Timed Relational Abstractions For Sampled Data Control Systems

Aditya Zutshi[1], Sriram Sankaranarayanan[1] and Ashish Tiwari[2].

1. University of Colorado, Boulder, CO. {aditya.zutshi,srirams}@colorado.edu
2. SRI International, Menlo Park, CA. ashish.tiwari@sri.com

Abstract. In this paper, we define timed relational abstractions for verifying sampled data control systems. Sampled data control systems consist of a plant, modeled as a hybrid system and a synchronous controller, modeled as a discrete transition system. The controller performs control actions on the plant by periodically sampling the state of the plant. The correctness of the system depends on the controller design as well as an appropriate choice of its sampling period.
Our approach constructs a timed relational abstraction of the hybrid plant by replacing the continuous plant dynamics by relations. These relations map a state of the plant to states reachable within the sampling time period. We present techniques for building timed relational abstractions, while taking care of discrete transitions that can be taken by the plant between samples. The resulting abstractions are better suited for the verification of sampled data control systems. The abstractions focus on the states that can be observed by the controller at the sample times, while abstracting away behaviors between sample times conservatively. As the abstractions are discrete, infinite-state transition systems, conventional verification tools can be used. We use k-induction to prove safety properties and bounded model checking (BMC) to find potential falsifications. We present our idea, its implementation and results on many benchmark examples.

1 Introduction

We present techniques for verifying safety properties of sampled data control systems using timed relational abstractions. Sampled data control systems consist of a discrete controller that periodically senses the state of a continuous physical plant, and actuates by setting inputs or sending control commands to the plant. Sampled data control systems are quite common in practice. Complex (network) control systems often involve many control tasks that are scheduled periodically, with each task controlling a different aspect of the plant. The cadencing of these tasks to enforce the safety and stability of the system is an important problem. The choice of sampling period is crucial: a small sampling time can place infeasible constraints on the scheduling policy, whereas large sampling times can cause instabilities or safety violations.

In this paper, we consider a simple and natural model of a sampled data control system. The controller is modeled by an infinite state (linear) transition system. It communicates synchronously with a plant modeled by an affine hybrid automaton. The controller runs periodically with a fixed sampling period $T_s > 0$. At each time period, the controller senses the state of the plant and performs controller actions that may include (a) setting control input signals for the plant, and (b) commanding the plant to

execute a controlled discrete transition, resulting in an instantaneous jump and a mode change in the plant.

Our verification approach uses the idea of timed relationalization, extending the idea of untimed relational abstractions [33]. A timed relational abstraction considers the set of states of the plant that are potentially observable by the controller at the sample times, while safely abstracting away all the intermediate states. To this end, we build relations that map a state of the plant at the beginning of a sampling period to states that can be reached at the end. Using these relations, the entire plant can be safely abstracted away by a discrete, infinite-state transition system. This system is composed together with the controller to yield an overall discrete system that can be analyzed by existing tools such as k-induction [34], bounded model-checking [4] and abstract interpretation [8, 18], while exploiting advances in abstract domains, SAT and SMT solvers.

There are two key challenges in constructing the timed abstraction: (a) dealing with the continuous dynamics inside a mode, and (b) handling autonomous transitions that can be taken by the plant between two sampling periods. For systems with affine dynamics the former problem is solved by computing a matrix exponential for the matrices defining the dynamics in each mode [22]. However, the numerical exponential computation is potentially unsound due to round-off and truncation errors. Likewise, solving for autonomous transitions involves computing a symbolic matrix exponential to deal with the unknown switching times for each transition. To solve both problems, we exploit advances in interval arithmetic to compute guaranteed enclosures to the matrix exponential [23, 26, 15, 5]. This yields interval linear relations. We then use a *template*-based mechanism using SMT solvers to abstract the resulting interval linear relations in terms of relations expressible in linear arithmetic.

We have implemented our approach to relationalization and present an extensive evaluation over a set of benchmark systems. Our evaluation performs a relational abstraction of the plant using the techniques described in this paper. The resulting abstraction is analyzed using the SAL tool-set from SRI [32, 37]. The results of our evaluation are quite promising: we show that our techniques can successfully handle complex sampled data control systems efficiently and soundly. Our implementation, the data from our experiments along with an extended version of the paper will be available online [1]. We now discuss the related work.

Relational Abstractions: Relational abstractions have been used primarily for checking liveness properties [3, 28]. There are many subtle distinctions between the various forms of relational abstractions used. Transition invariants [28], used in termination proofs, relate the current state to *any* previous state at a given program location. Likewise, progress invariants relate the current state and the *immediately* previous state at a given location [16]. Podelski and Wagner provide a verification procedure for (region) stability properties of hybrid systems [29], wherein they derive *binary reachability relations* over trajectories of a hybrid system, similar in spirit to a relational abstraction. Note that Podelski and Wagner use a hybrid system reachability tool to compute their abstractions in the first place. The techniques in this paper and our previous work [33] are meant to solve the reachability problem using these relations.

[1] `http://systems.cs.colorado.edu/research/cyberphysical/relational`

Our previous work explored the idea of abstracting the dynamics inside each discrete mode of a hybrid automaton by an *untimed relational abstraction* [33]. The relational abstractions presented here capture the relationship between the current state at time $t = t_0$ and any state reachable at time $t = t_0 + T_s$ units. The consideration of the sampling time T_s is essential for verifying sampled data control systems. Furthermore, it is also important to note that unlike our previous relational abstraction, it is essential for the abstraction presented here to account for the plant's discrete transitions that can be taken in the time interval $t \in [t_0, t_0 + T_s]$.

Abstractions of Hybrid Systems: *Discrete abstractions* have been studied for hybrid systems. These include predicate abstraction [1, 36] and invariance-based abstractions [25]. The use of counterexample-guided abstraction refinement for iterative refinement has also been investigated in the past [1, 6]. In this paper, the proposed abstraction yields a discrete but infinite state system.

Hybridization is a technique for converting nonlinear systems into affine systems by subdividing the invariant region into numerous subregions and approximating the dynamics as a hybrid system by means of a linear differential inclusion in each region [19, 2, 9]. However, such a subdivision is expensive as the number of dimensions increases and often infeasible if the invariant region is unbounded.

Reachability Analysis: Reasoning about the reachable set of states for flows of nonlinear systems is an important primitive that is used repeatedly in the analysis of nonlinear hybrid systems. This has been addressed using a wide variety of techniques in the past, including algebraic techniques, interval analysis, constraint propagation, and Bernstein polynomials [30, 24, 27, 31, 10].

Synchronous Systems: Techniques for verifying synchronous system models, with piecewise constant continuous dynamics, have been studied in the past, notably by Halbwachs et al. [18] and as part of the NBAC tool by Jeannet et al. [20]. Our work considers a synchronous controller with affine hybrid plants. Furthermore, we consider the idea of an up front relationalization of the plant dynamics, enabling a verification procedure to focus purely on discrete systems.

Motivating Examples: We discuss two simple motivating examples that clearly illustrate the need for verification of sampled data control systems.

Consider a proportional-integral (PI) controller defined by $u' := -30x - y$ composed with a plant defined by $\dot{x} = 5x + u, \quad \dot{y} = x$. With a period $T_s = 0.1s$, the controller is able to stabilize the plant but fails to do so with a period $T_s = 0.5s$.

Consider an inverted pendulum controller:

$$u' := \begin{cases} -16, & y \geq 2 \ \vee \ 16x - y \leq -10 \\ 16, & y \leq -2 \ \vee \ 16x - y \geq 10 \\ u, & \text{otherwise} \end{cases} .$$

The linearized plant has the dynamics $\dot{x} = y \quad \dot{y} = 20x + 16y + 4u$. If the controller is implemented in the continuous domain, it results in a stable system. However, a digital implementation, regardless of the sampling period, is unable to stabilize the pendulum.

2 Sampled Data Control Systems

Let $\mathbb{R}$ denote the set of real numbers. We use $\boldsymbol{a}, \ldots, \boldsymbol{z}$ with subscripts to denote (column) vectors and $A, \ldots, Z$ to denote matrices. For a $m \times n$ matrix A, the row vector A_i, for $1 \leq i \leq m$, denotes the i^{th} row.

We discuss models for sampled data control systems. Figure 1 shows the schematic diagram for a such a control system consisting of a discrete controller communicating with a hybrid plant. The controller has a time period $T_s > 0$. Every T_s time units, the controller senses the state of the hybrid plant and issues commands to control the plant. The commands can take the form of (a) events enabling a discrete transition of the plant, or (b) values for control inputs that are assumed to be held constant throughout the sample time period. We model the controller as a discrete transition system [21].

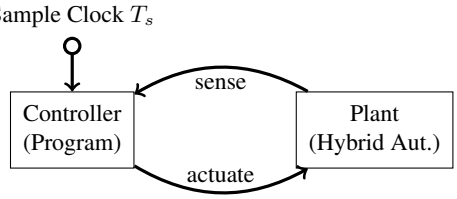

Fig. 1. Schematic for a Sampled Data Control System.

Definition 1 (Discrete Transition System). *A discrete transition system Π is a tuple $\langle L, \boldsymbol{x}, \mathcal{T}, \ell_0, \Theta \rangle$ wherein, L is a finite set of* discrete locations*; $\boldsymbol{x} : (x_1, \ldots, x_n)$ is a set of variables with variable x_i of type* $\mathsf{type}(x_i)$*; $\mathcal{T}$ is a set of discrete transitions; $\ell_0 \in L$ is the initial location; and $\Theta[\boldsymbol{x}]$ is an assertion capturing the initial values for $\boldsymbol{x}$.*

Each transition $\tau \in \mathcal{T}$ is of the form $\langle \ell, m, \rho_\tau \rangle$, wherein ℓ is the pre-state of the transition and m is the post-state. The relation $\rho_\tau[\boldsymbol{x}, \boldsymbol{x}']$ represents the transition relation over current state variables $\boldsymbol{x}$ and next state variables $\boldsymbol{x}'$.

We now discuss the overall model for the plant as a hybrid automaton with controlled and uncontrolled transitions.

Definition 2 (Plant Model). *A plant $\mathcal{P}$ is an extended hybrid automaton described by a tuple $\langle \boldsymbol{x}, \boldsymbol{u}, Q, \mathcal{F}, \mathcal{X}, \mathcal{T}, q_0, X_0 \rangle$, wherein,*

- *$\boldsymbol{x} : (x_1, \ldots, x_n)$ denotes the continuous state variables, and $\boldsymbol{u} : (u_1, \ldots, u_m)$ the control inputs,*
- *Q is a finite set of discrete modes. $q_0 \in Q$ is the initial mode and X_0 the initial set of states.*
- *$\mathcal{F}$ maps each discrete mode $q \in Q$ to an ODE $\frac{d\boldsymbol{x}}{dt} = F_q(\boldsymbol{x}, \boldsymbol{u}, t)$.*
- *$\mathcal{X}$ maps each discrete model $q \in Q$ to a mode invariant $\mathcal{X}(q) \subseteq \mathbb{R}^n$.*
- *$\mathcal{T}$ represents a set of discrete transitions. Each transition $\tau \in \mathcal{T}$ is a tuple $\langle s, t, \gamma, U \rangle$ wherein s, t represent the pre- and post- state respectively. $\gamma[\boldsymbol{x}]$ is the transition guard assertion, and U maps each variable $x_i \in \boldsymbol{x}$ to an update function $U_i(\boldsymbol{x})$. The transition relation for τ is defined as $\rho_\tau(\boldsymbol{x}, \boldsymbol{x}') : \gamma(\boldsymbol{x}) \wedge \boldsymbol{x}' = U(\boldsymbol{x})$.*
- *We partition the transitions in $\mathcal{T}$ as* autonomous transitions $\mathcal{T}_{aut}$ *and* controlled transitions $\mathcal{T}_{ctrl}$*. Autonomous transitions can be taken by the plant non-deterministically, whenever enabled. On the other hand, controlled transitions are taken upon an explicit command by the controller.*

The state of the plant is a tuple $(q, \boldsymbol{x}, \boldsymbol{u})$ consisting of the current mode q, state values $\boldsymbol{x}$ and controller input $\boldsymbol{u}$. Note that the control input is set at the beginning of a time period, and is assumed to remain constant throughout the period.

The overall sampled data control system is a tuple $\langle \mathcal{C}, \mathcal{P}, \mu, T_s \rangle$ of a discrete controller transition system $\mathcal{C}$, a hybrid plant model $\mathcal{P}$ and a mapping from variables in $\mathcal{C}$ to control inputs $\boldsymbol{u}$ of $\mathcal{P}$. A given sampling time $T_s > 0$ specifies the periodicity of the controller execution. The state of the system is represented by the joint state of the plant σ_P and σ_C of the controller. We assume that the computations of the controller take zero (or negligible time) compared to the sampling period. Furthermore, we assume that the commands issued by the controller are in the form of an input $\boldsymbol{u}$ for the next time period, and/or a command to execute a discrete transition . Finally, to avoid considering improbable "race conditions", we assume that the plant itself may not execute autonomous discrete transitions at sample time instances when the controller executes [2]. The overall system evolves in one of two ways:

1. At sample times $t = nT_s$ for $n \in \mathbb{Z}$, a controlled transition is taken based on the current state of the plant and the controller. The transition updates the controller state, the values of the plant inputs and can also command the plant to execute a discrete transition out of its current mode.
2. Between two sample times $t \in (nT_s, (n+1)T_s)$, the state of the plant evolves according to its current mode q, continuous variables $\boldsymbol{x}$ and input $\boldsymbol{u}$. If an autonomous transition $\tau \in \mathcal{T}_{aut}$ is enabled, then it may be non-deterministically executed by the plant, possibly changing the plant's state instantaneously.

A plant is *affine* iff (a) for each discrete mode q, the dynamics are of the form $\frac{d\boldsymbol{x}}{dt} = A_q\boldsymbol{x} + B_q\boldsymbol{u} + \boldsymbol{b}_q$, (b) the initial condition Θ and guards γ_τ for each transition τ, are linear arithmetic formulae, and (c) the update functions U_τ are affine.

Why Autonomous Transitions? The ability to model autonomous transitions is important for two main reasons: (a) real life plants are often multi modal with mode changes that can be affected by environmental factors such as user inputs, disturbance inputs, failures or other exceptional situations that are not within control; and (b) even though we assume that controller commands effect immediately, often there may be delays. Autonomous transitions can be used to model these delays if they can be bounded.

3 Relationalization

In this section, we discuss the notion of timed relationalizations for plants in a sampled data control system. The basic idea behind relationalization is to build a relation $R_P(q', \boldsymbol{x}', q, \boldsymbol{x}, \boldsymbol{u})$ of all possible pairs of states $(q', \boldsymbol{x}')$ and $(q, \boldsymbol{x})$ such that (a) the plant is in the state $(q, \boldsymbol{x})$ at the start time $t = t_0$, (b) it reaches the state $(q', \boldsymbol{x}')$ at time $t = t_0 + T_s$, and (c) $\boldsymbol{u}$ is the constant controller input for $t \in (t_0, t_0 + T_s]$. Note that the discrete modes q, q' may be different, depending on whether an autonomous transition is taken by the plant between two samplings.

[2] This assumption can be relaxed to allow such simultaneous executions, provided the plant and the controller do not attempt to update the same state variable.

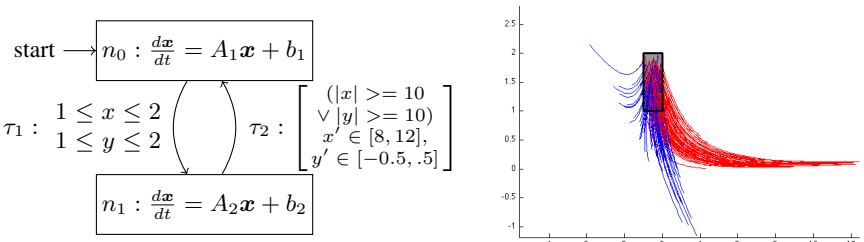

Fig. 2. A simple affine hybrid automaton with an autonomous transition τ_1 and controlled transition τ_2. Some sample trajectories of the automaton are shown with the autonomous transition being taken. The red colored trajectories belong to mode n_0 and the blue colored trajectories to mode n_1. The guard set is shown in thick lines.

Let us suppose a relation R_P can be built that can characterize all pairs $(q', \boldsymbol{x}')$ that a controller can observe at the next time step, given that $(q, \boldsymbol{x})$ was observed at the current time step and $\boldsymbol{u}$ was the control input. As a result, we may construct a purely discrete abstraction of the sampled data control system wherein the behavior of the plant between two samplings is entirely captured by R_P. Therefore, the resulting discrete transition system can be verified using a host of approaches for verification of discrete programs. Furthermore, since our goal is to perform safety verification, we do not need to compute the exact relation R_P, but only an over-approximation of it.

We will now describe techniques for constructing timed relational abstractions.

Example 1. Consider the hybrid plant model shown in Figure 2 with two state variables x, y and no control inputs. The matrices defining the dynamics are

$$A_1 : \begin{pmatrix} -1.5 & 1.2 \\ 1.3 & 0.2 \end{pmatrix} \quad b_1 : \begin{pmatrix} 1.0 \\ -0.5 \end{pmatrix} \quad A_2 : \begin{pmatrix} 2 & 1.2 \\ 0.1 & -3.6 \end{pmatrix} \quad b_2 : \begin{pmatrix} -0.6 \\ -0.6 \end{pmatrix}.$$

There are two modes n_0 and n_1 with an autonomous transition τ_1 from n_0 to n_1 and a controlled transition τ_2 from n_1 back to n_0. Relationalization of this automaton will need to consider 3 cases: (a) the automaton remains entirely inside the mode n_0 during a sample interval, (b) the automaton remains entirely inside mode n_1 and (c) the automaton switches from mode n_0 to n_1 sometime during a sampling interval.

We first discuss relational abstraction for the case when the plant remains in some mode q during a sampling period without any autonomous transitions taken in between. This situation is abstracted by a relation $R_q(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ that relate all plant states $(q, \boldsymbol{x})$ at some time t and state $(q, \boldsymbol{x}')$ at time $t + T_s$ with control input $\boldsymbol{u}$. The resulting R_q for each $q \in Q$ will form a disjunct in the overall relation R_P for the plant.

Definition 3 (Timed Relational Abstraction). *Consider a continuous system specified by a time invariant ODE $\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x}, \boldsymbol{u})$ for $\boldsymbol{x} \in X$ and control inputs $\boldsymbol{u} \in U$.*

A relation $R(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ is a timed relational abstraction *with sample time T_s of the continuous system iff for all time trajectories $\boldsymbol{x}(t)$ of the ODE with constant control input $\boldsymbol{u}(t) = \boldsymbol{u}$, we have $(\boldsymbol{x}(0), \boldsymbol{u}, \boldsymbol{x}(T_s)) \in R$.*

Since we assume that the dynamics are time invariant, the starting time of the observation can be arbitrarily set to $t_0 = 0$. Time varying dynamics can be treated by lifting this assumption and specifying the value of the time t as part of the state $\boldsymbol{x}$ of the system.

We now consider the timed relational abstraction for a system with affine dynamics given by $\frac{d\boldsymbol{x}}{dt} = A\boldsymbol{x} + B\boldsymbol{u} + \boldsymbol{b}$. We note that the solution of the ODE can be written as $\boldsymbol{x}(t) = e^{tA}\boldsymbol{x}(0) + \int_{s=0}^{t} e^{(t-s)A}(B\boldsymbol{u}(s) + \boldsymbol{b})\, ds$. If the matrix A is invertible and $\boldsymbol{u}(s) = \boldsymbol{u}$ for $s \in [0, T_s)$, we may write the resulting relation as

$$\boldsymbol{x}(T_s) = e^{T_s A}\boldsymbol{x}(0) + A^{-1}(e^{T_s A} - I)(B\boldsymbol{u} + \boldsymbol{b})\,.$$

For general A, we write the result as

$$\boldsymbol{x}(T_s) = e^{T_s A}\boldsymbol{x}(0) + P(A, T_s)(B\boldsymbol{u} + \boldsymbol{b}),\ \text{wherein } P(A,t) = \sum_{j=0}^{\infty} \frac{A^j t^{j+1}}{(j+1)!}\,.$$

In theory, given T_s and A, we may compute the matrices $e^{T_s A}$ and $P(A, T_s)$ to arbitrary precision. This yields an affine expression for $\boldsymbol{x}(T_s)$ in terms of $\boldsymbol{x}(0), \boldsymbol{u}$.

In practice, however, arbitrary precision computation of the exponential map is often impractical, unless the matrix A is known to be diagonalizable with restrictions on its eigenvalues, or nilpotent. Therefore, for a general matrix A, we resort to error-prone numerical computations of $e^{T_s A}$ and $P(A, T_s)$.

The loss of soundness can be alleviated by using sophisticated numerical approximation schemes [22]. In particular, we can estimate both matrices using interval arithmetic calculations as $e^{T_s A} \in [\underline{E_s}, \overline{E_s}]$ and $P(A, T_s) \in [\underline{P_s}, \overline{P_s}]$ by taking into account the arithmetic and truncation errors of the resulting power series expansions [5, 26, 15]. Therefore, the resulting relationalization obtained is *interval linear* of the form $\boldsymbol{x}' \in [\underline{E_s}, \overline{E_s}]\boldsymbol{x} - [\underline{P_s}, \overline{P_s}](B\boldsymbol{u} + \boldsymbol{b})$ which stands for the logical formula

$$R(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') :\ (\exists\, E \in [\underline{E_s}, \overline{E_s}],\ P \in [\underline{P_s}, \overline{P_s}])\ \boldsymbol{x}' = E\boldsymbol{x} - P(B\boldsymbol{u} + \boldsymbol{b})\,.$$

As such, the relation above cannot be expressed in linear arithmetic. We will expand upon the treatment of interval linear relations later in this section.

Example 2. Going back to the system in Ex. 1, we find relational abstractions for mode n_0 when the system does not take an autonomous transition within the sampling period of 0.2 time units. Using a numerically computed matrix exponential, we obtain the relation $\boldsymbol{x}' = \begin{pmatrix} 0.7669 & 0.214 \\ 0.232 & 1.07 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} 0.1635 \\ -0.079 \end{pmatrix}$. On the other hand, using the interval arithmetic based method described by Goldsztejn [15], we obtain the relation

$$\boldsymbol{x}' \in \begin{pmatrix} [0.7669282852020186, 0.7669282852020187] & [0.2139643726426075, 0.2139643726426076] \\ [0.2317947337083272, 0.2317947337083273] & [1.0700444963848672, 1.0700444963848673] \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} [0.1635149326785402, 0.1635149326785403] \\ [-0.0789845829507958, -0.0789845829507957] \end{pmatrix}.$$

While pathological cases for matrix exponential computation are known (Cf. Goldsztejn [15]), the rather tight interval bounds for the exponential seem to be quite common in our benchmarks, and therefore, the use of numerically computed matrix exponentials may be quite satisfactory for many applications, wherein the dynamics are obtained as an approximation of the physical reality in the first place.

Applying the same for mode n_2, we obtain the relation $\boldsymbol{x}' = \begin{pmatrix} 1.5245 & 0.2181 \\ 0.0182 & 0.4885 \end{pmatrix} \boldsymbol{x} + \begin{pmatrix} -0.1626 \\ -0.0867 \end{pmatrix}$. Once again, an interval computation yields intervals of width 10^{-16} or less centered around the numerically computed value.

3.1 Dealing with Autonomous Transitions

Thus far, we have described a simple relationalization scheme under the assumption that no autonomous transitions were taken by the plant during a sampling time period. We will now describe the treatment of autonomous transitions that can be taken by the plant between two successive samplings. In general, there is no *a priori* bound on the number of such transitions that a plant can take in any given period $(nT_s, (n+1)T_s)$. Even if the plant is assumed to be non-Zeno, any relationalization has to capture the effects of the plant executing a finite but unbounded number of transitions. We remedy this situation by making two assumptions regarding the plant: (a) There is a minimum dwell time $T_D > 0$ for each mode q of the plant. In other words, whenever a run of the plant enters some mode q, it remains there for at least T_D time units before an autonomous transition is enabled. (b) No autonomous transitions can be taken precisely at the time instant $t = jT_s$ for $j \in \mathbb{Z}$.

The first assumption provides a bound $N = \left\lceil \frac{T_s}{T_D} \right\rceil$ on the maximum number of autonomous transitions taken inside a sampling interval. For this paper, we will assume that $N = 1$ to simplify the presentation, i.e., the controller is assumed to sample the plant fast enough to restrict the number of autonomous transitions in any sample period to at most 1. The second assumption allows us to use the standard *interleaving semantics* when the relationalization of the plant and the system are composed. This assumption fails if the execution time of the controller is not negligible compared to the time scale of the plant dynamics, as is sometimes the case. However, if bounds are known on the execution times, we may compute relationalizations of the plant for two time steps, one for the controller step and the other for the sampling period. Likewise, the basic ideas presented here extend to more sophisticated task execution schedules for control tasks.

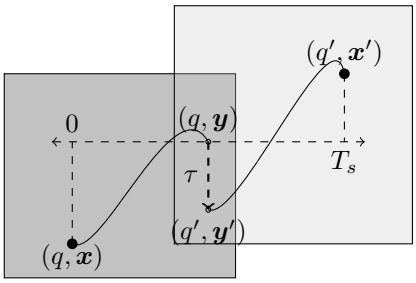

Fig. 3. Schematic for relational abstraction of a single autonomous transition.

Let us assume that a single autonomous transition $\tau : \langle q_1, q_2, \rho \rangle$ is taken during the time $t \in (0, T_s)$. Our goal is to derive a relation $R_\tau((q_1, \boldsymbol{x}), \boldsymbol{u}, (q_2, \boldsymbol{x}'))$ characterizing all possible pairs of states $(q_1, \boldsymbol{x})$ and $(q_2, \boldsymbol{x}')$ so that the plant may evolve from continuous state $\boldsymbol{x}$ in mode q_1 at time $t = 0$ to the state $(q_2, \boldsymbol{x}')$ at time $t = T_s$ with the transition τ taken at some time instant $0 < t < T_s$. The resulting R_τ for each $\tau \in \mathcal{T}_{aut}$ will form a disjunct of the overall relation R_P for the plant.

Figure 3 summarizes the situation diagrammatically. We let $\boldsymbol{y}$ be the valuation to continuous variables just prior to τ being taken and $\boldsymbol{y}'$ be the valuation just after τ is taken. Let t be the time instant at which τ is taken. Let the dynamics in mode q_i for $i = 1, 2$ be given by $\frac{d\boldsymbol{x}}{dt} = A_i\boldsymbol{x}+B_i\boldsymbol{u}+\boldsymbol{b}_i$. Therefore, $\boldsymbol{x} = e^{-tA_1}\boldsymbol{y}-P(A_1,-t)(B_1\boldsymbol{u}+\boldsymbol{b}_1) \wedge \boldsymbol{x}' = e^{(T_s-t)A_2}\boldsymbol{y}' - P(A_2, T_s - t)(B_2\boldsymbol{u} + \boldsymbol{b}_2)$. The overall relation is given by

$$R_\tau(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') : \ (\exists\, t, \boldsymbol{y}, \boldsymbol{y}') \begin{pmatrix} \boldsymbol{x} = e^{-tA_1}\boldsymbol{y} - P(A_1, -t)(B_1\boldsymbol{u} + \boldsymbol{b}_1) \\ \boldsymbol{x}' = e^{(T_s-t)A_2}\boldsymbol{y}' - P(A_2, T_s - t)(B_2\boldsymbol{u} + \boldsymbol{b}_2) \\ 0 < t < T_s \ \wedge \ \rho_\tau(\boldsymbol{y}, \boldsymbol{y}') \end{pmatrix}. \quad (1)$$

Note that we have chosen to encode $\boldsymbol{x} = e^{-tA_1}\boldsymbol{y}$ instead of encoding the dynamics in the forward direction $\boldsymbol{y} = e^{tA_1}\boldsymbol{x}$. This seemingly arbitrary choice will be seen to make the subsequent quantifier elimination problem easier.

Eliminating Quantifiers: The main problem with the relation R_τ derived in Eqn. (1) is that the matrices e^{tA_i} and $P(A_i, t)$ are, in general, *transcendental functions* of time. It is computationally intractable to manipulate these relations inside decision procedures. To further complicate matters, the variable t is existentially quantified. Removing this quantifier poses yet another challenge. However, our goal will be to derive an over-approximation of R_τ expressible in linear arithmetic.

To this end, the main challenge is to construct a good quality and linear over-approximation $R^a_\tau(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}')$ of the relation R_τ. We address this challenge using interval arithmetic techniques.

Interval Over-approximation We subdivide the interval $[0, T_s]$ into $M > 0$ subintervals each of width $\delta = \frac{T_s}{M}$. Next, we consider each subinterval of the form $t \in [i\delta, (i + 1)\delta)$ and use interval arithmetic evaluation for the functions e^{tA_i} and $P(A_i, t)$ to obtain a conservative approximation valid for the subinterval. In effect, we over-approximate R_τ as a disjunction

$$R^I_\tau : \bigvee_{0 \le i < M} (\exists \boldsymbol{y}, \boldsymbol{y}') \begin{pmatrix} \boldsymbol{x} \in [\underline{E_{i,1}}, \overline{E_{i,1}}]\boldsymbol{y} - [\underline{P_{i,1}}, \overline{P_{i,1}}](B_1\boldsymbol{u} + \boldsymbol{b}_1) \wedge \\ \boldsymbol{x}' \in [\underline{E_{i,2}}, \overline{E_{i,2}}]\boldsymbol{y}' - [\underline{P_{i,2}}, \overline{P_{i,2}}](B_2\boldsymbol{u} + \boldsymbol{b}_2) \wedge \\ \rho_\tau(\boldsymbol{y}, \boldsymbol{y}') \end{pmatrix}, \quad (2)$$

wherein $[\underline{E_{i,1}}, \overline{E_{i,1}}]$ is a safe interval enclosure of $e^{(-(i+1)\delta, -i\delta]A_1}$ while $[\underline{E_{i,2}}, \overline{E_{i,2}}]$ is an enclosure of $e^{(T_s-[i\delta,(i+1)\delta])A_2}$. Likewise, $[\underline{P_{i,1}}, \overline{P_{i,1}}]$ and $[\underline{P_{i,2}}, \overline{P_{i,2}}]$ are safe enclosures of $P(A_1, [-(i + 1)\delta, -i\delta])$ and $P(A_2, (\overline{T_s} - [i\delta, (i + 1)\overline{\delta}]))$, respectively.

The resulting over-approximation is a disjunction of M interval linear relations. In effect, the transcendental relation R_τ in Eq. (1) is over-approximated by an algebraic (bilinear) relation R^I_τ. The over-approximation error be made as small as necessary by increasing the number of subdivisions M, and by using a more expensive procedure for deriving a better approximation of the exponentials by intervals. The problem of evaluating safe interval enclosures to the matrices $e^{[t_1,t_2]A}$ and $P(A, [t_1, t_2])$ uses the idea of scaling and squaring with Horner's rule for evaluating the truncated power series, precisely as described by Goldsztejn [15]. A convenient trick used in our implementation folds the computation of $e^{A,[t_1,t_2]}$ and $P(A, [t_1, t_2])(Bu + \boldsymbol{b})$ into a single matrix exponential computation for a block matrix of the form $\begin{pmatrix} A & B & b \\ 0 & 0 & 0 \end{pmatrix}$.

Example 3. Consider the hybrid automaton described in Ex. 1. We wish to consider the relational abstraction when τ_1 is taken sometime during the sampling period of 0.2 seconds. To this end, we will choose $M = 2$ and consider two possible intervals for the switching time t when the transition τ_1 is taken $J_1 : [0, 0.1]$ and $J_2 : [0.1, 0.2]$. Considering interval J_1, we obtain the following relation (intervals are rounded to 2 significant digits for presentation):

$$R_{\tau,J_1} : (\exists\, \boldsymbol{y}) \begin{bmatrix} \boldsymbol{x} \in \begin{pmatrix} [0.99, 1.17] & [-0.13, 0.01] \\ [-0.14, 0.0] & [0.98, 1.01] \end{pmatrix} \boldsymbol{y} + \begin{pmatrix} [-0.11, 0] \\ [-0.01, 0.05] \end{pmatrix} \wedge \\ \boldsymbol{x}' \in \begin{pmatrix} [1.23, 1.53] & [0.09, 0.25] \\ [0.0, 0.02] & [0.48, 0.7] \end{pmatrix} \boldsymbol{y} + \begin{pmatrix} [-0.16, -0.07] \\ [-0.1, -0.04] \end{pmatrix} \wedge\ \boldsymbol{y} \in G_{\tau_1} \end{bmatrix}.$$

Templatization The next step is to use a *templatization* technique to effectively eliminate the quantifiers $\boldsymbol{y}, \boldsymbol{y}'$ from the relation R^I_τ in Equation (2) while, at the same time, over-approximating the result by means of a linear arithmetic over-approximation. Recall that each disjunct in Equation 2 is an *interval linear assertion* of the form

$$R^I_j : (\exists \boldsymbol{y}, \boldsymbol{y}') \begin{pmatrix} \boldsymbol{x} \in [\underline{E_{j,1}}, \overline{E_{j,1}}]\boldsymbol{y} - [\underline{P_{j,1}}, \overline{P_{j,1}}](B_1\boldsymbol{u} + \boldsymbol{b}_1) \wedge \\ \boldsymbol{x}' \in [\underline{E_{j,2}}, \overline{E_{j,2}}]\boldsymbol{y}' - [\underline{P_{j,2}}, \overline{P_{j,2}}](B_2\boldsymbol{u} + \boldsymbol{b}_2) \ \wedge\ \rho_\tau(\boldsymbol{y}, \boldsymbol{y}') \end{pmatrix},$$

An interval linear constraint of the form $\sum_{j=1}^{n} I_j x_j + I_0 \leq 0$ is a place holder for a *bi-linear* constraint $\sum_{j=1}^{n} w_j x_j + w_0 \leq 0$, wherein, $w_0, \ldots, w_n$ are freshly introduced variables and each w_j is constrained by requiring that $w_j \in I_j$.

In order to eliminate $\boldsymbol{y}, \boldsymbol{y}'$ from this relation, a technique for eliminating quantifiers for real arithmetic such as Cylindrical Algebraic Decomposition (CAD) [7], or a more efficient version for quadratic polynomials is called for [38, 12, 35]. However, the downside of using such complex techniques include (a) it is well known that QE over non-linear constraints is a hard problem with limited scalability, and (b) the result after elimination will, in general, be a set of polynomial inequalities (semi-algebraic constraint). Therefore, the resulting relationalization may not be easy to reason with for existing tools.

We present a more efficient alternative that side steps the elimination altogether, relying instead on the use of templates and optimization:

1. We choose a set of template expressions $e_k(\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{u})$ involving the variables $\boldsymbol{x},\ \boldsymbol{u}$ and $\boldsymbol{x}'$. We discuss a natural choice for these templates subsequently.
2. For each e_j, we carry out the optimization: $\mathsf{min}\ e_k$ s.t. $R^I_j(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{y}', \boldsymbol{x}')$. If the problem is feasible and bounded, the a_k allows us to conclude that

$$R^I_j(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}, \boldsymbol{y}', \boldsymbol{x}') \ \Rightarrow\ e_k(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') \geq a_k\,.$$

As a result by choosing some $K > 0$ templates $e_1, \ldots, e_K$, we obtain an assertion

$$e_1(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') \ \geq a_1\ \wedge\ e_2 \geq a_2\ \wedge\ \cdots\ \wedge\ e_K(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{x}') \geq a_K\,.$$

This assertion serves as an over-approximation to R^I_j with the quantified variables $\boldsymbol{y}, \boldsymbol{y}'$ eliminated through optimization. We now provide a natural scheme for the choice of

templates, and then discuss how the optimization problem for each template expression can be solved.

The overall relationalization of the plant R_P is the disjunction of the relations R_q, for each mode, and R^I_τ, for each autonomous transition τ.

Theorem 1. *For any pairs of states $\sigma : (q, \boldsymbol{x})$ and $\sigma' : (q', \boldsymbol{x}')$ such that $\boldsymbol{x}'$ is reachable from $\boldsymbol{x}$ in T_s seconds for constant control input $\boldsymbol{u}$, the computed relational abstraction R_P satisfies $R_P(\sigma, \sigma', \boldsymbol{u})$.*

Choosing Template Expressions A natural choice for template expression presents itself in our setup by considering the midpoints of the intervals used in the matrix exponential computations. We note that $\boldsymbol{y}$ is the state obtained starting from $\boldsymbol{x}$ and evolving in mode 1 for time $[i\delta, (i+1)\delta)$. Likewise, $\boldsymbol{x}'$ is obtained by evolving according to the state $\boldsymbol{y}'$ for time $[T_s - (i+1)\delta, T_s - i\delta)$. Finally, $\boldsymbol{y}' = U(\boldsymbol{y})$, wherein U is the affine update map for transition τ. In practice, δ is chosen to be small enough to yield tight enclosures to e^{tA_i} and $P(A_i, t)$ matrices. Therefore, a natural choice of template expression is obtained by considering the midpoints of the time intervals involved. Specifically, we consider the affine expressions defined by

$$\boldsymbol{x}' - e^{(T_s - t_m)A_2} U(e^{t_m A_1}\boldsymbol{x}), \text{ where } t_m = (i + \frac{1}{2})\delta\,.$$

Example 4. In Ex. 3, we showed the interval linear relation obtained by considering switching times in the interval $t \in [0.0, 0.1]$. The midpoint of this interval is $t_m = 0.05$. Therefore, we consider the mode n_0 taken for time 0.05 units followed by 0.15 units of mode 1 for generating a suitable template. These template expressions are given by $e_1 : x' - 1.31x - 0.25y$ and $e_2 : y' - 0.05x - 0.6y$. We seek to bound these expressions to obtain a linear arithmetic over-approximation.

Encoding Optimization Next, we turn our attention to setting up the optimization problem for a given template expression $\boldsymbol{c}\boldsymbol{x} + \boldsymbol{d}\boldsymbol{x}'$. The intermediate states $\boldsymbol{y}, \boldsymbol{y}'$ are related by interval linear expressions of the form

$$\boldsymbol{x}' \in [\underline{E_2}, \overline{E_2}]\boldsymbol{y}' + [\underline{P_2}, \overline{P_2}],\ \boldsymbol{x} \in [\underline{E_1}, \overline{E_1}]\boldsymbol{y} + [\underline{P_1}, \overline{P_1}]\,.$$

To set up the optimization problem, we substitute these expressions for $\boldsymbol{x}, \boldsymbol{x}'$ in the template to obtain $\boldsymbol{c}([\underline{E_1}, \overline{E_1}]\boldsymbol{y} + [\underline{P_1}, \overline{P_1}]) + \boldsymbol{d}([\underline{E_2}, \overline{E_2}]\boldsymbol{y}' + [\underline{P_2}, \overline{P_2}])$. This is, in fact, an interval linear expression involving $\boldsymbol{y}, \boldsymbol{y}'$. The overall optimization problem reduces to: $\min\ [\underline{\boldsymbol{c}}, \overline{\boldsymbol{c}}]\boldsymbol{y} + [\underline{\boldsymbol{d}}, \overline{\boldsymbol{d}}]\boldsymbol{y}' + [\underline{c_0}, \overline{c_0}]$ s.t. $\rho_\tau(\boldsymbol{y}, \boldsymbol{y}')$. Here $[\underline{\boldsymbol{c}}, \overline{\boldsymbol{c}}] = \boldsymbol{c}[\underline{E_1}, \overline{E_1}]$, $[\underline{\boldsymbol{d}}, \overline{\boldsymbol{d}}] = \boldsymbol{d}[\underline{E_2}, \overline{E_2}]$ and $[\underline{c_0}, \overline{c_0}] = \boldsymbol{c}[\underline{P_1}, \overline{P_1}] + \boldsymbol{d}[\underline{P_2}, \overline{P_2}]$. The problem has an interval linear objective and linear constraints. We now show that the constraints can be encoded into a disjunctive linear program.

Theorem 2. *The optimization of an interval linear objective w.r.t linear constraints*

$$\min\ [\underline{\boldsymbol{c}}, \overline{\boldsymbol{c}}] \times \boldsymbol{z} \text{ s.t. } A\boldsymbol{z} \leq \boldsymbol{b}\,,$$

can be equivalently expressed as a linear program with disjunctive constraints:

$$\min\ \ \underline{\boldsymbol{c}}\boldsymbol{z}^+ - \overline{\boldsymbol{c}}\boldsymbol{z}^- \text{ s.t. } A\boldsymbol{z}^+ - A\boldsymbol{z}^- \leq \boldsymbol{b},\ \boldsymbol{z}^+, \boldsymbol{z}^- \geq 0,\ z_i^+ = 0 \ \vee\ z_i^- = 0$$

where $\boldsymbol{z} = \boldsymbol{z}^+ - \boldsymbol{z}^-$.

Proof. We may decompose any vector $\boldsymbol{z}$ as $\boldsymbol{z} = \boldsymbol{z}^+ - \boldsymbol{z}^-$, where $\boldsymbol{z}^+, \boldsymbol{z}^- \geq 0$, and enforce $z_i^+ z_i^- = 0$. Next, consider the objective $[\underline{\boldsymbol{c}}, \overline{\boldsymbol{c}}] \times (\boldsymbol{z}^+ - \boldsymbol{z}^-)$. Since correspondent entries in $\boldsymbol{z}^+, \boldsymbol{z}^-$ cannot be positive at the same time, we may write the objective as a linear expression $\underline{\boldsymbol{c}}\boldsymbol{z}^+ - \overline{\boldsymbol{c}}\boldsymbol{z}^-$. Finally, the complementarity condition $z_i^+ z_i^- = 0$ is rewritten as $z_i^+ = 0 \ \vee \ z_i^- = 0$.

A simple approach to solve the optimization problem for disjunctive constraints is to use a linear arithmetic SMT solver to repeatedly obtain feasible solutions $\boldsymbol{z}^+, \boldsymbol{z}^-$. For a given feasible solution output by the SMT solver, we fix a minimal set of the values for $\boldsymbol{z}^+, \boldsymbol{z}^-$ to zero to enforce the complementarity constraints $z_i^+ z_i^- = 0$, leaving the remaining variables as unknowns. An LP solver is then used to compute an optimal value f^* for the objective function f, based on the remaining constraints. This yields a potential optimum. Next, we add a *blocking constraint* $f > f^*$ to the SMT solver and search for a different solution. The process is carried out until the SMT solver returns UNSAT. At this point, we output the last optimal solution as the final value.

Example 5. Continuing with the examples worked out in Ex. 3, we perform the optimization of the template expressions chosen in Ex. 4 to obtain the relational abstraction:

$$R_{\tau_1,J_1} : \ -1.0 \leq 1.31x + 0.25y - x' \leq 1.24 \ \wedge \ -0.32 \leq 0.05x + 0.6y - y' \leq 0.51 \, .$$

Likewise, considering the time interval $J_2 : [0.1, 0.2]$ for the switching time, we obtain the abstraction:

$$R_{\tau_1,J_2} : \ -1 \leq 0.94x + 0.25y - x' \leq 0.88 \wedge -0.54 <= 0.17x + 0.9y - y' <= 0.7708 \, .$$

The overall timed relational abstraction for the sampling period where τ_1 can be taken sometime in between is $R_{\tau_1} : R_{\tau_1,J_1} \ \vee \ R_{\tau_1,J_2}$.

4 Experimental Evaluation

We first briefly describe our implementation of the relational abstractor using the techniques presented here.

Implementation: The relational abstractor takes in a plant description including the sample time T_s, and outputs the relation as a SAL transition system [32]. The relationalization is performed for the continuous dynamics in each mode by computing a matrix exponential. A numerical approximation of the matrix exponential function is obtained using Pade's approximation [22]. We have also implemented a procedure that provides a sound interval enclosure of the exponential function over interval matrices using the ideas described by Goldsztejn [15]. However, this procedure is used solely for dealing with autonomous transitions.

Autonomous transition between modes are handled using the algorithms presented so far. We implicitly assume minimum dwell time greater than or equal to the sampling time for the controller. The optimization problems encountered for autonomous transitions are solved using the SMT solver Z3 [11]. SAL provides a k-induction and BMC engine using the solver Yices [13]. This was used for analyzing the resulting composed transition system for our evaluations.

Model	Description	# Var	# Mode	# Trs	Prop.	Description
InvPen	Inverted Pendulum Control	5	1	1	$\theta_b(0.05)$	Angle $\theta \in [-0.05, 0.05]$
SNCS	Network Control System [39]	2	1	2	P_1	$(x, y) \in [-100, 100]^2$
					P_2	$(x, y) \in [-10^4, 10^4]^2$
ACC	Adaptive Cruise Control [17]	4	1	2	SAFE	No collision between cars.
ACC-T	ACC + transmission [17]	3	20	24	SAFE	No collision between cars.
Heat-x	Room heater [14]	9	8	20	LB	Lower bounds on temp. Cf. [14]
Nav-y	NAV benchmarks [14]	4	[7,16]	[9,16]	RA	Cell A is unreachable
					RB	Cell B is unreachable
Toy	Example 1	2	2	5	$bnd(k)$	$n_1 \Rightarrow \|\boldsymbol{x}\| \leq k$
Ring(n,m)	Cf. Section 4	n	m+1	m+1	$bnd(k)$	$n_4 \Rightarrow \|\boldsymbol{x}\| \leq k$

Table 1. The benchmarks used in our experiments at a glance.

Benchmarks: Table 1 shows the benchmarks used in our evaluation along with their sources. The benchmarks vary in dimensionality and number of transitions. Note that many benchmarks do not contain autonomous transitions. For each benchmark, we performed the relational abstraction for different sampling times T_s, and used SAL to analyze safety properties.

The NAV benchmarks, due to Ivancic and Fehnker [14], model a particle traveling through many $2D$ cells that each have a different dynamics. We consider two versions of this benchmark (a) the transitions in the benchmark are all interpreted as controlled, commanded by a controller, or (b) transitions are autonomous in nature. Starting with all controlled transitions, we introduce uncontrolled transitions incrementally into these benchmarks.

Ring Benchmarks: We created a set of sampled data control systems with autonomous transitions. We consider a plant with $k+1$ modes, wherein modes $m_1, \ldots, m_k$ are governed by stable dynamics, while mode m_{k+1} is an unstable mode. The controller seeks to stabilize this mode by periodically sensing the plant's state and applies a control that reverts it back to mode m_1.

The benchmark instance `Ring(n,k)` consists of n state variables and $k+1$ modes. The autonomous transitions are added from mode i to mode $i+1$ for $i \leq k$, while the controlled transition leads from mode $k+1$ to mode 1. The dynamics in each mode is of the form $\frac{d\boldsymbol{x}}{dt} = A_i(\boldsymbol{x} - \boldsymbol{b}_i)$, wherein, for the stable modes A_i is a Hurwitz matrix and $\boldsymbol{b}_i$ is a designated equilibrium for m_i. For the unstable mode, we ensure that A_i has a positive eigenvalue. The switch from m_i to m_{i+1} takes place inside a box $[\boldsymbol{b}_i - \boldsymbol{\epsilon}, \boldsymbol{b}_i + \boldsymbol{\epsilon}]$. The controller periodically senses the plant and whenever $|\boldsymbol{x}| > \boldsymbol{c}$ for some fixed $\boldsymbol{c}$, it brings the dynamics back into the box $|\boldsymbol{x}| < \boldsymbol{c}$ while transitioning to mode m_1. We wish to check whether all trajectories lie inside a box $|\boldsymbol{x}| \leq \boldsymbol{c} + \boldsymbol{d}$, for varying tolerances $\boldsymbol{d}$.

Results: Table 2 shows the experimental results on benchmarks that do not have autonomous transitions. Our experiments are attempted using numerous values of sample times for each property until either a proof is obtained for the controller or the SAL tool fails due to a timeout. In the absence of autonomous transitions, the relational-

Model	Prop	T_s	Result	Depth	Time
InvPen	$\theta_b(0.05)$	0.1	CE	1	0.1
		0.05	P	12	0.9
SNCS	P_1	1.7	P	2	0.1
	P_2	1.8	CE	93	5.6
ACC	SAFE	0.1	P	7	0.1
	gap(100)	0.1	P	4	0.1
ACC-T	SAFE	1	P	14	2.2
Nav 1-7	1	RB	P	<11	<5
	1	RA	CE	<13	<2
Nav 8	1	RB	CE	7	0.27
	0.2	RB	F	25	$> 1h$
Nav 9	1	RB	P	19	213.05
	1	RA	CE	9	0.37
Nav 10	1	RB	CE	19	28.37
	0.5	RB	F	25	$> 1h$

Model	T_s	Prop	result	depth	time
Heat1	LB	1	CE	4	0.1
		.2	CE	8	0.1
		0.1	P	37	1967
Heat2	LB	1	CE	4	0.1
		0.2	P	17	160
Heat3	LB	1	CE	2	0.1
		0.2	CE	17	27
		0.1	F	30	$> 1h$
Heat4	LB	1	CE	2	0.1
		0.1	CE	10	1.22
		0.02	F	25	$> 1h$

Table 2. Results on benchmarks without autonomous transitions. All timings were measured in seconds on a laptop running Intel Core i7-2820Q 2.30GHz processor (x86_64 arch) with 8GB RAM running Ubuntu 11.04 Linux 2.6.38-13. Legend: **CE** indicates true counter-example, **P** indicates proofs, **F** indicates failure due to timeout.

ization time for all these benchmarks was well under 1 second. We also note that the counterexamples generated by SAL can be concretized, since the timed abstractions involving matrix exponentials are seen to be quite precise.

Table 3 shows the results for systems with controlled and autonomous transitions. These include the system from Ex. 1, the `Ring(n,k)` systems for varying n and the NAV benchmark instances as we increase the number of autonomous transitions. We observe that making all the transitions autonomous leads to a counterexample. This counterexample may potentially be an artifact of the precision loss due to relationalization of autonomous transitions. Future work will consider the refinement of these counterexamples by subdividing the transition switching time intervals further based on spurious counterexamples. We note that the time for relationalization remains a small fraction of the time needed to check the system. The relationalization scheme can be improved further if SMT solvers such as Z3 can be modified to support the optimization of objective functions.

Comparison With SpaceEx: We now compare the results obtained for our approach with the SpaceEx tool over the same set of benchmarks [**?**]. While performing the comparison with SpaceEx, we reiterate two key points of difference: (a) SpaceEx handles general hybrid systems with support for synchronous time-triggered semantics as well as the standard event-triggered semantics given by guards and resets. Our technique is specialized to sampled data control systems. (b) SpaceEx attempts to characterize the reachable sets for all time instances, whereas our approach focuses on proving properties at the periodic sampling times.

Typically, running the benchmarks in SpaceEx required choosing from a range of parameters including template domains, underlying implementation, flowpipe tol-

Model	Prop	result	depth	T_{sal}	T_{rel}
Toy	$bnd(8)$	P	2	0.1	$< .1$
	$bnd(6)$	P	2	0.1	
	$bnd(5.5)$	P	2	0.3	
	$bnd(5)$	CE	70	204	
Ring(3,4)	$bnd(20)$	P	10	5.2	0.8
	$bnd(15)$	P	30	451	
Ring(5,4)	$bnd(25)$	P	10	34.7	2.8
	$bnd(20)$	P	10	56.2	
	$bnd(15)$	F	20	$>$ 1h	
Ring(7,4)	$bnd(25)$	P	10	157	11.9
	$bnd(20)$	P	10	357	
	$bnd(15)$	F	20	$>$1h	
Ring(9,4)	$bnd(25)$	P	10	515	19.1
	$bnd(20)$	P	10	2929	
Ring(11,4)	$bnd(25)$	F	10	$>$ 1h	150

Model	Prop	T_s	# Aut.	result	depth	time
Nav1	RB	0.2	6	P	9	199
			14	P	9	72
			21	P	9	96
			24	F	9	169
			All	CE	9	161
Nav2	RB	0.2	20	P	9	92
			21	F	9	160
			All	CE	7	151
Nav3	RB	0.2	22	P	9	83
			All	CE	6	13
Nav4	RB	0.2	9	P	18	1305
			20	F	18	$>$1h
			All	CE	6	7

Table 3. Results on systems with autonomous transitions. For the NAV benchmarks, autonomous transitions between cells were incrementally enabled over the controlled transitions until all transitions were autonomous. T_{sal} refers to running time for SAL and T_{rel} the running time for the relationalization.

erances, error tolerances, local and global time horizons and limits on the number of iteration. We ran SpaceEx for each benchmark using multiple option sets, choosing the option that provided the "best answer" with as few warnings as possible. A detailed table summarizing our experiences is available upon request.

Table **??** presents a summary of the results obtained by running SpaceEx on our benchmarks. We note that in many cases, SpaceEx did not reach a fixed point. Therefore, whenever a property proof was obtained, we report if the proof was obtained over a finite time horizon. Likewise, for cases where the property was not proved, we ran SpaceEx for the minimum number of iterations until a potential violation is observed.

The comparison between our approaches clearly showcases some of the relative merits and demerits of our approach vis-a-vis SpaceEx. There are many benchmarks wherein our approach is able to establish the property over an infinite time horizon using k-induction, whereas SpaceEx either proves the property over a finite time horizon or fails. On the other hand, the NAV benchmarks are an interesting case where SpaceEx's performance is at par or clearly superior to that of our approach.

For some of the Ring examples, we observed that using the bounds obtained by SpaceEx as inductive strengthenings enabled the k-induction technique to prove the property for a much smaller value of k, leading to improved running times. On the other hand, for the inverted pendulum example, the inductive strengthenings found by SpaceEx did not improve the performance of k-induction.

Our future work will focus on an integration of the approaches considered here in combination with tools such as SpaceEx to achieve infinite horizon safety property proofs. Another important area of future research will be to extend our approach to analyze non linear hybrid systems, which are much more challenging.

Model	Prop	T_s	Result	Time
InvPen	$\theta_b(0.05)$	0.05	F	6
SNCS	P_1	1.7	F	371
Heat1	LB	0.1	F	557
Ring(3,4)	$bnd(20)$	0.2	P (FT)	3475
Ring(5,4)	$bnd(25)$	0.2	P (FT)	2051
Ring(7,4)	$bnd(25)$	0.2	F	6323
Ring(9,4)	$bnd(25)$	0.2	F	709

Model	Prop	T_s	Result	Time
Nav 4(A)	RB	–	P (FT)	1501
Nav 6(A)	RB	–	F	1223
Nav 7(A)	RB	–	P (FT)	615
Nav 9(A)	RB	–	F	619
Nav 4(C)	RB	1	P (FT)	109
Nav 6(C)	RB	1	P (FT)	167
Nav 7(C)	RB	1	P	7
Nav 9(C)	RB	1	F	6

Table 4. Results of SpaceEx tool on the benchmark. Legend: **A:** All transitions were Autonomous, **C:** All transitions were controlled Autonomous, **Prop:** property, **F:** Found potentially spurious counter-example (our approach proves model+property), **P:** Proved, **FT:** proof valid over a finite time horizon, **Time:** Running time in seconds.

References

1. R. Alur, T. Dang, and F. Ivančić. Counter-example guided predicate abstraction of hybrid systems. In *TACAS*, volume 2619 of *LNCS*, pages 208–223. Springer, 2003.
2. E. Asarin, T. Dang, and A. Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43:451—476, 2007.
3. J. Berdine, A. Chawdhary, B. Cook, D. Distefano, and P. W. O'Hearn. Variance analyses from invariance analyses. In *POPL*, pages 211–224. ACM, 2007.
4. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *TACAS*, volume 1579 of *LNCS*, pages 193–207, 1999.
5. P. Bochev and S. Markov. A self-validating numerical method for the matrix exponential. *Computing*, 43(1):59–72.
6. E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003.
7. G. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In H.Brakhage, editor, *Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183. Springer, 1975.
8. P. Cousot and R. Cousot. Abstract Interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *ACM Principles of Programming Languages*, pages 238–252, 1977.
9. T. Dang, O. Maler, and R. Testylier. Accurate hybridization of nonlinear systems. In *HSCC '10*, pages 11–20. ACM, 2010.
10. T. Dang and D. Salinas. Image computation for polynomial dynamical systems using the bernstein expansion. In *CAV*, volume 5643 of *LNCS*, pages 219–232. Springer, 2009.
11. L. M. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.
12. A. Dolzmann and T. Sturm. REDLOG: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, June 1997.
13. B. Dutertre and L. de Moura. The YICES SMT solver. Cf. `http://yices.csl.sri.com/tool-paper.pdf`, last viewed Jan. 2009.
14. A. Fehnker and F. Ivančić. Benchmarks for hybrid systems verification. In *HSCC*, volume 2993 of *LNCS*, pages 326–341. Springer, 2004.

15. A. Goldsztejn. On the exponentiation of interval matrices, 2009. Preprint (Working Paper) # hal-00411330, version 1. Cf. http://hal.archives-ouvertes.fr/hal-00411330/fr/.
16. S. Gulwani, S. Jain, and E. Koskinen. Control-flow refinement and progress invariants for bound analysis. In *PLDI*, 2009.
17. S. Gulwani and A. Tiwari. Constraint-based approach for hybrid systems. In *CAV*, volume 5123 of *LNCS*, pages 190–203, 2008.
18. N. Halbwachs, Y.-E. Proy, and P. Roumanoff. Verification of real-time systems using linear relation analysis. *Formal Methods in System Design*, 11(2):157–185, 1997.
19. T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 43:540–554, 1998.
20. B. Jeannet, N. Halbwachs, and P. Raymond. Dynamic partitioning in analyses of numerical properties. In *SAS*, volume 1694 of *LNCS*, pages 39–50, 1999.
21. Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer, New York, 1995.
22. C. Moler and C. V. Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM review*, 45(1):161–208, 2003.
23. R. Moore, R. B. Kearfott, and M. Cloud. *Introduction to Interval Analysis*. SIAM, 2009.
24. V. Mysore, C. Piazza, and B. Mishra. Algorithmic algebraic model checking II: Decidability of semi-algebraic model checking and its applications to systems biology. In *ATVA*, volume 3707 of *LNCS*, pages 217–233. Springer, 2005.
25. M. Oishi, I. Mitchell, A. M. Bayen, and C. J. Tomlin. Invariance-preserving abstractions of hybrid systems: Application to user interface design. *IEEE Trans. on Control Systems Technology*, 16(2), Mar 2008.
26. E. P. Oppenheimer and A. N. Michel. Application of interval analysis techniques to linear systems. II. the interval matrix exponential function. *IEEE Trans. on Circuits and Systems*, 35(10):1230–1242, 1988.
27. A. Platzer and E. Clarke. Computing differential invariants of hybrid systems as fixedpoints. *Formal Methods in Systems Design*, 35(1):98–120, 2009.
28. A. Podelski and A. Rybalchenko. Transition invariants. In *LICS*, pages 32–41. IEEE Computer Society, 2004.
29. A. Podelski and S. Wagner. Model checking of hybrid systems: From reachability towards stability. In *HSCC*, volume 3927 of *LNCS*, pages 507–521. Springer, 2006.
30. S. Prajna and A. Jadbabaie. Safety verification using barrier certificates. In *HSCC*, volume 2993 of *LNCS*, pages 477–492. Springer, 2004.
31. S. Ratschan and Z. She. Safety verification of hybrid systems by constraint propagation based abstraction refinement. In *HSCC*, volume 3414 of *LNCS*, pages 573–589. Springer, 2005.
32. J. Rushby, P. Lincoln, S. Owre, N. Shankar, and A. Tiwari. Symbolic analysis laboratory (sal). Cf. http://www.csl.sri.com/projects/sal/.
33. S. Sankaranarayanan and A. Tiwari. Relational abstractions for continuous and hybrid systems. In *CAV*, volume 6806 of *LNCS*, pages 686–702. Springer, 2011.
34. M. Sheeran, S. Singh, and G. Stålmarck. Checking safety properties using induction and a sat-solver. In *FMCAD*, volume 1954 of *LNCS*, pages 108–125. Springer, 2000.
35. T. Sturm and A. Tiwari. Verification and synthesis using real quantifier elimination. In *ISSAC*, pages 329–336. ACM, 2011.
36. A. Tiwari. Abstractions for hybrid systems. *Formal Methods in Systems Design*, 32:57–83, 2008.
37. A. Tiwari and S. International. HybridSAL: A tool for abstracting HybridSAL specifications to SAL specifications, 2007. Cf. http://sal.csl.sri.com/hybridsal/.

38. V. Weispfenning. Quantifier elimination for real algebra—the quadratic case and beyond. In *Applied Algebra and Error-Correcting Codes (AAECC) 8*, pages 85–101, 1997.
39. W. Zhang, M. S. Branicky, and S. M. Phillips. Stability of networked control systems. *IEEE Control Systems Magazine*, 21:84–99, 2001.