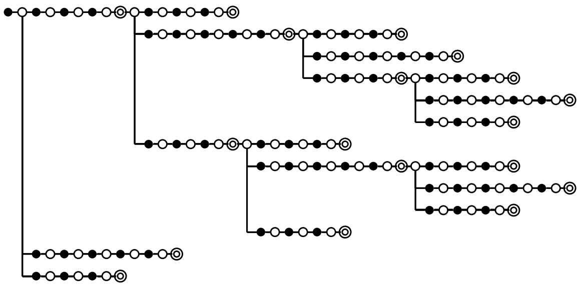
The *next extent*, $x(\text{next})$, of a behavior $x$ in $M$ is the set of possibly truncated behaviors in $\widehat{M}$ which are sufficiently extended to answer a propositional `next` question.

**Definition 7** *The* next extent $x(next)$ *of a finite behavior $x = \langle s_0, s_1, \ldots, s_n \rangle$ is the set of paths $x'$ in $\widehat{M}$ extending $x$ but truncated at the first state $s \in x'$ satisfying one of the following properties:* `(status cycle)` *is true of $s_n$ and $s$ satisfies the Until Property or* `(status quiescent)` *is true at $s$ and $s$ is not in $x$.*

A similar argument as the one given above shows that the next extent of a finite behavior in a finite behavior tree is a finite set of finite behaviors.



$M$before unwinding



$\overline{M}(\Phi)$after partial unwinding

Figure 1: Partial unwinding for $\Phi$ = `(until` $p$ $q$`)`, along first behavior only.

Each cycle state is expanded, stopping each branch at the second occurrence of a given cycle state. The double circles represent cycle states.

**Definition 8** *We define the* partial extension $\overline{M}(\Phi)$ *generated by a tree $M$ and an EBTL expression $\Phi$ recursively as follows:*

*If $\Phi$ is a proposition then $\overline{M}(\Phi) = M$,*
*if $\Phi$ is* `(and` $p$ $q$`)` *then $\overline{M}(\Phi)$ is the union of $\overline{M}(p)$ and $\overline{M}(q)$.*
*if $\Phi$ is* `(not` $p$`)`, `(possibly` $p$`)` *or* `(necessarily` $p$`)`, *then $\overline{M}(\Phi) = \overline{M}(p)$,*
*if $\Phi$ is* `(strong-until` $p$ $q$`)` *then $\overline{M}(\Phi)$ is the union over each behavior $x \in \overline{M}(p) \cup \overline{M}(q)$ of $x$(until), or*
*if $\Phi$ is* `(next` $p$`)`, *or* `(strong-next` $p$`)`, *then $\overline{M}(\Phi)$ is the union over each behavior $x \in \overline{M}(p)$ of $x$(next).*

Notice that the paths in the until and next extents of a behavior in $M$ are generally not behaviors in $M$ or $\widehat{M}$. They may, however, be behaviors in $\overline{M}(\Phi)$ for some $\Phi$.

Our implemented prover TL, given as inputs a QSIM tree $M$ and an EBTL expression $\Phi$, returns `true` if and only if $\widehat{M}, s \models \Phi$ where $s$ is the root state of $M$. We will prove that it is enough for the truth checker to examine $\overline{M}(\Phi)$, which in fact is what TL does. To be specific, we have defined the partial extension of a behavior tree generated by a QSIM behavior tree $M$ and an EBTL statement $\Phi$ to be at least as large as the largest tree generated by TL in the process of checking $\Phi$ on $M$. It, however, should be clear that this tree must be finite. This is true because each statement in EBTL is finite and each QSIM behavior tree is finite.

We will prove that given an EBTL expression $\Phi$ and a QSIM behavior tree $M$, TL correctly returns the truth or falsity of $\Phi$ in the behavior tree $\widehat{M}$ represented by $M$ in finite time.

We need a way of distinguishing, for a given subexpression $\Phi_0$ of $\Phi$, whether $\Phi_0$ is in the scope of a `necessarily` or of a `possibly` quantifier. The definition in the next subsection fulfills this need.

### 3.5 IMMEDIATE SCOPE

We recursively define an occurrence of $p$ being in the *immediate scope* of a behavior quantifier as follows:

**Definition 9 (Immediate Scope)**
*The occurrence of $p$ in* `(possibly` $p$`)` *is in the immediate scope of* `possibly`*.*
*If* `(and` $p$ $q$`)`, `(not` $p$`)`, `(strong-until` $p$ $q$`)`, `(next` $p$`)`, `(strong-next` $p$`)`, `(always` $p$`)`, *or* `(eventually` $p$`)` *occurs in the immediate scope of* `possibly` *then these occurrences of $p$ and $q$ are said to occur in the immediate scope of* `possibly`*.*
*The occurrence of any EBTL expression which can not be shown to be in the immediate scope of* `possibly` *by the above conditions is in the immediate scope of* `necessarily`*.*

Other temporal operators are treated as abbreviations of expressions involving the operators mentioned above.

Consider the following example of an EBTL expression:

```
(and (possibly
        (strong-until p (necessarily q)))
     (next (possibly r)))
```

The `and` and its arguments are in the immediate scope of `necessarily` as is the occurrence of $q$. Also `(possibly` $r$`)` is in the immediate scope of `necessarily`. However, the `strong-until` statement and its arguments are in the immediate scope of `possibly`.

### 3.6 CORRECTNESS OF THE IMPLEMENTATION

Now we come to the promised proof. Because TL examines the partial extension $\overline{M}(\Phi)$ of $M$, what we really need to prove is the following:

**Theorem 1** *If $M$ is a QSIM behavior tree with root $s$ and $\Phi$ is an EBTL state expression, then*

$$\widehat{M}, s \models \Phi \Leftrightarrow \overline{M}(\Phi), s \models \Phi.$$

The proof goes by induction on the structure of $\Phi$. If $\Phi$ is a proposition, then the theorems are obvious.

Also it is clear how to handle the booleans, i.e. we simply pass the proof on to their arguments.

If $\Phi$ is of the form (`necessarily` $p$) then we must really prove the following: $\widehat{M}, x \models p$ for all behaviors $x \in \widehat{M}$ if and only if $\overline{M}(p), x' \models p$ for all behaviors $x' \in \overline{M}(p)$. If $\Phi$ is of the form (`possibly` $p$) then we must prove the following: $\widehat{M}, x \models p$ for some behavior $x \in \widehat{M}$ if and only if $\overline{M}(p), x' \models p$ for some behavior $x' \in \overline{M}(p)$.

So the interesting parts of the proof will be when $\Phi$ begins with a temporal operator within the immediate scope of either `necessarily` or `possibly`. Theorem 2 takes care of the case when a `strong-until` statement occurs in the immediate scope of `necessarily`.

**Theorem 2** *If $\Phi$ is of the form* (`strong-until` $p$ $q$)*, then $\widehat{M}, x \models \Phi$ for all rooted behaviors $x \in \widehat{M}$ if and only if $\overline{M}(\Phi), x' \models \Phi$ for every rooted behavior $x' \in \overline{M}(\Phi)$.*

**Proof:** ($\Rightarrow$) Suppose $\widehat{M}, x \models \Phi$ for every behavior $x \in \widehat{M}$ starting at the root. Let $x'$ be a behavior in $\overline{M}(\Phi)$ starting at the root. Suppose for the sake of contradiction that for every nonnegative integer $k \leq \Lambda(x') - 1$, if $\overline{M}(\Phi), x'^k \models q$, then there is a number $l < k$ such that $\overline{M}(\Phi), x'^l \not\models p$. There is a behavior $x$ in $\widehat{M}$ which extends $x'$. The existence of this behavior in $\widehat{M}$ contradicts our hypothesis.

($\Leftarrow$) Now suppose that for every behavior $x' \in \overline{M}(\Phi)$ starting at the root

$$\overline{M}(\Phi), x' \models \Phi \tag{1}$$

Suppose for refutation that for every behavior $x$ in $\widehat{M}$ starting at the root, for all $i \leq \Lambda(x) - 1$ if $\widehat{M}, x^i \models q$, then there is a nonnegative integer $j < i$ such that $\widehat{M}, x^j \not\models p$. There are two cases to consider.

First we suppose there is a rooted behavior $x \in \widehat{M}$ such that for all $i \leq \Lambda(x) - 1, \widehat{M}, x^i \not\models q$. Let $x'$ be a behavior in $\overline{M}(\Phi)$ which $x$ extends. (I.e. cut off $x$ at its first state which satisfies the Until Property.) The existence of this $x' \in \overline{M}(\Phi)$ contradicts our hypothesis (1).

Now suppose that there is a rooted behavior $x \in \widehat{M}$ such that for every $i \leq \Lambda(x) - 1$ such that $\widehat{M}, x^i \models q$ there is a $j < i$ such that $\widehat{M}, x^j \not\models p$ and that such an $i$ exists. Choose $i$ to be the smallest number such that $\widehat{M}, x^i \models q$. Let $j < i$ be such that $\widehat{M}, x^j \not\models p$. Let $y$ denote the path from the root to the first state, $s_i$ in $x^i$. Suppose there is a state $s$ in $y$ which satisfies the Until Property. If such a state exists then the path $\langle s_0, s_1, \ldots, s \rangle$ is a behavior in $\overline{M}(\Phi)$. The existence of this behavior contradicts the hypothesis (1). If there is no state in $y$ satisfying the Until Property, then there is some behavior $x'$ in $\overline{M}(\Phi)$ which extends the path $y$. This behavior once again contradicts our hypothesis (1).

This completes the proof.

Theorem 3 takes care of the case when a `strong-until` statement occurs in the immediate scope of `possibly`.

**Theorem 3** *If $\Phi$ is of the form* (`strong-until` $p$ $q$)*, then there is a rooted behavior $x \in \widehat{M}$ such that $\widehat{M}, x \models \Phi$ if and only if there is a rooted behavior $x' \in \overline{M}(\Phi)$ such that $\overline{M}(\Phi), x' \models \Phi$.*

**Proof:** ($\Rightarrow$) First, suppose there is a rooted behavior, call it $x$, in

$\widehat{M}$ such that $\widehat{M}, x \models \Phi$. There is a path $x' \in \overline{M}(\Phi)$ which $x$ extends. Suppose for contradiction that for all nonnegative integers $k \leq \Lambda(x') - 1$ if $\overline{M}(\Phi), x'^k \models q$ then there is a nonnegative integer $l < k$ such that $\overline{M}(\Phi), x'^l \not\models p$.

If $\overline{M}(\Phi), x'^k \models q$ for some $k \leq \Lambda(x') - 1$ and $\overline{M}(\Phi), x'^l \not\models p$ for some $l < k$ then the same is true for $x \in \widehat{M}$ which is a contradiction.

If there is no $k \leq \Lambda(x') - 1$ such that $\overline{M}(\Phi), x'^k \models q$, then it must be the case that for every state $t \in x', \overline{M}(\Phi), t \models p$. Thus far, we have assumed nothing about the behavior $x \in \widehat{M}$ except that $\widehat{M}, x \models \Phi$. We know that for any such behavior there is a smallest number $i(x)$ such that $\widehat{M}, x^{i(x)} \models q$. Let $i$ be the smallest of all of the $i(x)$ ranging over behaviors $x$ for which $\widehat{M}, x \models \Phi$ and let $x$ now denote the behavior corresponding to $i$. If there is no state, $s$, in the path $y = \langle s_0, s_1, \ldots, s_i \rangle$ satisfying the Until Property, then there is a behavior $x' \in \overline{M}(\Phi)$ which extends $y$. In this case we are done becuase $\overline{M}(\Phi), x' \models \Phi$.

If there is a state in $y$ which satisfies the Until Property, then we let $s$ denote the first such state. Either (`status quiescent`) or (`status cycle`) must be true at $s$ (or $s$ has no successors). In the former case we are done becuase the path $\langle s_0, s_1, \ldots, s \rangle$ is a behavior in $\overline{M}(\Phi)$. In the latter case, we delete from $y$ the state of which $s$ is a copy and the states between it and $s$. What remains of the path $y$ is again a truncated path in $\widehat{M}$, but a shorter path, and $\widehat{M}, z \models \Phi$ for any path $z \in \widehat{M}$ which extends $y$. But this contradicts our assumption that the state $s_i \in x$ was the nearest state to the root satisfying these conditions.

($\Leftarrow$) Now suppose there is a behavior $x' \in \overline{M}(\Phi)$ such that $\overline{M}(\Phi), x' \models \Phi$. There is a behavior $x \in \widehat{M}$ which extends $x'$. Thus we are done.

This completes the proof.

Similar theorems follow for `next` expressions and the other temporal operators can be treated as abbreviations of these two.

Therefore, it is enough for the truth checker to examine only $\overline{M}(\Phi)$ when trying to check $\widehat{M}, s_0 \models \Phi$. Since the until (or next) extent of a behavior in a QSIM tree starting at any state is finite and each EBTL expression is finite, the truth checker will terminate with the correct answer.

## 4 THE MAIN THEOREM

Our main theorem states that, under appropriate hypotheses, the answer that TL gives to an EBTL statement concerning a QSIM behavior tree will be true of the solution to any differential equation consistent with the qualitative differential equation which produced the QSIM behavior tree.

Before we state our main theorem we need some notation, definitions and a lemma. We define the parity of a position in an EBTL expression as follows:

**Definition 10**

*The first operator in any EBTL expression given to TL is in a position of parity 0.*

*If* (`not` $p$) *occurs in a position of parity* $n \in \{0,1\}$, *then* $p$ *is in a position of parity* $n+1$ *(mod 2).*

*If* ($O$ $p$) *or* ($O$ $p$ $q$) *occurs in a position of parity* $n \in \{0,1\}$ *and* $O$ *is some temporal, boolean, or modal operator other than* `not`, *then* $p$ *and* $q$ *occur in positions of parity* $n$.

Recall that (`implies` $p$ $q$) is an abbreviation of (`not` (`and` $p$ (`not` $q$))) so if (`implies` $p$ $q$) occurs in a position of parity $n \in \{0,1\}$ then $p$ is in a position of parity $n+1$ (mod 2) and $q$ is in a position of parity $n$. This follows the use of "positive" and "negative" position in [Wang, 60].

**Definition 11** *An EBTL expression* $\Phi$ *is said to be* universal *if every occurence of the behavior quantifier* `possibly` *is in a position of parity 1 and every occurrence of the behavior quantifier* `necessarily` *is in a position of parity 0.*

With a little thought, the reader will see that if a formula is universal, then the truth checker should examine the entire tree in order to establish the truth of the formula. This is the motivation for the definition.

If $\Phi$ is a universal formula in EBTL, then $\Phi'$ denotes the linear-time behavior formula obtained from $\Phi$ by deleting all occurrences of the behavior quantifiers. For example, if

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

then

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

We are now ready to proceed to the details of our main theorem. We say a real-valued function, $u$, *satisfies* a given QSIM qualitative behavior description if the qualitative description of the function matches the given qualitative behavior. The following theorem is proved in [Kuipers, 86].

**Theorem 4 (Guaranteed Coverage)** *Let* $F = 0$ *be an ordinary differential equation with solution* $u$, *a real valued function. Let* $C$ *be a QDE with which* $F = 0$ *is consistent. Let* $M$ *be the QSIM behavior tree generated by the QSIM algorithm applied to* $C$. *If* $M$ *is closed, then* $u$ *satisfies some QSIM behavior* $x = \langle s_0, s_1, s_2, \ldots \rangle$ *in* $\widehat{M}$ *where* $s_0$ *is the root state of* $M$.

**Theorem 5 (The Main Theorem)** *Let* $\Phi$ *be a universal state formula in EBTL. Let* $u$ *and* $M$ *be as in the hypotheses of the Guaranteed Coverage Theorem. Let* $s_0$ *be the root state of* $M$. *If* $\overline{M}(\Phi), s_0 \models \Phi$, *then* $\Phi'$ *is true of the qualitative description of* $u$.

**Proof:** Suppose $\overline{M}(\Phi), s_0 \models \Phi$, as in the hypotheses of the theorem. By definition, $\Phi'$ is a behavior formula. For simplicity, let us start by replacing every occurence of the temporal operators `weak-until`, `always`, `precedes`, `strong-precedes`, `infinitely-often`, `almost-everywhere`, and `eventually` with expressions involving only the temporal operator `strong-until` and `next`. (Since $M$ is closed, it makes no difference whether we consider `next` to be strong or weak.) This is made possible by the abbreviations on page 3. So now $\Phi'$ is a behavior formula whose only temporal operators are `next` and `strong-until`. By the Guaranteed Coverage Theorem and the fact that $\Phi$ is universal, it is enough to show that $\Phi'$ is true of every behavior in $\widehat{M}$ starting at $s_0$. So, by the results in section 3.6, we need only to show that $\overline{M}(\Phi'), x \models \Phi'$ for every behavior $x$ in $\overline{M}(\Phi')$ starting at $s_0$. So let $x$ be a behavior in $\overline{M}(\Phi)$ starting at $s_0$. We will induct on the complexity of $\Phi'$. Unless otherwise noted, references to (S1-S3,B1-B3) refer to the definition of the semantics.

If $\Phi'$ is an atomic proposition, then $\Phi'$ is a state formula by (S1) of the definition of the syntax of EBTL. Since $\Phi'$ is a propositional state formula, $\Phi = \Phi'$ (Cf.

(S3) of the definition of the syntax of EBTL). Therefore $\overline{M}(\Phi'), x \models \Phi'$ by hypothesis and we are done.

Suppose $\Phi'$ is of the form `(and` $p$ $q$`)`. Then we reduce to the case of showing $\overline{M}(p), x \models p$ and $\overline{M}(q), x \models q$.

Suppose $\Phi'$ is of the form `(not` $p$`)`. Then we reduce to the case of showing that it is not the case that $\overline{M}(p), x \models p$.

Suppose $\Phi'$ is of the form `(strong-until` $p$ $q$`)`. We reduce to showing that for some nonnegative integer $j \leq \Lambda(x) - 1$ and for all nonnegative integers $k \leq j$, $\overline{M}(q), x^j \models q$ and $\overline{M}(p), x^k \models p$.

Suppose $\Phi'$ is of the form `(next` $p$`)` where $p$ is a behavior formula. It must be the case that $\Lambda(x) > 1$, otherwise $M, x \models \Phi$ could not have been true (Cf. (B3)). Thus we reduce to proving $\overline{M}(p), x^1 \models p$.

In each case we have reduced $\Phi'$ to a more simple expression. The obvious induction argument on the complexity of $\Phi'$ finishes the proof.

# 5 APPLICATIONS OF EBTL AND QSIM

EBTL 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.

## 5.1 PROVING PROPERTIES OF CONTROLLERS

Kuipers & Åström [1994] 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 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 EBTL, the validity of the guarantee can be automatically checked against the set of possible behaviors. Kuipers & Åström [1994] 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.

It should be noted that [Moon, et. al., 92] used CTL to prove a guarantee for a discrete-time control system. EBTL and QSIM make it possible to apply temporal logic to continuous-time control systems, and indeed to dynamical systems in general.

The program TL is equally easily applied to the behavior trees output by QSIM extensions such as NSIM and Q2, which use quantitative bounding information and produce quantitative bounds on the predictions. For these applications a slight extension of the propositional part of the language is helpful. We add the ability to include numerical information in the state propositions. This added expressiveness does not add to the complexity of the algorithm.

The program can be and has been used on terminals which do not support the graphics needed to see QSIM trees. In these circumstances, the user can learn everything he may need to know about a QSIM tree by evaluating a few carefully chosen EBTL statements.

## 5.2 TL AS A DEBUGGING TOOL FOR QSIM MODELS

Because QSIM is not complete, a QSIM tree may contain behaviors which do not correspond to real behaviors. Therefore, the truth of an EBTL statement (e.g. one beginning with the quantifier `possibly`), does not imply the truth of the corresponding statement in an actual behavior. This apparent limitation, however, can be and has been used as a debugging tool. For example, 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 behaviors in the QSIM tree. The implemented program TL allows EBTL formulæ to have side effects. Therefore, it can be used to print out the undesirable behaviors or states which satisfy a certain EBTL formula. In the actual TL code, there are features which make this process very easy.

## 5.3 EXAMPLES

We demonstrate the use of TL to ask and answer questions about two simple models: the undamped oscillator, whose behavior tree is rooted in the initial state `SS`; and the damped oscillator, whose behavior tree is rooted in the state `DS`.

**Undamped Oscillator** The simple spring conserves energy, so all behaviors are cycles, as shown by the behavior tree in figure 1. The three behaviors differ according to whether the amplitude of the oscillations passes a predefined landmark value. The queries shown demonstrate that the simple spring never becomes quiescent, always reaches a cycle state, and necessarily has an infinite sequence of events crossing $x = 0$ in opposite directions.

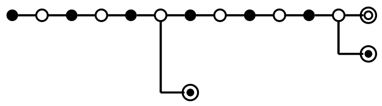
```
(TL SS '(necessarily
          (always (not (status quiescent)))))
  => T
```

```
(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
             (precedes (qval x (0 dec))
                       (qval x (0 inc))))))
  => T
```

**Damped Oscillator** The damped spring loses energy. The first behavior is a cycle representing a decreasing oscillation. The second two are partial cycles followed by "nodal" convergence to quiescent states at the origin (indicated by circled dots in the behavior tree). This finite behavior tree represents an infinite family of behaviors, oscillating a finite number of half-cycles around the origin before "nodal" convergence. Each of the universal questions asked about the simple spring behavior 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)))
               (infinitely-often (qval x (0 dec))))))
  => T

(TL DS '(possibly
          (infinitely-often
            (precedes (qval x (0 dec))
                      (qval x (0 inc))))))
  => T
```

# 6 FUTURE DIRECTIONS

QSIM and EBTL can be combined to help in the design of a QDE. One possibility is to allow EBTL formulæ as part of the input to the QSIM program. In this case, QSIM would only generate those behaviors which are models for the EBTL formulæ. I.e. QSIM would test the satisfiability of the conjunction of the EBTL formulæ. This would allow a qualitative model to be described jointly by a QDE and an EBTL description of its behavior.

The limiting case, with an EBTL specification of the desired behavior and no QDE, raises an intriguing possibility. QSIM would predict all behaviors consistent with continuity and the EBTL specifications. A recently-developed program called MISQ takes as input a set of qualitative behaviors and produces the minimal QDE capable of producing that behavior [Richards, et al, 92]. This would be useful, for example, to a controller designer who knows that he wants certain qualitative events to occur, not to occur, or to occur infinitely often. By providing this specification in the form of EBTL formulæ, this combination of EBTL, QSIM, and MISQ might be able to design the appropriate QDE model.

Work is currently being done with the goal of automatically generating natural explanations of the structures associated with QSIM. This requires the detection of certain common features in physical systems, e.g. negative feed-back loops, oscillation, etc. While EBTL is useful for many parts of this process, more expressiveness is clearly required.

In particular, it will be important to compare (not just quantify over) behaviors and states, and to compare and quantify over variables in the QDE. In some cases this can be done in EBTL, though awkwardly. It would not be enough to build EBTL on a first-order logic instead of a propositional logic, since quantification to compare behaviors, states or variables must be scoped *outside* of the modal and temporal operators. This would undoubtedly have a substantial impact on complexity.

# 7 MISCELLANY

## 7.1 COMPUTATIONAL COMPLEXITY

Checking the validity of statements in BTL is polynomial, and EBTL is exponential, in the size of the statement. However, since the statements are typically not enormous, the more important constraint is that validity checking is linear in the size of the behavior tree.

## 7.2 CODE

The code for QSIM is available via anonymous ftp at `cs.utexas.edu` in the directory `ftp/pub/qsim`. The up-to-date version of TL will be included with the release of QSIM by KR'94.

## 7.3 RELATED WORK

Related work has been done in applying temporal logics to various models. Some of the logics developed have been able to express more quantitative *time* information. Since QSIM does not express information about the "real" length of time intervals, these languages are not practicable in our situation. We specifically mention for example [Jahanian, 88]. In this paper, real time systems are modeled in the Modechart language. Statements in Real Time Logic can be checked against a Modechart model. Real Time Logic is undecidable in general but certain classes of statements are shown to be decidable. These languages are suited for time-critical systems. However, if all that is important is the *order* of events, then languages such as CTL* are sufficiently expressive. In [Moon, 92], statements in CTL are checked against state transition graphs generated from programmable logic controller ladder diagrams. The specific application in [Moon, 92] is to chemical process control. Possibly the most work has been done in applications of temporal logics to computer processes such as parallel computing. [Emerson, 90] and [Lichtenstein, 84] are examples of such work. [Collins, 89] took an early step in the application of temporal logic to QSIM.

### 7.4 HISTORY

In 1989, Kuipers began discussing the application of branching-time temporal logic to QSIM with David W. Franke and E. Allen Emerson. In 1990, Kuipers wrote the code on which TL is based. In 1992-93, Shults added the finite unwinding of cycle states and discovered the new theorems presented in this paper.

## 8 CONCLUSION

This paper has presented a method using modal and temporal logic to prove properties of the behavior of a continuous physical system. If the user can describe a physical system in terms of a set of qualitative constraints, then by using QSIM and TL, he or she can prove theorems about the behavior of any real system consistent with those constraints. We therefore provide a meaningful and sound interpretation for the phrase, "proof by simulation."

We expect that this link between logic-based and simulation-based inference methods will support a variety of hybrid reasoning techniques that could be of substantial value.

### Acknowledgements

We would like to thank Daniel Clancy, Sowmya Ramachandran, Rich Mallory, Jeff Rickel and Robert Schrag for fruitful discussions.

The work of Benjamin Kuipers and the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, The University of Texas at Austin is supported in part by NSF grants IRI-8904454, IRI-9017047, and IRI-9216584, and by NASA contracts NCC 2-760 and NAG 9-665.

### References


- Tim Collins. A Temporal Logic for QSIM. unpublished term paper. 1989.
- E. Allen Emerson. 1990. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, (J. van Leeuwen, ed.), Elsevier Science Pub. B. V./MIT Press, 1990, pp. 995-1072.
- David W. Franke. 1991. Deriving and using descriptions of purpose. *IEEE Expert*, April 1991, pp. 41-47.
- Farnam Jahanian and Douglas A Stewart, A Method for Verifying Properties of Modechart Specifications. *Proceedings of the Real-time Systems Symposium.* Huntsville, AL December 1988.
- Benjamin J. Kuipers. 1986. Qualitative simulation. *Artificial Intelligence* **29**: 289 - 338.
- Benjamin J. Kuipers. Reasoning with qualitative models. 1993. *Artificial Intelligence* **59**: 125-132.
- Benjamin J. Kuipers. Qualitative simulation: then and now. 1993. *Artificial Intelligence* **59**: 133-140.
- Benjamin J. Kuipers. 1994. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge.* Cambridge, MA: MIT Press, in press.
- Benjamin J. Kuipers and K. Åström. 1994. The composition and validation of heterogeneous control laws. *Automatica* **30**(2), February 1994, to appear.
- O. Lichtenstein & A. Pnueli. 1984. Checking that finite state concurrent programs satisfy their linear specifications. *Twelfth Annual ACM Symposium on Principles of Programming Languages*, pp. 97-107.
- I. Moon, G. J. Powers, J. R. Burch & E. M. Clarke. 1992. Automatic verification of sequential control systems using temporal logic. *AIChE Journal* **38**(1): 67-75.
- Bradley L. Richards, Ina Kraan and Benjamin J. Kuipers. 1992. Automatic abduction of qualitative models. *Proceedings of the National Conference on Artificial Intelligence (AAAI-92)*, AAAI/MIT Press, 1992.
- Hao Wang. 1960. Toward Mechanical Mathematics. reprinted in Automation of Reasoning I. ed. Jörg Siekmann and Graham Wrightson. Springer-Verlag 1983 pp. 244-264.