Time-Aware Relational Abstractions for Hybrid Systems

Sergio Mover[*]
FBK, Trento
mover@fbk.eu

Alessandro Cimatti
FBK, Trento
cimatti@fbk.eu

Ashish Tiwari[*†]
CSL, SRI International
tiwari@csl.sri.com

Stefano Tonetta
FBK, Trento
stonetta@fbk.eu

ABSTRACT

Hybrid Systems model both discrete switches and continuous dynamics and are suitable to represent embedded systems where discrete controllers interact with a physical plant.

Relational abstraction is a new approach for verifying hybrid systems. In relational abstraction, the continuous dynamics in each location of the hybrid system is abstracted by a binary relation that relates the current value of the continuous variables with *all* future values of the variables that are reachable after a time elapse (continuous) transition. The abstract system is an infinite-state system, which can be verified using k-induction or abstract interpretation.

Existing techniques for computing relational abstractions are time-agnostic: they do not construct any relationship between the state variables and the time elapsed during the continuous evolution. Time-agnostic abstractions cannot verify timing properties.

We present a technique to compute a time-aware relational abstraction for verifying (timing-related) safety properties of cyber-physical systems. We show the effectiveness of the new abstraction on several case studies on which the previous techniques fail.

Categories and Subject Descriptors

C.3 [**Special-purpose and application-based systems**]: Real-time and embedded systems; D.2.4 [**Software verification**]: Formal methods

General Terms

Design, Verification

Keywords

Hybrid Systems, Formal Methods, Verification, Abstraction

1. INTRODUCTION

Hybrid systems exhibit both discrete and continuous behaviors. For this reason, they are a suitable formalism to model the embedded systems used in several domains (e.g. automotive, avionics, . . .), where digital controllers interact with the physical environment. Since many such systems are safety critical, it is imperative to have effective verification techniques that can prove safety of such systems. However, ensuring safety of these kind of systems is a challenging task, due to the interaction between the discrete and the continuous dynamics.

Hybrid automata is the formal framework used to model the discrete and continuous behavior of hybrid systems [19]. One approach for verifying safety properties of hybrid automata is based on constraint solvers called Satisfiability Modulo Theories (SMT) solvers. In this approach, hybrid automata are encoded as infinite-state transition systems using SMT constraints, which are then verified using techniques such as k-induction. The encoding may be either precise or it may be an over-approximation of the original system.

Linear hybrid systems is an important class of hybrid systems, where the dynamics of the continuous variables are described by means of linear Ordinary Differential Equations (ODEs). Unfortunately, the SMT-based verification approach can not directly be applied to linear hybrid systems. This is because it is unclear how to map continuous transitions specified by linear ordinary equations into SMT constraints.

One way to enable SMT-based verification for linear hybrid systems is based on using *relational abstraction* [28]. Relational abstraction replaces ordinary differential equations by a binary relation over the state space that over-approximates the binary reachability relation induced by the ordinary differential equations. Thus, by replacing the ODEs in each mode by their relational abstraction, it is possible to over-approximate a linear hybrid system with an infinite-state transition system. Relational abstraction has been used successfully to verify several systems [28, 33].

There are two shortcomings in the procedure for computing relational abstractions as implemented in HybridSal [33]. First, HybridSal generates one fixed abstraction for any given ODE. Hence, if relational abstraction fails to prove a valid property, then there is no option for refining the abstraction. Second, the abstractions generated by Hybrid-

[*] Supported in part by NSF grants CSR-0917398 and SHF:CSR-1017483.

[†] Supported in part by DARPA under contract FA8750-12-C-0284 and under subcontract VA-DSR 21806-S4 under prime contract FA8650-10-C-7075.

Sal are time-agnostic: they do not relate the time elapsed in the continuous transitions with the state variables.

Time-agnostic relational abstractions can prove several nontrivial properties in many cases, but they fail in two important cases. First, obviously, time-agnostic relational abstractions cannot be used to verify timing properties. Second, time-agnostic abstractions are very coarse (imprecise). In fact, they often do not capture the relationship between different variables of the system. Even in simple cases, the loss of information in time-agnostic relational abstraction is very significant; see Section 3 for an example.

In this paper, we present a new technique to compute a more precise, *time-aware*, relational abstraction. Time-aware relational abstraction overcomes both the shortcomings of relational abstraction procedure implemented in HybridSal. First, it is not time-agnostic and it contains information about the relationship between the state variables and the time-elapse variable. Second, time-aware relational abstraction enables tuning the precision of the abstraction.

Our main contribution is a procedure for computing time-aware relational abstraction that can be tuned to achieve any desired precision-efficiency trade-off. The abstraction can be made more and more precise, but that increases the cost of analyzing it using bounded model checking.

Our approach for generating time-aware relational abstractions is based on exploiting the eigenstructure of the matrix A of the linear ordinary differential equations. The technique for creating time-agnostic relational abstractions, as implemented in HybridSal [33], was also based on the same basic idea. However, we have to non-trivially extend that idea to preserve information about time (rather than throwing it away) in the abstraction. The key challenge in extending the procedure for creating time-agnostic relational abstractions to time-aware relational abstractions is that the relationship between the time elapsed (Δt) and the change in the value of any state variable (Δx) is seldom linear. It is often nonlinear, and it often contains quadratic terms and transcendental functions, such as the exponential function and the trigonometric functions. Since we wish to perform bounded model checking on the abstraction, we need the abstract system to be specified in "easy" theories, namely, in linear real arithmetic. The main technical contribution of the paper shows how to create time-aware relational abstraction of linear ODEs using piecewise linear approximations of nonlinear (transcendental) functions.

We extended the HybridSal tool to optionally also compute time-aware relational abstractions. Time-aware abstractions generated by HybridSal are then analyzed using the SAL infinite bounded model checker and k-induction prover [15]. Both tools are publicly available from SRI International. We show the effectiveness of our approach on two case studies, a PID controller and an active suspension controller. We show that, while the previous relational abstraction was too coarse, time-aware abstraction is able to verify time properties on both benchmarks.

The paper is organized as follows. In Section 2 we introduce some background notions while in Section 3 we show with an example the need of a more precise abstraction. In Section 4 we present the time-aware abstraction. We compare our technique with the related works in Section 5 and then we show the effectiveness of the novel approach in Section 6. In Section 7 we draw some conclusion and outline possible directions of future works.

2. PRELIMINARIES

We fix our definition and notation for hybrid automata, infinite-state transition systems, SMT-based verification, and relational abstractions below.

2.1 Hybrid Automata

Definition 1. A *Hybrid Automaton* (HA) [19] is a tuple $\langle Q, Q_0, R, X, \mu, \iota, \xi, \theta\rangle$ where:

- Q is the set of modes/locations,
- $Q_0 \subseteq Q$ is the set of initial modes,
- $R \subseteq Q \times Q$ is the set of discrete transitions,
- X is the set of continuous variables,
- $\mu : Q \to P(X \cup \dot{X})$ is the flow condition,
- $\iota : Q \to P(X)$ is the initial condition,
- $\xi : Q \to P(X)$ is the invariant condition,
- $\theta : R \to P(X \cup X')$ is the jump condition,

where $\dot{X}$ represents the derivative of the variables X during a continuous evolution, X' represents the variables X after one transition and P represents the set of predicates over the specified set of variables.

A *state* of a hybrid automaton H is a tuple $\langle l, v\rangle$, where $l \in Q$ is a mode, and $v : X \mapsto \mathbb{R}$ is a valuation of all the continuous variables X.

Definition 2. A sequence $\langle l_0, v_0\rangle \xrightarrow{\delta_1} \dots \xrightarrow{\delta_k} \langle l_k, v_k\rangle$ of states is a *trace* of the hybrid automaton H if:

- $l_0 \in Q_0$, $v_0 \models \iota(l_0)$ and for $0 \le i \le k$ $l_i \in Q, v_i \in \mathbb{R}^n$,
- for $1 \le i \le k$, $\delta_j \in \mathbb{R}$ and $\langle l_{i-1}, v_{i-1}\rangle \xrightarrow{\delta_i} \langle l_i, v_i\rangle$ we have that either:
  - *Discrete transition:* $\delta_i = 0$ and $\langle l_{i-1}, l_i\rangle \in R$, $v_{i-1} \models \xi(l_{i-1})$, $v_i \models \xi(l_i)$, $\langle v_{i-1}, v_i\rangle \models \theta(\langle l_{i-1}, l_i\rangle)$.
  - *Continuous transition:* $\delta_i > 0$ and there exists a continuous and differentiable function $f_{l_{i-1}} : \mathbb{R}^{n+1} \to \mathbb{R}^n$ such that $\dot{f}_{l_{i-1}} \models \mu(l_{i-1})$, $v_{i-1} = f_{l_{i-1}}(v_{i-1}, 0)$, $v_i = f_{l_{i-1}}(v_{i-1}, \delta_i)$, $v_i \models \xi(l_i)$ and for all $t \in [0, \delta_i)$, $f_{l_{i-1}}(v_{i-1}, t) \models \xi(l_{i-1})$.

A state $\langle l, v\rangle \in Q \times \mathbb{R}^n$ is *reachable* if there exists a trace π of H such that $\pi = \langle l_0, v_0\rangle \xrightarrow{\delta_1} \dots \xrightarrow{\delta_k} \langle l, v\rangle$. A set S of states is an *invariant* if, for all the reachable states $\langle l, v\rangle$, we have $\langle l, v\rangle \in S$.

Definition 3. Given an hybrid automaton H and a set of states $S \in Q \times \mathbb{R}^n$, the *safety verification problem* consists of checking if S is an invariant for the hybrid system H.

We will denote with $\vec{x}$ a vector of all the variables $x \in X$, where the order of the variables is chosen arbitrarily. This notation enables to describe a system of ODEs. In this paper, we deal with *Linear Hybrid Systems*, where for all $q \in Q$ the flow condition $\mu(q)$ is of the form $\dot{\vec{x}} = A\vec{x} + \vec{b}$, with $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$.

2.2 Infinite-state Transition system (ITS)

Infinite-state transition systems are defined by first-order logic formulas. Here, we assume the standard definition of first-order logic formulas (a comprehensive treatment of this topic may be found in [7]). *Satisfiability Modulo Theory* $\mathcal{T}$ (SMT($\mathcal{T}$)) [5] is the problem of checking if a first-order logic formula ϕ is satisfiable, for some background theory $\mathcal{T}$. In this paper, we will interpret the formulas in the *Linear Real Arithmetic* Theory (i.e. a quantifier-free Boolean combinations of atoms in the form $\sum a_i \cdot x_j \bowtie a$, where x_j is a real-valued variable, $a_i, a \in \mathbb{Q}$ and $\bowtie \in \{<, \leq, >, \geq, \neq\}$).

Given a set V of variables, we denote with $V', V^0, V^1, \ldots$ copies of such set. An infinite-state transition system (ITS) is a tuple $S = \langle V, Init, Inv, Trans \rangle$ such that:

- V is a set of variables;
- $Init$ is a first-order formula over V (called initial condition);
- Inv is a first-order formula over V (called invariant condition);
- $Trans$ is a first-order formula over $V \cup V'$ (called transition condition).

A *state* s is an assignment to the variables V. We denote with $s', s^0, s^1, \ldots$ the corresponding assignment to the copy $V', V^0, V^1, \ldots$ of V. A sequence $s_0, s_1, \ldots, s_k$ of states is a model (also called *path*) of the transition system $S = \langle V, Init, Inv, Trans \rangle$ iff:

- s_0 satisfies $Init$;
- for every $0 \leq i \leq k$, s_i satisfies Inv;
- for every $0 \leq i < k$, s_i, s'_{i+1} satisfy $Trans$.

Many verification techniques for infinite-state transition systems, such as Bounded Model Checking (BMC) [6] and k-induction [29], are based on satisfiability checking using SMT solvers.

SMT-solvers may also handle formulas in the *Theory of Reals* (i.e. elementary algebra). However, despite of the recent results [23], SMT-based verification techniques are not efficient for systems with non-linear arithmetic constraints [11]. Moreover, in the case of non-linear constraints with transcendental functions the satisfiability problem is undecidable. Thus, a basic requirement to keep the analysis of infinite-state transition systems tractable is to avoid formulas in the *Theory of Reals*, and restrict to the more tractable *Linear Real Arithmetic.*

2.3 SMT-based verification of hybrid automata

In the Satisfiability Modulo Theory (SMT) verification an hybrid automaton H is encoded as an infinite-state transition system S. Then, the infinite-state transition system is analyzed using SMT-based verification techniques.

The discrete structure of H (i.e. the locations and the discrete transitions) can be easily encoded in S. Here, we do not give the details of the encoding of the discrete locations and discrete transitions of H (see [30, 1] for examples of this encoding). In fact, we assume we have a one-to-one mapping between the discrete locations and the discrete transitions of the hybrid automaton and their encoding in the transition system S. (Intuitively, S has a copy of the discrete structure of H).

The encoding of the continuous dynamics (i.e. the flow condition μ) is more complex, since it requires to have an explicit solution to the differential equations and to faithfully encode the invariants during the continuous evolution. In practice, the precise encoding can be done only for some very restrictive sub-classes of *Linear Hybrid Systems* [30, 14, 1, 11]. By compromising precision, relational abstraction provides a general way to encode the continuous dynamics μ for any *Linear Hybrid System.*

We will write $loc = q$ to refer to the encoding in S of the location $q \in Q$. Also, we assume that for each continuous variable $x \in X$ of H, there is a corresponding real-valued variable $x \in V$ of S. Thus, we map a state $\langle l, v \rangle$ of H to a state of the transition system $\rho(\langle l, v \rangle) = \langle loc = l, x_1 = v_1, \ldots, x_n = v_n \rangle$.

In the following, we assume that both the continuous and the discrete transition of H are encoded in the $Trans$ constraint of S:

$$Trans := \bigwedge_{q \in Q} (loc = q \rightarrow (Trans_D(q) \vee Trans_C(q)))$$

where $Trans_D(q)$ and $Trans_C(q)$ encode respectively the discrete transitions leaving the location q and the continuous dynamics in q.

2.4 Relational Abstraction

Relational abstraction [28] abstracts the continuous evolution in each location of the hybrid automaton, leaving unchanged its discrete locations and transitions. In particular, the abstraction relates all the values assigned to the continuous variables with all the possible future values assigned to the continuous variables after a continuous transition. We encode the relational abstraction for a location q in the formula $Trans_C(q)$.

Definition 4. $Trans_C(q)$ is a *relational abstraction* for the location $q \in Q$ if $\forall v_i \in \mathbb{R}^n, \delta \in \mathbb{R}$, it is the case that:

$$\rho(\langle q, v_i \rangle), \rho(\langle q, f_q(v_i, \epsilon) \rangle) \models Trans_C(q)$$

where $f_q : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^n$ is the solution to the flow condition $\mu(q)$.

We now define when an infinite-state transition system is an abstraction of a hybrid system.

Definition 5. An infinite-state transition system S is an *abstraction* of the hybrid system H if for all traces $\pi_H = \langle l_0, v_0 \rangle \xrightarrow{\delta_1} \ldots \xrightarrow{\delta_k} \langle l_k, v_k \rangle$ of H there exists a path $\pi_S = \rho(\langle l_0, v_0 \rangle), \ldots, \rho(\langle l_k, v_k \rangle)$ of S.

It follows immediately from the definitions above that a relational abstraction of a hybrid automaton H is an infinite-state transition system S, and moreover, S is an abstraction of H.

Similarly to the set X', we will write x' and $\vec{x}'$ for the value of the variable x and the vector of variables $\vec{x}$ after a transition. If μ_q describes a linear system $\dot{\vec{x}} = A\vec{x} + b$ a relational abstraction may be computed exploiting the eigenstructure of the matrix A [33]. First, the system of differential equations is rearranged partitioning the variables $\vec{x}$ into $\vec{y}$ and $\vec{z}$ such that:[1]

$$\begin{bmatrix} \dot{\vec{y}} \\ \dot{\vec{z}} \end{bmatrix} = \begin{bmatrix} A_1 & A_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{y} \\ \vec{z} \end{bmatrix} + \begin{bmatrix} \vec{b_1} \\ \vec{b_2} \end{bmatrix}$$

[1] Note that if there are no variables with constant derivative the dimension of $\vec{z}$ is 0, which is a simpler special case.

Given an $n \times n$ matrix M, we denote with Λ_M the set of all the pairs $\langle \lambda, \vec{c} \rangle$, where λ is an eigenvalue of the matrix M and $\vec{c}$ is one of its associated left eigenvectors. More precisely, for each eigenvalue λ we consider only linearly independent eigenvectors. The abstraction $Trans_C(q)$ is computed from each pair $\langle \lambda, \vec{c} \rangle \in \Lambda_{A_1}$:

$$Trans_C(q) := \bigwedge_{\langle \lambda, \vec{c} \rangle \in \Lambda_{A_1}} \phi_{\langle \lambda, \vec{c} \rangle} \wedge \bigwedge_{z_1, z_2 \in \vec{z}} \frac{z_1' - z_1}{b_{2_1}} = \frac{z_2' - z_2}{b_{2_2}}$$

where $\phi_{\langle \lambda, \vec{c} \rangle}$ is a formula that depends on the kind of eigenvalues and eigenvectors, which may be real or complex.

If $\vec{c} \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$, the abstraction is defined with the predicate $p(\vec{x}) = \vec{c}^T \vec{y} + \vec{d}^T \vec{z} + e$, where $\vec{d}^T = \vec{c}^T \frac{A_2}{\lambda}$, $e = \frac{\vec{c}^T \vec{b_1} + \vec{d}^T \vec{b_2}}{\lambda}$. In the following, we use p to denote the linear expression $p(\vec{x})$ and p' to denote the linear expression $p(\vec{x}')$ over the next-state variables.

$$\phi_{\langle \lambda, \vec{c} \rangle} := \begin{cases} (p' \leq p < 0) \vee (0 < p \leq p') \vee (0 = p = p') & \text{if } \lambda > 0 \\ (p \leq p' < 0) \vee (0 < p' \leq p) \vee (0 = p = p') & \text{if } \lambda < 0 \\ t' - t = \frac{\vec{c}^T (\vec{y'} - \vec{y})}{\vec{c}^T \vec{b_1}} & \lambda = 0 \end{cases}$$

If $\vec{c} \in \mathbb{C}^n$ and $\lambda \in \mathbb{C}$, the abstraction is defined using two predicates, $p_1(\vec{x})$ and $p_2(\vec{x})$. Suppose $\vec{c} = \vec{d} + i\vec{e}$ and $\lambda = \alpha + i\beta$. We define p_1 and p_2 as follows: $p_1(\vec{x}) = \vec{d}^T \vec{y} + \vec{c_1}^T \vec{z} + e_1$, $p_2(\vec{x}) = \vec{e}^T \vec{y} + \vec{c_2}^T \vec{z} + e_2$ and $c_1, c_2, \vec{c_1}^T, \vec{c_2}^T, e_1, e_2$ are such that $\dot{p_1} = \alpha p_1 - \beta p_2$ and $\dot{p_2} = \beta p_1 + \alpha p_2$. The formula $\phi_{\langle \lambda, \vec{c} \rangle}$ added to $Trans_C(q)$ is:

$$\phi_{\langle \lambda, \vec{c} \rangle} := \begin{cases} p_1^2(\vec{x}) + p_2^2(\vec{x}) \geq p_1^2(\vec{x'}) + p_2^2(\vec{x'}) & \text{if } \alpha \leq 0 \\ p_1^2(\vec{x'}) + p_2(\vec{x'}) \geq p_1^2(\vec{x}) + p_2^2(\vec{x}) & \text{if } \alpha \geq 0 \end{cases}$$

A practical requirement for the abstraction is that it has to be expressed in a formula in the *Linear Real Arithmetic* Theory. Note that the abstraction, $\phi_{\langle \lambda, \vec{c} \rangle}$, generated in the complex eigenvalue case is non-linear and thus it is approximated in the original approach [33].

3. SIMPLE MOTIVATING EXAMPLE

We illustrate the main idea underlying time-aware relational abstraction with a simple contrived example.

Consider the linear system

$$\begin{aligned} \dot{x} &= -2x, \\ \dot{y} &= 0.5 - y \end{aligned}$$

with initial condition $x \in [-1, 0.9], y \in [1.1, \infty)$. We want to prove that x is always less-than y; that is, $\mathtt{G}(x < y)$.

One way to prove safety of such systems involves constructing a relational abstraction of the system and then verifying the abstract system using infinite bounded model checking and k-induction.

Relational abstraction replaces the differential equation by a discrete transition. The abstract transition relates the current value of x, y with *any future* value x', y'. The default relational abstraction, constructed by HybridSal, for the above differential equation is the following transition:

$$\begin{aligned} (x = x' = 0 \;\vee\; 0 < x' \leq x \;\vee\; 0 > x' \geq x) \quad \wedge \\ (y = y' = 0.5 \;\vee\; 0.5 < y' \leq y \;\vee\; 0.5 > y' \geq y) \end{aligned}$$

In the abstract system, there is a transition from (x, y) to a new state (x', y') if these four values satisfy the above constraint.

The abstraction is sound: starting from (x, y) and following the solutions of the differential equations, if we reach (x', y') at any future time instance, then x, y, x', y' will necessarily satisfy the above constraint. However, the relational abstraction is very coarse (imprecise). In particular, the rate at which x and y are changing is abstracted away. If $\dot{x} = -2x$ were replaced by $\dot{x} = -0.2x$, we would still get the same relational abstraction.

As a result of this imprecision, the default relational abstraction is insufficient to prove the safety property. (We get a spurious counterexample when trying to prove the safety property using the above relational abstraction.)

In this paper, we construct more precise relational abstractions, called *time-aware* relational abstraction. First, we make the implicit time variable explicit by adding a new variable called $\mathtt{t}$ to the state space and the differential equation $\dot{\mathtt{t}} = 1$ to the dynamics. Time t is the glue that will help relate the change in x to the change in y.

The new time-aware relational abstraction relates the old values x, y, t to the new values x', y', t'. For the above system, the new abstract transition is the conjunction of two constraints: the first constraint, shown below, relates x, x', t, t':

$$\begin{aligned} &(x = x' = 0) \quad \vee \\ &(0 < x' \leq x \quad \wedge \quad \ln_{\mathtt{lb}}(x) - \ln_{\mathtt{ub}}(x') \geq -2(t' - t) \geq \\ &\qquad\qquad\qquad\quad \ln_{\mathtt{ub}}(x) - \ln_{\mathtt{lb}}(x')) \quad \vee \\ &(0 > x' \geq x \quad \wedge \quad \ln_{\mathtt{lb}}(-x) - \ln_{\mathtt{ub}}(-x') \geq -2(t' - t) \geq \\ &\qquad\qquad\qquad\quad \ln_{\mathtt{ub}}(-x) - \ln_{\mathtt{lb}}(-x')) \end{aligned}$$

There is a similar second constraint that relates y, y', t, t'.

There are two key points to note here. First, there is no simple *linear* relationship between the time variable and the x, y variables. They are related through the natural logarithm (ln) function; specifically, $x(t) = x(0)e^{-2t}$ is (part of) the solution of the above differential equation; and hence we have $-2(t' - t) = \ln(x') - \ln(x)$. To enable analysis using BMC/SMT tools, we need a *piecewise linear* function that computes a sound lower- and upper-bound of the ln function. These approximate functions, called $\ln_{\mathtt{lb}}$ and $\ln_{\mathtt{ub}}$ respectively, will be defined later.

Second, all interactions between different state variables are captured via the time variable. Note that the first constraint above relates x, x' with the time elapsed $t' - t$, and similarly the second constraint relates y, y' with the time elapsed $t' - t$. By reasoning over the conjunction, we deduce the relationship between x and y.

In particular, using the refined time-aware relational abstraction, we can prove the safety property $\mathtt{G}(x < y)$.

4. TIME-AWARE RELATIONAL ABSTRACTION

A time-aware relational abstraction of a dynamical system is a binary relation that holds between the current state of the system, including the current time, and any future reachable state of the system (including the future time). In this section, we describe a procedure for constructing time-aware relational abstractions of linear dynamical systems; that is, systems whose dynamics are specified using linear ordinary differential equations of the form $\dot{\vec{x}} = A\vec{x} + \vec{b}$.

4.1 Overall Approach

Consider a linear system $\dot{\vec{x}} = A\vec{x} + \vec{b}$. The exact relationship between t and the variables $\vec{x}$ is given by the explicit solution:

$$\vec{x}(t) \quad := \quad x_0 e^{At} + \int_{s=0}^{t} e^{(t-s)A}\vec{b}\,\mathrm{d}s \qquad (1)$$

It is hard to reason with this solution directly. In some very special cases, this explicit solution can be used to effectively solve the reachability problem for linear systems [24]. This happens, for example, when either A is nilpotent, or its eigenvalues are either all reals or all purely imaginary [24], but even in these restricted cases, the solution requires reasoning over nonlinear real arithmetic. Hence, working with the explicit solution in Equation 1 is not very practical.

In our approach, we create an abstraction of the solution. Specifically, we construct a relationship between the current value $\vec{x}$ of the state variables, the future value $\vec{x}'$ of the state variables, the current value t of time, and the future value t' of time. The relationship we construct will overapproximate the binary reachability relation.

To ease the presentation, we assume that the set of continuous variables X of the hybrid automaton H contains a clock variable t, which counts the total time elapsed in the system. Initially t is 0 and its derivative is 1 in all the locations. Also, t is never used in a jump or in an invariant condition[2]. For keeping the presentation simple, we assume $\vec{b} = \vec{0}$. The results extend easily to the case when $\vec{b} \neq \vec{0}$.

Let R be a set of tuples $\langle r, \vec{c}\rangle$ such that $\frac{d\vec{c}^T\vec{x}}{dt} = r$, for some $r \in \mathbb{R}$. For each location $q \in Q$, for each $\langle \lambda, \vec{c}\rangle \in \Lambda_A$ and for each $\langle r, \vec{c}\rangle \in R$, we obtain the following time-aware relational abstraction:

$$Trans_C(q)^t \quad := \quad \bigwedge_{\langle\lambda,\vec{c}\rangle\in\Lambda} \phi^t_{\langle\lambda,\vec{c}\rangle} \ \wedge \ \bigwedge_{\langle r,\vec{c}\rangle\in R} \phi^c_{\langle r,\vec{c}\rangle} \qquad (2)$$

where $\phi^t_{\langle\lambda,\vec{c}\rangle}$ is the time-aware abstraction generated from $\langle\lambda,\vec{c}\rangle$ and $\phi^c_{\langle r,\vec{c}\rangle}$ is the abstraction generated for linear expressions with a constant-rate derivative. We define $\phi^t_{\langle\lambda,\vec{c}\rangle}$ and $\phi^c_{\langle r,\vec{c}\rangle}$ below. As in the case of the eigenstructure-based abstraction, the definition of $\phi^t_{\langle\lambda,\vec{c}\rangle}$ depends on whether the eigenvalue λ is real or complex.

4.2 Real Eigenvalues

We define $\phi^t_{\langle\lambda,\vec{c}\rangle}$ for real λ now. Consider a linear dynamical system $\dot{\vec{x}} = A\vec{x}$. Let $\vec{c}$ be a left eigenvector of A corresponding to some real eigenvalue λ; that is,

$$\vec{c}^T A = \lambda\vec{c}^T$$

where $\vec{c}^T$ is a column vector obtained by transposing (the row vector) $\vec{c}$. (Equivalently, $\vec{c}$ is a eigenvector of the transpose of A). Consider the linear expression

$$p(\vec{x}) \quad = \quad \vec{c}^T\vec{x}$$

Clearly, we have

$$\frac{dp(\vec{x})}{dt} \quad = \quad \frac{d\vec{c}^T\vec{x}}{dt} \quad = \quad \vec{c}^T\frac{d\vec{x}}{dt} \quad = \quad \vec{c}^T A\vec{x} \quad = \quad \lambda\vec{c}^T\vec{x} = \lambda p(\vec{x})$$

Hence, the value of p changes exponentially; that is:

$$p(\vec{x}(t)) = p(\vec{x}(0))e^{\lambda t} \qquad (3)$$

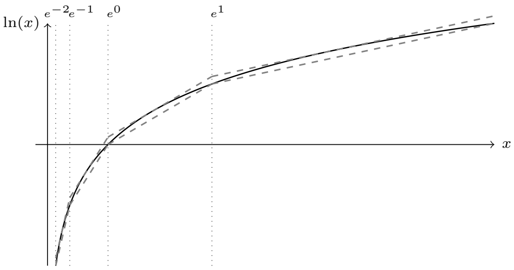

Figure 1: Piecewise linear approximation for natural logarithm function. The solid line plots $\ln(x)$ and the dotted lines shows the piecewise linear under- and over-approximations.

The linear expression p can be used to constrain the future value, $\vec{x}'$, and the current value, $\vec{x}$, of the state variables. We use p to denote the linear expression $p(\vec{x})$ and p' to denote the linear expression $p(\vec{x}')$ over the next-state variables. When $\lambda > 0$, the following constraint holds between any future value $\vec{x}'$ and the current value $\vec{x}$ of the state variables:

$$\begin{aligned}\psi(p,p') := (p = p' = 0) \quad & \vee \\ (0 < p \leq p' \quad & \wedge \quad \ln_{\texttt{lb}}(p') - \ln_{\texttt{ub}}(p) \leq \lambda(t'-t) \leq \\ & \qquad \ln_{\texttt{ub}}(p') - \ln_{\texttt{lb}}(p)) \quad \vee \\ (0 > p \geq p' \quad & \wedge \quad \ln_{\texttt{lb}}(-p') - \ln_{\texttt{ub}}(-p) \leq \lambda(t'-t) \leq \\ & \qquad \ln_{\texttt{ub}}(-p') - \ln_{\texttt{lb}}(-p))\end{aligned}$$

When $\lambda < 0$, the constraint is the same as above, but with p and p' swapped. We thus have

$$\phi^t_{\langle\lambda,\vec{c}\rangle} \quad = \quad \begin{cases} \psi(p,p') & \text{if } \lambda > 0 \\ \psi(p',p) & \text{if } \lambda < 0 \\ p = p' & \text{if } \lambda = 0 \end{cases} \qquad (4)$$

where ψ is as defined above.

We remark that we only need a linear expression $p(\vec{x})$ that satisfies the equation $\frac{dp(\vec{x}(t))}{dt} = \lambda p$. For linear dynamical systems, such a p can be found using the left eigenvectors of the A matrix. For nonlinear dynamics, one needs to develop other techniques to obtain such a p, but once found, it can be used to construct time-aware relational abstractions of nonlinear systems too [34].

We now define the functions $\ln_{\texttt{lb}}$ and $\ln_{\texttt{ub}}$. These functions are piecewise linear approximations of the (lower and upper bounds for the) nonlinear natural logarithm function ln; that is, they satisfy the following condition:

$$\ln_{\texttt{lb}}(x) \ \leq \ \ln(x) \ \leq \ \ln_{\texttt{ub}}(x), \quad \forall x \in \mathbb{R}^+$$

Piecewise Linear Approximation for Natural Logarithm

The natural logarithm function, $\ln(x)$, can be approximated using a piecewise linear function, as described in [21]. Figure 1 illustrates this approximation.

We first divide the real number line into infinitely many intervals. Consider the infinitely many intervals $I_k := [e^k, e^{k+1}]$, $k \in \mathbb{Z}$. Clearly, we have $\bigcup_{k\in\mathbb{Z}} I_k = (0, \inf)$.

Since the logarithm function is concave, it is easy to obtain a piecewise linear underapproximation of the function. This underapproximation is obtained by just linearly extrapolating within each interval, as shown in Figure 1. Specifically,

[2] If we are not interested in the total amount of time elapsed, we can keep only the time elapsed during a continuous transition and not its total.

if x is in the interval I_k, then $\ln(x)$ is approximated by:

$$\ln_{\mathtt{lb}}(x) \;=\; \frac{e^{-k}}{e-1}x + k - \frac{1}{e-1} \tag{5}$$

This idea for approximation works not only for the base e, but also for any base $a > 1$. For base e, the approximation error is defined by $\gamma(x) = \ln(x) - \ln_{\mathtt{lb}} x$. In any interval I_k, γ is bounded by

$$\ln(e-1) - 1 + \frac{1}{e-1} \tag{6}$$

In general, γ depends only on the base of the logarithm and not on the interval itself.

The function $\ln_{\mathtt{lb}}(x)$ gives a lower bound for the function $\ln(x)$. The upper bound can be obtained by just adding γ to the lower bound.

$$\ln_{\mathtt{ub}}(x) \;=\; \frac{e^{-k}}{e-1}x + k - 1 + \ln(e-1) \tag{7}$$

Thus, we get a piecewise linear function for both under and over approximating the natural logarithm function.

In practice, we cannot use the above piecewise linear function because it is defined over infinitely many intervals. We pick finitely many intervals.

In our implementation, we use two parameters l, m to specify the intervals that are used to create the piecewise linear lower- and upper-bound functions. Given natural numbers l, m, our implementation uses the intervals

$$(-\infty, e^{-l}],\ [e^{-l}, e^{-l+1}],\ \ldots,\ [e^{m-1}, e^{m}],\ [e^{m}, \infty) \tag{8}$$

We use linear-interpolation based approximation on the bounded intervals and we use a sound coarse approximation on the unbounded intervals. Clearly, we can *refine* our abstraction by increasing the number of intervals; that is, by increasing the values of the parameters l, m. Automated abstraction-refinement of this kind is left for future work.

One advantage of this approximation is that the size of the intervals grows exponentially. Hence, a few intervals can approximate the logarithm for a "reasonable" range of x values. Also, note that the error is bounded and depends only on the base of the logarithm. Thus, changing the base of the logarithm provides another way to get a better approximations (refinements).

4.3 Constant Rate

We define $\phi^{c}_{\langle r,\vec{c}\rangle}$ now. Consider the linear expression $p = \vec{c}^T\vec{x}$ such that $\dot{p} = r$, where r is a nonzero constant, then we get the following time-aware relational abstraction $\phi^{c}_{\langle r,\vec{c}\rangle}$:

$$\phi^{c}_{\langle r,\vec{c}\rangle} \;:=\; (p' - p) \;=\; r(t' - t) \tag{9}$$

where t is the time variable. We can then add the conjunct $\phi^{c}_{\langle r,\vec{c}\rangle}$ to the time-aware relational abstraction of linear system.

4.4 Complex Eigenvalues

We define $\phi^{t}_{\langle \lambda,\vec{c}\rangle}$ for complex λ now. Consider a linear dynamical system $\dot{\vec{x}} = A\vec{x}$, where A contains only real (in practice, rational) entries. Let $\vec{c} := \vec{d} + \iota\vec{e}$ be a left eigenvector of A corresponding to the complex eigenvalue $\lambda := a + \iota b$; that is,

$$(\vec{d}^T + \iota\vec{e}^T)A \;=\; (a + \iota b)(\vec{d}^T + \iota\vec{e}^T)$$

Now, consider the two linear expressions

$$p(\vec{x}) \;=\; \vec{d}^T\vec{x} \qquad q(\vec{x}) \;=\; \vec{e}^T\vec{x}$$

Computing the time derivative (Lie derivative) of these two expressions, we find that the expressions p and q satisfy the differential equation $\dot{p} = ap - bq, \dot{q} = bp + aq$, and hence the closed-form solution for them is given by

$$\begin{aligned} p(\vec{x}(t)) &= re^{at}\cos(bt + \phi) \\ q(\vec{x}(t)) &= re^{at}\sin(bt + \phi) \end{aligned}$$

where r, ϕ are determined by the initial conditions (that is, values of $p(\vec{x}(0))$ and $q(\vec{x}(0))$) as follows:

$$\begin{aligned} r^2 &= p(\vec{x}(0))^2 + q(\vec{x}(0))^2 \\ \tan(\phi) &= q(\vec{x}(0))/p(\vec{x}(0)) \end{aligned}$$

We want to find linear relationships that hold between the initial value p, q of these two expressions, any future value p', q' of these two expressions, and the initial value t of the time, and the future value t' of time. We divide the task into two parts: first, we will find relationships $\phi^{\mathtt{ampl}}_{\langle\lambda,\vec{c}\rangle}$ that result from the exponential change in the amplitude (re^{at}) with time, and second, we will find relationships $\phi^{\mathtt{phase}}_{\langle\lambda,\vec{c}\rangle}$ that result from the linear change in phase ($bt + \phi$) with time. Then, the conjunct added to the time-aware relational abstraction will be

$$\phi^{t}_{\langle\lambda,\vec{c}\rangle} \;:=\; \phi^{\mathtt{ampl}}_{\langle\lambda,\vec{c}\rangle} \wedge \phi^{\mathtt{phase}}_{\langle\lambda,\vec{c}\rangle} \tag{10}$$

where $\phi^{\mathtt{ampl}}_{\langle\lambda,\vec{c}\rangle}$ and $\phi^{\mathtt{phase}}_{\langle\lambda,\vec{c}\rangle}$ will be defined below.

Relating Amplitude and Time

Let us denote $p(\vec{x}(t'))$ by p' and $p(\vec{x}(t))$ by p; and similarly for q, q'. We are given p, q and the fact that

$$({p'}^2 + {q'}^2)^{0.5} = (p^2 + q^2)^{0.5} e^{a(t'-t)} \tag{11}$$

We wish to find linear constraints that are implied by the above equation. Those linear constraints can then be added to the time-aware relational abstraction without compromising soundness.

We use the piecewise linear approximation of the natural logarithm to deal with the exponential in the above expression, but there still remains the problem of handling the other quadratic sub-expressions. We will use coarse linear lower- and upper-bounds for the quadratic sub-expressions to finally obtain the conservative linear constraint that approximates Equation 11. Specifically, the following derivation shows how we obtain the linear approximation $\phi^{\mathtt{ampl}}_{\langle\lambda,\vec{c}\rangle}$ of Equation 11:

$$\begin{array}{rrcl} & ({p'}^2 + {q'}^2)^{0.5} &=& (p^2 + q^2)^{0.5} e^{a(t'-t)} \\ \Rightarrow & a(t'-t) &=& \ln({p'}^2 + {q'}^2)^{0.5} - \ln(p^2 + q^2)^{0.5} \\ \Rightarrow & a(t'-t) &\leq& \ln_{\mathtt{ub}}(\mathtt{q}_{\mathtt{ub}}({p'}^2 + {q'}^2)^{0.5}) - \\ & & & \ln_{\mathtt{lb}}(\mathtt{q}_{\mathtt{lb}}(p^2 + q^2)^{0.5}) \quad \wedge \\ & a(t'-t) &\geq& \ln_{\mathtt{lb}}(\mathtt{q}_{\mathtt{lb}}({p'}^2 + {q'}^2)^{0.5}) - \\ & & & \ln_{\mathtt{ub}}(\mathtt{q}_{\mathtt{ub}}(p^2 + q^2)^{0.5}) \end{array}$$

Let $\phi^{\mathtt{ampl}}_{\langle\lambda,\vec{c}\rangle}$ denote the last conjunction above.

We have already defined the functions $\ln_{\mathtt{lb}}$ and $\ln_{\mathtt{ub}}$ before. We now define the functions $\mathtt{q}_{\mathtt{lb}}$ and $\mathtt{q}_{\mathtt{ub}}$ that compute linear lower and upper bound for the expression $(x^2 + y^2)^{0.5}$. In

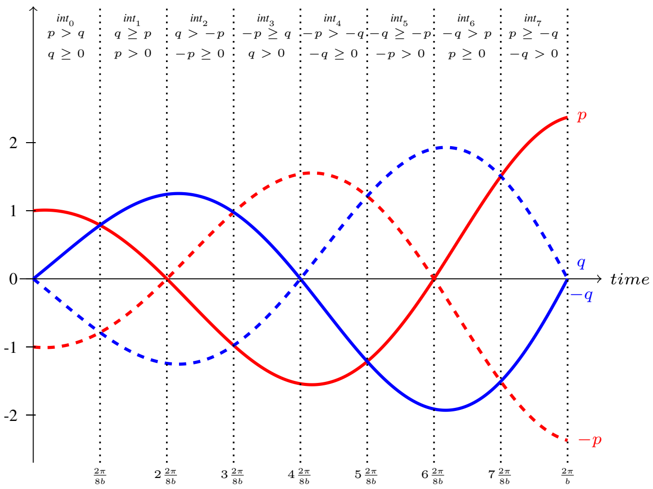

Figure 2: Extracting time elapsed information from analyzing the phase of the sinusoidal signals. The red line shows p, the blue line shows q, and the partition of the time axes based on the sign of p, q and $p \geq q$.

other words, $\mathtt{q_{lb}}$ and $\mathtt{q_{ub}}$ are piecewise linear and satisfy the following condition:

$$\mathtt{q_{lb}}((x^2+y^2)^{0.5}) \;\leq\; (x^2+y^2)^{0.5} \;\leq\; \mathtt{q_{ub}}((x^2+y^2)^{0.5})$$

We use the value $|x|+|y|$ as a linear upper bound for $(x^2+y^2)^{0.5}$ and the expression $\mathtt{max}(|x|,|y|)$ as the lower bound for $(x^2+y^2)^{0.5}$.

$$\begin{aligned}\mathtt{q_{lb}}((x^2+y^2)^{0.5}) &\;:=\; \mathtt{max}(|x|,|y|)\\ \mathtt{q_{ub}}((x^2+y^2)^{0.5}) &\;:=\; |x|+|y|\end{aligned}$$

Note that the functions $\mathtt{q_{lb}}$ and $\mathtt{q_{ub}}$ are both piecewise linear functions.

We thus get a linear and sound relationship between the current values p, q, the future values p', q' of the two linear expressions and the current and future values t, t' of time based on analyzing the change in the amplitude with time.

Relating Phase and Time

We now consider the problem of finding a linear relationship between p, p', q, q', t, t' based on analyzing the change in the phase with time. We are given p, q and the fact that

$$\begin{aligned}p' &= (p^2+q^2)^{0.5}e^{a(t'-t)}\cos(b(t'-t)+\tan^{-1}(q/p))\\ q' &= (p^2+q^2)^{0.5}e^{a(t'-t)}\sin(b(t'-t)+\tan^{-1}(q/p))\end{aligned}$$

We wish to find linear constraints that are implied by the above equation based on analyzing the phase.

Given p', q', let $\omega(p', q')$ denote the angle $b(t'-t)+\tan^{-1}(q/p)$. For example, $\omega(p, q)$ is just $b(t-t)+\tan^{-1}(q/p) = \tan^{-1}(q/p)$. We have the following relationship based on analyzing the phase.

$$b(t'-t) \;=\; \omega(p', q') - \omega(p, q)$$

Now, we need piecewise linear approximations of the ω function. The main point to note is that the phase determines the sign of p', q' and the value of $p' \geq q'$. Hence, we can get an estimate of the phase of p, q if we analyze the signs of $p, q, p-q$. Depending on the sign of $p, q, p-q$, the time axis is partitioned into infinitely many intervals, as shown in Figure 2.

Let us define the function $\omega_{\mathtt{a}}(p, q)$ that takes the values of p, q and returns a number based on the phase as illustrated in Figure 2.

$$\omega_{\mathtt{a}}(p,q) \;=\; \begin{cases} 0 & \text{if } p > q \geq 0\\ 1 & \text{if } q \geq p > 0\\ 2 & \text{if } q > -p \geq 0\\ 3 & \text{if } -p \geq q > 0\\ 4 & \text{if } -p > -q \geq 0\\ 5 & \text{if } -q \geq -p > 0\\ 6 & \text{if } -q > p \geq 0\\ 7 & \text{if } p \geq -q > 0\end{cases}$$

Now, given $\omega_{\mathtt{a}}(p, q)$ and $\omega_{\mathtt{a}}(p', q')$, we can compute bounds on $t'-t$. Specifically, if $\omega_{\mathtt{a}}(p', q') \geq \omega_{\mathtt{a}}(p, q)$, then for some natural number $n \geq 0$, it will be the case that

$$\begin{aligned}&\phi^{\mathsf{phase}}_{\langle\lambda,\vec{c}\rangle} := \exists n \geq 0 :\\ &b(t'-t) \;\geq\; 2\pi n + ((\omega_{\mathtt{a}}(p',q') - \omega_{\mathtt{a}}(p,q)) - 1)\frac{2\pi}{8} \quad \wedge\\ &b(t'-t) \;\leq\; 2\pi n + ((\omega_{\mathtt{a}}(p',q') - \omega_{\mathtt{a}}(p,q)) + 1)\frac{2\pi}{8} \quad (12)\end{aligned}$$

The value of n indicates the number of complete cycles that lie between the initial and final state.

Similarly, if $\omega_{\mathtt{a}}(p', q') \leq \omega_{\mathtt{a}}(p, q)$, then for some natural number $n \geq 0$, it will be the case that

$$\begin{aligned}&\phi^{\mathsf{phase}}_{\langle\lambda,\vec{c}\rangle} := \exists n \geq 0 :\\ &b(t'-t) \;\geq\; 2\pi n + ((\omega_{\mathtt{a}}(p',q') - \omega_{\mathtt{a}}(p,q)) + 7)\frac{2\pi}{8} \quad \wedge\\ &b(t'-t) \;\leq\; 2\pi n + ((\omega_{\mathtt{a}}(p',q') - \omega_{\mathtt{a}}(p,q)) + 9)\frac{2\pi}{8} \quad (13)\end{aligned}$$

In both cases, the constraint $\phi^{\mathsf{phase}}_{\langle\lambda,\vec{c}\rangle}$ is an infinite disjunction: there is one disjunct for each value of n. There are two different ways to handle the infinite disjunction. First, in the time-aware relational abstraction, we can introduce a new *input* variable n. When we perform infinite bounded model checking on the abstract system, the input variable n is automatically existentially quantified and all possible values for n are considered. The alternative is to replace the infinite disjunction by a finite disjunction (picking specific values for n, say $n = 0, 1, 2$) and then over-approximating the rest ($n = 3, 4, \ldots$) of the disjuncts conservatively. Our implementation uses the latter approach. We have a parameter that fixes the range of values we use for n. We will call the parameter n subsequently.

4.5 Correctness

We can now formally state the correctness of the procedure outlined above for creating a time-aware relational abstraction. First, we note the following immediate fact.

LEMMA 1. *If $\phi_1(\vec{x}, \vec{x}')$ and $\phi_2(\vec{x}, \vec{x}')$ are two relational abstractions of the same system, then $\phi_1(\vec{x}, \vec{x}') \;\wedge\; \phi_2(\vec{x}, \vec{x}')$ is also a relational abstraction of that system.*

Thus, for a given linear system $\dot{\vec{x}} = A\vec{x} + \vec{b}$, let Λ denote all pairs $\langle\lambda, \vec{c}\rangle$ s.t. $\frac{d\vec{c}^T\vec{x}}{dt} = \lambda\vec{c}^T\vec{x}$, and let R denote all pairs $\langle r, \vec{c}\rangle$ s.t. $\frac{d\vec{c}^T\vec{x}}{dt} = r$ for some real number r. Then, we can construct the following time-aware relational abstraction for $\dot{\vec{x}} = A\vec{x} + \vec{b}$:

$$Trans_C(q)^t \quad := \quad \bigwedge_{\langle \lambda,\vec{c}\rangle \in \Lambda} \phi^t_{\langle \lambda,\vec{c}\rangle} \quad \wedge \quad \bigwedge_{\langle r,\vec{c}\rangle \in R} \phi^c_{\langle r,\vec{c}\rangle} \tag{14}$$

where depending on whether λ is real or complex, $\phi^t_{\langle \lambda,\vec{c}\rangle}$ is defined in Equation 4 or Equation 10, and $\phi^c_{\langle r,\vec{c}\rangle}$ is defined in Equation 9.

The following theorem states the correctness of this construction.

THEOREM 1. *Let $\dot{\vec{x}} = A\vec{x} + \vec{b}$ be the continuous dynamics of the location $q \in Q$ of H. Then, $Trans_C(q)^t$ defined in Equation 14 is a relational abstraction for the continuous dynamics $\mu(q)$.*

PROOF. (Sketch) We have to prove that $\forall s_i, s_{i+1} \in \mathbb{R}^n, \delta \in \mathbb{R}$ s.t. $s_{i+1} = f_q(s_i, \delta)$, $s, s' \models Trans_C(q)^t$, where $f_q : \mathbb{R}^{n+1} \to \mathbb{R}^n$ is the solution to the flow condition $\mu(q)$.

We will prove that s, s' is a model for each conjunct $\phi^t_{\langle \lambda,\vec{c}\rangle}$ and $\phi^c_{\langle r,\vec{c}\rangle}$. Then, the proof will follow from Lemma 1.

The proof for the constant rate case, i.e. $s, s' \models \phi^c_{\langle r,\vec{c}\rangle} := p' - p = r\delta$, follows directly from the fact that $\dot{p} = r$.

We prove that $s, s' \models \phi^t_{\langle \lambda,\vec{c}\rangle}$ considering the cases when λ is real or complex. Suppose $\lambda \in \mathbb{R}$ and $\vec{c} \in \mathbb{R}^n$. If $\lambda > 0$, then $\phi^t_{\langle \lambda,\vec{c}\rangle} = \psi(p, p')$, where $p = \vec{c}^T\vec{x}$. We have that $s, s' \models \psi(p, p')$ because:
(1) If $s \models p = 0$, then by the explicit solution of p (Equation 3), we also have $p' = 0$ and thus $s, s' \models 0 = p = p'$.
(2) If $s \models 0 < p$, then by Equation 3 we have $s, s' \models 0 < p \leq p'$ and $s, s' \models \ln(p') - \ln(p) = \lambda(t' - t)$. Thus, $s, s' \models \ln_{\mathtt{lb}}(p') - \ln_{\mathtt{ub}}(p) \leq \lambda(t' - t) \leq \ln_{\mathtt{ub}}(p') - \ln_{\mathtt{lb}}(p)$.
(3) The proof for the case when $s \models 0 > p$ is similar.
The proof for $\lambda < 0$ is similar to the one for $\lambda > 0$. Similarly, for the case when λ is complex, we can show that $\phi^t_{\langle \lambda,\vec{c}\rangle}$ is always a conservative (piecewise linear) approximation of the exact relation between p, p', q, q' and time t, t' variables. □

5. RELATED WORK

Verification of hybrid systems has been performed applying different techniques (see [2] for a recent survey).

Among these techniques there are symbolic reachability and deductive verification. Symbolic reachability [20, 18, 34] consists of computing the reachable set of states, which will be used to prove the system safety. In deductive verification [26, 27, 31], the user interacts with a theorem prover to produce a proof of correctness. Both approaches may handle properties that involve predicates over time. However, we stress that the main goal of the time-aware abstraction is to provide a more precise relational abstraction, widening its applicability in terms of hybrid systems and properties that can be verified. Also, since the timed-aware abstraction is a relational abstraction, it separates the reasoning task on the continuous dynamics from the verification task on the infinite-state transition system.

Another viable technique to the hybrid systems verification consists of computing an abstraction of the system, which may be subsequently verified [3, 13, 32]. Relational abstraction falls in this class of techniques. There exists several ways to compute relational abstractions. However, the current techniques used to compute a relational abstraction [28, 33, 35] are not suitable to verify real-time properties or to analyze a network of hybrid systems.

The template-based relational abstraction [28] does not capture the relation among time and the continuous variables of the system. Moreover, the problem of finding the coefficients of the templates requires the use of non-linear real arithmetic, which may be solved using expensive real quantifier elimination techniques [16, 8] or SMT solvers which handle non-linear real arithmetic [23].

The timed relational abstraction [35] is suitable to analyze control systems that sample the physical plant at fixed time intervals. In that case, the relation is precise over time since it relates the current values with the future values of the variables after a continuous transition of fixed duration (the sampling interval). However, since the continuous evolution has a fixed duration, the relation does not capture the possible evolution of the system for different intervals of time.

The eigenstructure-based relational abstraction [28, 33], as shown in Section 3, is not precise enough to capture the relation between the continuous variables and the time elapsed in a continuous transitions and, in general, between the variables which evolve with different "rates". Hence, it is not suitable to analyze properties which predicate about time or about other variables with a piecewise constant derivative (like drifted clocks or resources which evolve with a non-deterministic but bounded derivative). The time-aware abstraction increases the precision of the eigenstructure-based abstraction.

Other works focuses on the analysis of hybrid systems using Satisfiability Modulo Theory (SMT) solvers [4, 1]. However, these approaches are currently limited, since they may only handle a subset of Linear Hybrid Systems [11]. Relational abstraction handles all the class of linear hybrid systems, even if in an approximate way. The time-aware abstraction also widens the applicability of the verification techniques developed for networks of linear hybrid automata, like scenario verification [12], to Linear Hybrid Systems.

Since the produced abstraction is an infinite-state transition system, it can be verified by SMT-based verification techniques such as k-induction [29] or ic3 [22, 9]. The time-aware abstraction is orthogonal to these approaches, since it only abstracts a dynamical system. Our approach will benefit from any improvement in the performance of the verification algorithms for infinite state transition systems and of SMT solvers.

6. EXPERIMENTS

We describe some initial results obtained by using an implementation of time-aware relational abstraction in HybridSal. We used two examples for our initial evaluation: a small, but challenging, example of a PID controller, and a medium-size example of active suspension.

Consider a proportional-integral-derivative (PID) controller (taken from an online tutorial at `ctms.engin.umich.edu/CTMS/`) that is used to control a simple mass, spring, and damper problem. The modeling equation of the mass, spring, and damper system (plant) is

$$M\ddot{x} + b\dot{x} + kx \quad = \quad F$$

where $M = 1kg$, $b = 10Ns/m$, $k = 20N/m$ are the given parameters of the plant, and F is the (controllable) force. Suppose the goal is to make the plant reach a steady state where $x = 1$ with the some requirements on the overshoot and rise time (that we will precisely specify later). Suppose the desired trajectory $r(t)$ for reaching the steady state $x = 1$ is a step function: at time $t = 0$, we want the system to go from its initial state (say, $x = 0$) to its steady state $x = 1$; that is, $r(t) = 0$ for $t < 0$ and $r(t) = 1$ for $t \geq 0$.

Let us assume that we are given a PID controller that has gains $K_p = 350, K_i = 300, K_d = 50$. The equation describing the composed controller and plant system is

$$M\ddot{x} + b\dot{x} + kx \quad = \quad K_d(r \dot{-} x) + K_p(r - x) + K_i \int (r - x)$$

Note that $r - x$ is the error in tracking the desired trajectory r. Substituting the parameters given above in this equation, we get the following state-space model of the controller and the plant subsystem. (Since r is not differentiable at $t = 0$, we have used $\dot{r} = 0$ here).

$$\begin{aligned} \frac{dxint}{dt} &= x \\ \frac{dx}{dt} &= xder \\ \frac{dxder}{dt} &= -60 * xder - 370 * x - 300 * xint + 350 + 300 * \mathtt{t} \\ \frac{d\mathtt{t}}{dt} &= 1 \end{aligned}$$

where x, $xder$ (denoting $\dot{x}$), $xint$ (denoting $\int x$), and t are the four state variables.

For this model, we wish to check the following rise time requirement: after t=0.5 units, x reaches within 90% of its steady-state value. We check a stronger variant of this requirement, namely

$$\mathtt{G}(\mathtt{t} > 0.5 \Rightarrow x \geq 0.9 \wedge x \leq 1.1)$$

which says that it is always true that whenever time is greater than 0.5, then x is in the $[0.9, 1.1]$ interval.

In Table 1, we present the results of analyzing the rise-time requirements for different controllers. We pick three controllers: a PD controller, a PI controller and a PID controller. For each of the three controllers, we consider two variants: one in which the integral term goes through a saturation block (suffixed with "Sat") and one in which there is no such saturation block. (In PD, the coefficient of the integral term is zero, but the integral is still computed and hence, saturated in PDSat.) We analyze the above six system models using time-agnostic relational abstraction (Col 2-3), and using time-aware relational abstraction (Col 4-15). The analysis with time-aware relational abstraction is carried out with four different settings of the three parameters: l,m,n. Recall that precision of abstraction improves as we increase these parameters (see Equations 8 for l, m, and text following Equation 13 for n).

We note that we cannot prove the rise-time requirement for any of the system models using only time-agnostic relational abstraction. Using time-aware abstraction, we also cannot prove the rise-time requirement for any of the models unless we pick $l \geq 3$. Note that the PD and PID systems (and their saturated counterparts) satisfy the desired rise-time requirement, whereas the PI system does not. The bounded model checking time increases as we increase the precision of the abstraction (which is achieved by using larger values for the parameters l, m, n).

Note that we perform bounded model checking on the computed time-aware abstractions. In general, failure to find a counter-example in a bounded run does not imply the validity of the property. But, for single mode hybrid systems (such as PD, PI and PID), if there is no depth 1 counter example, then the property is valid. This is because (time-agnostic and time-aware) relational abstractions over-approximate unbounded time reach sets (for each mode). For hybrid systems with multiple modes, a proof using k-induction is required to prove a property.

As a second slightly more complex example, we applied the time-aware relational abstraction technique to verifying bounded deflection in a model of an active suspension. The model was derived from the paper [17]. The 1/4-car active suspension model consists of 5 state variables. There are four modes in the hybrid automaton. The modes arise from gain scheduling – essentially different parameters are used in the controller in different regions of the state space. The correctness property stated that the deflection of the suspension always remains within a safe interval. Time-agnostic relational abstraction is unable to verify this safety property; it always produces a spurious counterexample. However, the time-aware abstraction constructed using parameter values $l = 2, m = 2, n = 2$, was sufficient to show that there were no counterexamples up to depth 4.

7. CONCLUSION

In this paper, we have proposed a technique to compute time-aware relational abstractions for linear hybrid systems, which are suitable to prove time-related properties. Our approach to compute a time-aware abstraction is based on the analysis of the eigenstructure of the matrix of the dynamical system, and on the piecewise linear approximation of non-linear functions.

In contrast to the previous works, time-aware relational abstraction captures (an approximation of) the time elapsed during a continuous transition. Also, as a consequence of the more precise relation over time, the overall abstraction is more precise. We show that the increased precision provided by time-aware abstraction is necessary to prove time-related properties on some interesting case studies.

As future work, time-aware relational abstraction can be applied to networks of hybrid automata [10, 25], where the information about time is needed to precisely compose the abstraction of the automata. Also, techniques for abstraction-refinement, based on changing the base of the logarithm or increasing the number of intervals, can be developed.

8. REFERENCES

[1] E. Ábrahám, B. Becker, F. Klaedtke, and M. Steffen. Optimizing Bounded Model Checking for Linear Hybrid Systems. In *VMCAI*, pages 396–412, 2005.

[2] R. Alur. Formal verification of hybrid systems. In *EMSOFT*, pages 273–278, 2011.

[3] R. Alur, T. Dang, and F. Ivancic. Predicate abstraction for reachability analysis of hybrid systems. *ACM Trans. Embedded Comput. Syst.*, 5(1):152–199, 2006.

[4] G. Audemard, M. Bozzano, A. Cimatti, and R. Sebastiani. Verifying Industrial Hybrid Systems with MathSAT. *ENTCS*, 119(2):17–32, 2005.

[5] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability Modulo Theories. In *Handbook of Satisfiability*, pages 825–885. 2009.

Model	RA t_1, t_2		time-aware RA parameters, t_1, t_2											
PD	0.6	CE 0.4	2,0,0	1.1	CE 0.2	3,0,0	0.4	CE 0.2	0,2,0	0.4	CE 0.2	4,2,2	0.6	P 0.6
PI	0.4	CE 0.1	2,0,0	0.6	CE 0.6	3,0,0	0.4	CE 0.2	0,2,0	1.2	CE 0.6	4,2,2	0.5	CE 0.2
PID	1.1	CE 0.1	2,0,0	0.4	CE 0.2	3,0,0	0.4	P 0.2	0,2,0	0.8	CE 0.6	4,2,2	1.2	P 0.8
PDSat	1.5	CE 0.4	2,0,0	1.5	CE 0.7	3,0,0	0.5	CE 0.2	0,2,0	0.8	CE 0.7	4,2,2	0.8	P 0.7
PISat	0.6	CE 0.2	2,0,0	1.2	CE 0.3	3,0,0	0.6	CE 0.3	0,2,0	0.9	CE 1.1	4,2,2	0.6	CE 0.5
PIDSat	1.5	CE 0.4	2,0,0	1.2	CE 1.1	3,0,0	0.6	P 0.3	0,2,0	0.8	CE 0.35	4,2,2	1.3	P 1.6

Table 1: Results on verifying feedback PD, PI and PID controllers, with and without saturation, using different parameters for the time-aware relational abstraction. Model names with suffix "sat" are versions that have saturation applied on the integral term. Time t_1 is the time to create the abstraction and time t_2 (both in seconds) is the time to perform infinite bounded model checking (inf-bmc) on the abstract model. The return value of inf-bmc is either a counterexample (CE) or no counterexample (P). For single mode hybrid systems, such as, PD,PI,PID, the absence of CE is equivalent to the property being "proved".

[6] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic Model Checking without BDDs. In *TACAS*, pages 193–207, 1999.

[7] A. R. Bradley and Z. Manna. *The calculus of computation - decision procedures with applications to verification*. Springer, 2007.

[8] C. Brown. QEPCAD B: A program for computing with semi-algebraic sets using CADs. *SIGSAM BULLETIN*, 37:97–108, 2003.

[9] A. Cimatti and A. Griggio. Software Model Checking via IC3. In *CAV*, pages 277–293, 2012.

[10] A. Cimatti, S. Mover, and S. Tonetta. HYDI: a language for symbolic hybrid systems with discrete interaction. In *EUROMICRO-SEAA*, pages 275–278, 2011.

[11] A. Cimatti, S. Mover, and S. Tonetta. A quantifier-free SMT encoding of non-linear hybrid automata. In *FMCAD*, pages 187–195, 2012.

[12] A. Cimatti, S. Mover, and S. Tonetta. SMT-based scenario verification for hybrid systems. *Formal Methods in System Design*, 42(1):46–66, 2013.

[13] E. M. Clarke, A. Fehnker, Z. Han, B. H. Krogh, J. Ouaknine, O. Stursberg, and M. Theobald. Abstraction and counterexample-guided refinement in model checking of hybrid systems. *Int. J. Found. Comput. Sci.*, 14(4):583–604, 2003.

[14] L. de Moura, H. Rueß, and M. Sorea. Bounded Model Checking and induction: from refutation to verification. In *CAV*, pages 14–26, 2003.

[15] L. M. de Moura, S. Owre, H. Rueß, J. M. Rushby, N. Shankar, M. Sorea, and A. Tiwari. SAL 2. In *CAV*, pages 496–500, 2004.

[16] A. Dolzmann and T. Sturm. REDLOG Computer Algebra Meets Computer Logic. *ACM SIGSAM Bulletin*, 31:2–9, 1996.

[17] I. Fialho and G. J. Balas. Road adaptive active suspension design using linear parameter-varying scheduling. *IEEE Trans. on Control Sys. Tech.*, 10(1), 2002.

[18] G. Frehse, C. L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable Verification of Hybrid Systems. In *CAV*, pages 379–395, 2011.

[19] T. A. Henzinger. The Theory of Hybrid Automata. In *LICS*, pages 278–292. IEEE CS, 1996.

[20] T. A. Henzinger, P. Ho, and H. Wong-Toi. HYTECH: A Model Checker for Hybrid Systems. *STTT*, 1(1-2):110–122, 1997.

[21] R. Hilscher. Surprises about some elementary functions: uniform linear approximations. Technical report, 2000.

[22] K. Hoder and N. Bjørner. Generalized Property Directed Reachability. In *SAT*, pages 157–171, 2012.

[23] D. Jovanovic and L. M. de Moura. Solving Non-linear Arithmetic. In *IJCAR*, pages 339–354, 2012.

[24] G. Lafferriere, G. J. Pappas, and S. Yovine. Symbolic Reachability Computation for Families of Linear Vector Fields. *J. Symb. Comput.*, 32(3):231–253, 2001.

[25] L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli. Decentralized cooperative policy for conflict resolution in multivehicle systems. *IEEE Transactions on Robotics*, 23(6):1170–1183, 2007.

[26] A. Platzer. Differential Dynamic Logic for Hybrid Systems. *J. Autom. Reasoning*, 41(2):143–189, 2008.

[27] S. Prajna and A. Jadbabaie. Safety Verification of Hybrid Systems Using Barrier Certificates. In *HSCC*, pages 477–492, 2004.

[28] S. Sankaranarayanan and A. Tiwari. Relational Abstractions for Continuous and Hybrid Systems. In *CAV*, pages 686–702, 2011.

[29] M. Sheeran, S. Singh, and G. Stålmarck. Checking Safety Properties Using Induction and a SAT-Solver. In *FMCAD*, pages 108–125, 2000.

[30] M. Sorea. Bounded Model Checking for Timed Automata. In *Electronic Notes in Theoretical Computer Science*, page 2002. Elsevier, 2002.

[31] T. Sturm and A. Tiwari. Verification and synthesis using real quantifier elimination. In *ISSAC*, pages 329–336, 2011.

[32] A. Tiwari. Abstractions for hybrid systems. *Formal Methods in System Design*, 32(1):57–83, 2008.

[33] A. Tiwari. HybridSAL Relational Abstracter. In *CAV*, pages 725–731, 2012.

[34] A. Tiwari and G. Khanna. Nonlinear Systems: Approximating Reach Sets. In *HSCC*, pages 600–614, 2004.

[35] A. Zutshi, S. Sankaranarayanan, and A. Tiwari. Timed Relational Abstractions for Sampled Data Control Systems. In *CAV*, pages 343–361, 2012.