Compositional Termination Analysis of Symbolic Forward Analysis

Witold Charatonik, Supratik Mukhopadhyay, and Andreas Podelski

Max-Planck-Institut für Informatik
Im Stadtwald, 66123 Saarbrücken, Germany
{witold|supratik|podelski}@mpi-sb.mpg.de

Abstract. Existing model checking tools for infinite state systems, such as UPPAAL, HYTECH and KRONOS, use symbolic forward analysis, a possibly nonterminating procedure. We give termination criteria that allow us to reason compositionally about systems defined with asynchronous parallel composition; we can prove the termination of symbolic forward analysis for a composed system from the syntactic conditions satisfied by the component systems.

Our results apply to nonlinear hybrid systems; in particular to rectangular hybrid systems, timed automata and o-minimal systems. In the case of integer-valued systems we give negative results: forward analysis is not well-suited for this class of infinite-state systems.

1 Introduction

Recently, there has been a lot of research effort directed to automatic verification of infinite state systems. Research on decidability issues (e.g., [ACJT96,ACHH93,Boi98,LPY99,LPY00,HKPV95,CJ98]) has resulted in many nontrivial algorithms for the verification of different subclasses of infinite state systems. These results do not, however, imply the termination of the semi-algorithms on which practical tools are based (for example, the decidability of the model checking problem for timed automata does not entail termination for the symbolic forward analysis used in UPPAAL or HYTECH or KRONOS).

This paper addresses the termination for such a procedure in a compositional setting; we give sufficient *compositional* conditions for the termination of the symbolic forward analysis for nonlinear hybrid systems. We can prove the termination of this analysis for a composed system from the syntactic conditions satisfied by the components, without computing the explicit representation of the composition (which is usually exponentially bigger than the components). The conditions roughly express that, in each loop, the variables are initialized before they are used. Our sufficient conditions apply to several interesting examples such as the railroad crossing example. As a corollary we obtain termination for the subclass of *o-minimal* hybrid systems (for which backward analysis is known to be terminating [LPY99,LPY00]).

Sufficient termination conditions for symbolic forward analysis seem interesting for three reasons. First, since they apply to concrete examples such as practical mutual exclusion protocols, they may shed a new light on the practical success of symbolic model checking for infinite-state systems (see e.g., [LPY95]). Second, for a concrete verification problem in a practical setting, the model to be checked can possibly be adapted to meet the sufficient termination conditions (e.g., by adding semantically redundant initializations of variables or hiding not used variables); we give such examples in the paper. This can be interesting to obtain a theoretical guarantee for a problem where practical termination has not yet been obtained. Such a guarentee is pragmatically useful in a limited sense, as follows. If one has waited for two hours for an experiment that is known to terminate theoretically, it may be worth to wait two hours more.

Third, our results suggest a potential optimization of the symbolic forward analysis procedure. Namely, the termination guarantees given in this paper continue to hold even when the fixpoint test is made more efficient by weakening it to *local entailment* (explained below; e.g., for linear arithmetic constraints over reals, the complexity of fixpoint test reduces from co-NP hard to polynomial; such a fixpoint test is used in the model checker UPPAAL [BLL$^+$96] and in the model checker described in [DP99]).

2 Preliminaries

2.1 Infinite State Systems

We use guarded-command programs to specify (possibly infinite-state) transition systems. A guarded-command program consists of a set $\mathcal{E}$ of guarded commands e (called edges) of the form

$$e \;\equiv\; \ell :\; \gamma_e(\boldsymbol{x}) \;[\!]\, \alpha_e(\boldsymbol{x}, \boldsymbol{x'});\; \mathsf{goto}\; \ell'$$

where ℓ and ℓ' are labels ranging over a finite set of program locations, $\boldsymbol{x} = \langle x_1, \dots, x_n \rangle$ is the tuple of program variables (ranging over a possibly infinite data domain); $\gamma_e(\boldsymbol{x})$ is a formula (the guard) whose free variables are among $\boldsymbol{x}$; $\alpha_e(\boldsymbol{x}, \boldsymbol{x'})$ is a formula (the action of e) whose free variables are among $\boldsymbol{x}, \boldsymbol{x'}$. Intuitively, the primed version of a variable stands for its value in the successor state after taking a transition through a guarded command. We allow more than one command labeled with the same location ℓ, which corresponds to a nondeterministic choice in the language.

We translate a guarded command e to the logical formula ψ_e simply by by replacing the guard $[\!]$ with conjunction and introducing a new variable L for locations.

$$\psi_e \equiv L = \ell \wedge \gamma_e(\boldsymbol{x}) \wedge L' = \ell' \wedge \alpha_e(\boldsymbol{x}, \boldsymbol{x'})$$

A state of the system is a pair $\langle \ell, \boldsymbol{v} \rangle$ consisting of the values for the location variable and for each program variable. The state $\langle \ell, \boldsymbol{v} \rangle$ can make a transition to the state $\langle \ell', \boldsymbol{v'} \rangle$ through the edge e provided that the values of ℓ for L, ℓ' for L', $\boldsymbol{v}$ for $\boldsymbol{x}$ and $\boldsymbol{v'}$ for $\boldsymbol{x'}$ define a solution for ψ_e. A run of the system is a sequence $\langle \ell^1, \boldsymbol{v^1} \rangle \longrightarrow \langle \ell^2, \boldsymbol{v^2} \rangle \longrightarrow \dots$ such that for each $i = 1, 2, \dots$ there exists an edge e such that the state $\langle \ell^i, \boldsymbol{v^i} \rangle$ can make a transition to the state $\langle \ell^{i+1}, \boldsymbol{v^{i+1}} \rangle$ through the edge e.

In this paper, we consider two basic classes of infinite state systems. In the first, we deal with the so-called hybrid systems in which the program variables range over the set of reals $\mathbb{R}$. Examples of such systems include the railroad crossing example and the Fischer's mutual exclusion protocol. In the second, the program variables range over the set of integers $\mathbb{Z}$, and the guard and the action formulas are arithmetic constraints. Examples of such systems include the bakery or ticket algorithms, the bounded buffer producer-consumer problem etc.

Systems with Integer-valued Variables. A system with integer-valued variables can be defined as a set of guarded commands as above where the variables $\boldsymbol{x}, \boldsymbol{x'}$ are interpreted over the set of integers $\mathbb{Z}$. We consider these systems in Section 5.

Hybrid Systems We write $OF(\mathbb{R})$ for the theory of the ordered field of reals; it is interpreted over the structure $\langle \mathbb{R}, <, +, \cdot, 0, 1 \rangle$.

A (possibly non-linear) hybrid system can be defined as a set of guarded commands as above where the guard $\gamma_e(\boldsymbol{x})$ is an $OF(\mathbb{R})$ formula, and the action $\alpha_e(\boldsymbol{x}, \boldsymbol{x'})$ is an $OF(\mathbb{R})$ formula given by

$$\alpha_e(\boldsymbol{x}, \boldsymbol{x'}) \;\equiv\; \exists z \geq 0 \; \exists \boldsymbol{x''} \; \delta_e(\boldsymbol{x}, \boldsymbol{x''}) \wedge \beta_{\ell'}(\boldsymbol{x''}, \boldsymbol{x'}, z).$$

Here, δ_e is an $OF(\mathbb{R})$ formula defining the "update" in e, and $\beta_{\ell'}$ is the $OF(\mathbb{R})$ formula defining the continuous evolution at the target location ℓ'. For example, in a timed system with two clocks x_1, x_2 where x_2 runs twice as fast as x_1, the action part of a command resetting the first clock would be $\exists z \geq 0 \; \exists x_1'' \exists x_2'' \; x_1'' = 0 \wedge x_2'' = x_2 \; \wedge \; x_1' = x_1'' + z \wedge x_2' = x_2'' + 2z$.

A transition according to a guarded command e represents an instantaneous 'jump' followed by a continuous evolution over time at the target location ℓ'. Namely, a state $\langle \ell, \boldsymbol{v} \rangle$ can make a transition through e to the state $\langle \ell', \boldsymbol{v'} \rangle$ if the values ℓ for the location variable L and $\boldsymbol{v}$ for the tuple of data variables $\boldsymbol{x}$ satisfy the guard $L = \ell \wedge \gamma_e(\boldsymbol{x})$ of e and there exists a $\boldsymbol{v''}$ such that $\boldsymbol{v}, \boldsymbol{v''}$ satisfies the update δ_e of e and there exists a real value d of the delay variable z such that $\boldsymbol{v'}$ is obtained from $\boldsymbol{v''}$ through continuous evolution over the delay d at the location ℓ'. A particular case of a transition in a hybrid system is the *time transition* (continuous evolution over time at a location) from the state $\langle \ell, \boldsymbol{v} \rangle$ to the state $\langle \ell, \boldsymbol{v'} \rangle$ where the update part $\delta(\boldsymbol{x}, \boldsymbol{x''})$ is simply the equality $\boldsymbol{x} = \boldsymbol{x''}$.

Rectangular hybrid systems A rectangular hybrid system is a hybrid system where the guards $\gamma_e(\boldsymbol{x})$ are conjunctions of constraints of the form $x_i \sim c_i$ where $\sim \in \{<, \leq, >, \geq\}$ and $c_i \in \mathbb{Z}$; the update part of the action formulas consists of the jump to a location with an initialization of some variables (the only allowed constraints are $L' = \ell', x'' = x$ and $x'' \sim c$ where $c \in \mathbb{Z}$) and the continuous evolution $\beta_{\ell'}(\boldsymbol{x''}, \boldsymbol{x'}, z)$ is of the form $\bigwedge_{i=1}^{n} x_i'' + a_i z \sim x_i' \sim x_i'' + b_i z$ possibly in conjunction with location invariants of the form $c_i \sim x_i' \sim d_i$ where $a_i, b_i, c_i, d_i \in \mathbb{Z} \cup \{-\infty, +\infty\}$.

We will often use a notation like $\dot{x} \in [a, b)$ as a shortcut for $x'' + az \leq x' < x'' + bz$. The continuous evolution of the timed system from the example above can be then described by $\dot{x_1} = 1, \dot{x_2} = 2$.

Timed automata Timed automata are particular case of rectangular hybrid systems where the update part consists of the jump to a location with reset of some clocks (the only allowed constraints are of the form $L' = \ell'$, $x'' = x$ and $x'' = 0$) and the continuous evolution is the increment of the clocks according to the time passing ($\beta_{\ell'}(\boldsymbol{x''}, \boldsymbol{x'}, z) \equiv \boldsymbol{x'} = \boldsymbol{x''} + z$), possibly in conjunction with location invariants of the form $x' \sim c$ where c is an integer.

O-minimal hybrid systems The o-minimal hybrid systems were introduced in [LPY99,LPY00], where it is shown that the *backward* analysis for these systems terminates. Our results generalize this one in two ways: we prove the termination of not only backward, but also forward analysis; second, our systems are less restrictive by allowing parallel composition and continuous change of variables between different locations.

Below we rephrase the definition from [LPY99,LPY00]. In o-minimal hybrid systems, the action formula $\alpha_e(\boldsymbol{x}, \boldsymbol{x'})$ of e is of the form $\alpha_e(\boldsymbol{x}, \boldsymbol{x'}) \equiv \exists z \geq 0 \exists \boldsymbol{x''}(\delta_e(\boldsymbol{x''}) \wedge \boldsymbol{x'} = exp^{zA_{\ell'}}\boldsymbol{x''})$ with free variables among $\{\boldsymbol{x}, \boldsymbol{x'}\}$, where the free variables in the "update" formula δ_e are among $\boldsymbol{x''}$, exp is the base of the natural logarithms, $A_{\ell'}$ is an $n \times n$ rational matrix that is either nilpotent or is diagonalizable with rational eigenvalues ($\boldsymbol{x'} = exp^{zA_{\ell'}}\boldsymbol{x''}$ represents the continuous evolution at the target location ℓ'). It can be shown [LPY99,LPY00] that in these cases, $\alpha_e(\boldsymbol{x}, \boldsymbol{x'})$ is definable in $OF(\mathbb{R})$. In the context of this paper, the most important property of o-minimal systems is that the action in the guarded command does not depend on the current values of variables (these values are relevant only for the guard of the command).

2.2 Parallel Composition

In this section we consider asynchronous parallel composition of hybrid systems [LPY95]. Parallel composition of integer-valued systems is considered in Section 5. We assume that the component programs do not share variables (except for the synchronizing labels). For the purpose of parallel composition, we assign to each guarded command a synchronizing label. Thus with each guarded-command program $\mathcal{S}$ we associate a (finite) set Σ of synchronizing labels and a mapping $lab : \mathcal{E} \longrightarrow \Sigma$ that assigns to each guarded command (or edge) a synchronizing label from Σ.

Given two guarded command programs $\mathcal{S}_1$ and $\mathcal{S}_2$ with label sets Σ_1 and Σ_2 and labeling functions lab_1 and lab_2 respectively, their parallel composition $\mathcal{S} = \mathcal{S}_1 || \mathcal{S}_2$ with set of synchronizing labels $\Sigma_1 \cup \Sigma_2$ and labeling function lab is defined as the set of all guarded commands of the form

$$e \equiv \langle \ell_1, \ell_2 \rangle : \gamma_e(\boldsymbol{x}, \boldsymbol{y}) \, [\!] \alpha_e(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{x'}, \boldsymbol{y'}); \; \mathsf{goto} \; \langle \ell_1', \ell_2' \rangle$$

with $lab(e) = \sigma$ such that either only $\mathcal{S}_1$ "moves" (i.e., takes a transition through an edge) while $\mathcal{S}_2$ undergoes continuous evolution at the same location (if the synchronizing label σ is in Σ_1 but not in Σ_2) or $\mathcal{S}_2$ "moves" while $\mathcal{S}_1$ undergoes continuous evolution at the same location (provided the synchronizing label σ is in Σ_2 but not in Σ_1) or both "move" (if the synchronizing label $\sigma \in \Sigma_1 \cap \Sigma_2$) with the same label $lab(e_1) = lab(e_2) = \sigma$.

Formally, the composed program $\mathcal{S}$ consists of all guarded commands of the form

$$e \;\equiv\; \langle \ell_1, \ell_2 \rangle :\; \gamma_e(\boldsymbol{x}, \boldsymbol{y}) \,[\!]\alpha_e(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{x'}, \boldsymbol{y'});\; \mathsf{goto}\; \langle \ell_1', \ell_2' \rangle$$

with $lab(e) = \sigma$ such that either

- • there is an edge $e_1 \;\equiv\; \ell_1 :\; \gamma_{e_1}(\boldsymbol{x}) \,[\!]\alpha_{e_1}(\boldsymbol{x}, \boldsymbol{x'});\; \mathsf{goto}\; \ell_1'$ in $\mathcal{S}_1$
  - $lab_1(e_1) = \sigma \in \Sigma_1 - \Sigma_2$
  - $\ell_2 = \ell_2'$ is a location in $\mathcal{S}_2$
  - $\gamma_e(\boldsymbol{x}, \boldsymbol{y}) \equiv \gamma_{e_1}(\boldsymbol{x})$
  - $\alpha_e(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{x'}, \boldsymbol{y'}) \equiv \exists z \geq 0\; \varphi_{e_1}(\boldsymbol{x}, \boldsymbol{x'}, z) \wedge \beta_{\ell_2}(\boldsymbol{y}, \boldsymbol{y'}, z)$ where $\alpha_{e_1}(\boldsymbol{x}, \boldsymbol{x'}) \equiv \exists z \geq 0\; \varphi_{e_1}(\boldsymbol{x}, \boldsymbol{x'}, z)$.
- Or same as the previous point but with the roles of $\mathcal{S}_1$ and $\mathcal{S}_2$ reversed.
- Or
  - there is an edge $e_1 \;\equiv\; \ell_1 :\; \gamma_{e_1}(\boldsymbol{x}) \,[\!]\alpha_{e_1}(\boldsymbol{x}, \boldsymbol{x'});\; \mathsf{goto}\; \ell_1'$ in $\mathcal{S}_1$ and an edge $e_2 \;\equiv\; \ell_2 :\; \gamma_{e_2}(\boldsymbol{y}) \,[\!]\alpha_{e_2}(\boldsymbol{y}, \boldsymbol{y'});\; \mathsf{goto}\; \ell_2'$ in $\mathcal{S}_2$
  - $lab_1(e_1) = lab_2(e_2) = \sigma \in \Sigma_1 \cap \Sigma_2$
  - $\gamma_e(\boldsymbol{x}, \boldsymbol{y}) \equiv \gamma_{e_1}(\boldsymbol{x}) \wedge \gamma_{e_2}(\boldsymbol{y})$
  - $\alpha_e(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{x'}, \boldsymbol{y'}) \equiv \exists z \geq 0\; \varphi_{e_1}(\boldsymbol{x}, \boldsymbol{x'}, z) \wedge \varphi_{e_2}(\boldsymbol{y}, \boldsymbol{y'}, z)$ where $\alpha_{e_1}(\boldsymbol{x}, \boldsymbol{x'}) \equiv \exists z \geq 0\; \varphi_{e_1}(\boldsymbol{x}, \boldsymbol{x'}, z)$ and $\alpha_{e_2}(\boldsymbol{y}, \boldsymbol{y'}) \equiv \exists z \geq 0\; \varphi_{e_2}(\boldsymbol{y}, \boldsymbol{y'}, z)$.

A state of the composed program is a tuple $\langle \ell, \ell', \boldsymbol{v}, \boldsymbol{w} \rangle$ consisting of values of the locations and all variables. The semantics of the composed program is defined in the usual way. The parallel composition operation defined above is commutative and associative. For guarded command programs $\mathcal{S}_1, \ldots, \mathcal{S}_k$, we write $\mathcal{S}_1 || \ldots || \mathcal{S}_k$ to denote $(\ldots(\mathcal{S}_1 || \mathcal{S}_2) || \mathcal{S}_3) || \ldots) || \mathcal{S}_k)$. Tools like UPPAAL [BLL+96], HYTECH [HHWT95] use the kind of parallel composition described above (they also use urgent transitions; the framework described below can be easily made to take into account such urgent transitions).

2.3 Constraints Representing Sets of States

In this paper, by constraints we will mean $OF(\mathcal{R})$ formulas. We use constraints φ to represent certain sets of positions. We will consider only conjunctive constraints. A constraint φ is a conjunction of atomic constraints of the form $t \sim c$ where t is a term, $c \in \mathbb{Z}$ and $\sim \in \{>, <, \geq, \leq\}$. We identify solutions of the constraints with states of the system. We write $\mathcal{D}, \langle \ell, \boldsymbol{v} \rangle \models \varphi$ to denote that the state $\langle \ell, \boldsymbol{v} \rangle$ is a solution of the constraint φ where $\mathcal{D}$ is the structure under consideration, i.e., $\langle \mathbb{R}, <, +, \cdot, 0, 1 \rangle$. For a constraint φ, we define the denotation of φ, by $[\varphi]$ as $[\varphi] = \{\langle \ell, \boldsymbol{v} \rangle \mid \mathcal{D}, \langle \ell, \boldsymbol{v} \rangle \models \varphi\}$.

By a set of constraints we mean their disjunction; i.e., if Φ is a set of constraints then $[\Phi] = \bigcup_{\varphi \in \Phi} [\varphi]$. For two constraints φ and φ', we say that φ entails φ', denoted by $\varphi \models \varphi'$, if $[\varphi] \subseteq [\varphi']$. *We identify two constraints φ and φ' if they have the same denotations; i.e.,* $[\varphi] = [\varphi']$. It is known that given two constraints φ and φ' over reals, it is decidable whether $\varphi \models \varphi'$. For a constraint φ with free variables $\boldsymbol{x}$, by $\varphi(\boldsymbol{x'})$ we denote the constraint obtained by renaming the free variables $\boldsymbol{x}$ to $\boldsymbol{x'}$.

2.4 Constraint Transformer $\varphi \mapsto [\![w]\!](\varphi)$

The notion of constraint transformers is inspired by the notion of syntactic transformation monoids in classical automata theory [Eil76]. We write $e_1 \dots e_m$ for the word w obtained by concatenating the 'letters' $e_1, \dots, e_m$ (where each e_i is an edge of a guarded command program); thus, w is a word over the set of edges (or guarded commands) $\mathcal{E}$, i.e., $w \in \mathcal{E}^*$ (assuming that the target location of e_i is the source location of e_{i+1}). The *constraint transformer* with respect to an edge e is the successor constraint function $[\![e]\!]$ that assigns to a constraint φ with free variables $\boldsymbol{x}$ the constraint

$$[\![e]\!](\varphi) \equiv ((\exists \boldsymbol{x}(\varphi \wedge \psi_e))(\boldsymbol{x})).$$

Recall that ψ_e is a formula with free variables among $\boldsymbol{x}, \boldsymbol{x'}$; the variables $\boldsymbol{x}$ are here existentially quantified, and then the variables $\boldsymbol{x'}$ are renamed to $\boldsymbol{x}$. For example, if $\psi_e \equiv L = l_1 \wedge x_1 > 3 \wedge L' = l_2 \wedge \exists z.x'_1 = z \wedge x'_2 = x_2 + z$ then $[\![e]\!](L = l_1 \wedge x_1 > 1 \wedge x_2 = x_1 + 2) \equiv L = l_2 \wedge x_1 > 0 \wedge x_2 > 5$.

The successor constraint function $[\![w]\!]$ with respect to a string $w = e_1 \dots e_m$ of length $m \geq 0$ is the functional composition of the functions with respect to the edges $e_1, \dots, e_m$, i.e., $[\![w]\!] = [\![e_1]\!] \circ \dots \circ [\![e_m]\!]$. Thus $[\![\varepsilon]\!](\varphi) = \varphi$ and $[\![w.e]\!](\varphi) = [\![e]\!]([\![w]\!](\varphi))$. The solutions of $[\![w]\!](\varphi)$ are exactly the successors of the solutions of φ obtained by taking the sequence of transitions through the guarded commands $e_1, \dots, e_m$ (in that order).

We will next recall the definition of special classes of strings called cycles that correspond to cycles in the control graph of the system. We say that an edge of the form $\ell : \dots \mathsf{goto}\ \ell'$ leads from the location ℓ to the location ℓ'. We canonically extend the terminology 'leads to' from edges e to strings w of edges.

Definition 1 (Cycle). *The string $w = e_1 \dots e_m$ of length $m \geq 1$ is a cycle if the sequence of edges $e_1, \dots, e_m$ lead from a location ℓ to itself. A cycle is* trivial *if it consists of one edge whose update part is the constraint $\boldsymbol{x''} = \boldsymbol{x}$. A cycle $e_1 \dots e_m$ is called* simple *if it is not trivial and does not contain a proper subcycle.*

The notion of simple cycles will be used in providing sufficient termination conditions. We call an edge e an *entry* to a cycle w if it leads from a location outside the cycle to a location on the cycle; similarly, e is an *exit* from w if it leads from a location on the cycle to a location outside w.

Initializing strings A transformer $[\![w]\!]$ corresponding to the composition of transformers $[\![e_1]\!], \dots, [\![e_m]\!]$ can be presented by two constraints $\gamma_w(\boldsymbol{x})$ and $\alpha_w(\boldsymbol{x}, \boldsymbol{x'})$ such that for all φ we have $[\![w]\!](\varphi)(\boldsymbol{x'}) \equiv \exists \boldsymbol{x}.\varphi(\boldsymbol{x}) \wedge \gamma_w(\boldsymbol{x}) \wedge \alpha_w(\boldsymbol{x}, \boldsymbol{x'})$ (for better readability we omit here the final renaming of the variables $\boldsymbol{x'}$ to $\boldsymbol{x}$). A string w is called *initializing* if the constraint α_w does not contain any occurrences of the variables $\boldsymbol{x}$; so whenever $\varphi(\boldsymbol{x}) \wedge \gamma_w(\boldsymbol{x})$ is satisfiable, the value of $[\![w]\!](\varphi)(\boldsymbol{x'})$ is simply $\alpha(\boldsymbol{x'})$. An initializing edge is an initializing string consisting of one edge.

We say that w is *weakly initializing* if the set of variables $\boldsymbol{x}$ can be split into two sets $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ such that $\alpha_w(\boldsymbol{x}, \boldsymbol{x'}) \equiv \bigwedge_{x \in \boldsymbol{x_1}} x = x' \wedge \alpha'(\boldsymbol{x'_2})$. We call the variables in $\boldsymbol{x_1}$ the fixed variables of w. We say that a weakly initializing cycle w is *guarded* if either all its entries or all its exits are edges that initialize all fixed variables. Note that every initializing cycle is always guarded (the quantification is over the empty set of variables).

For the cases of non-linear hybrid systems (with the underlying theory being the theory of real closed fields), it can be effectively decided using the methods presented in [Lib00] whether a string is (weakly) initializing. Due to the lack of space we do not detail it out here.

3 Constraint Trees and Symbolic Forward Analysis

Given an infinite state system $\mathcal{S}$ with set of edges $\mathcal{E}$, we define the constraint tree for $\mathcal{S}$ as follows.

Definition 2 (Constraint Tree). *The constraint tree for $\mathcal{S}$ is an infinite tree whose domain is a subset of $\mathcal{E}^*$ (i.e., the nodes are strings over $\mathcal{E}$) that labels the node w by the constraint $[\![w]\!](\varphi^0)$ where φ^0 is the initial constraint.*

Clearly, the (infinite) disjunction of all constraints labeling a node of the constraint tree represents all reachable states of $\mathcal{S}$. We are now in a position to define symbolic forward analysis formally. A symbolic forward analysis is a traversal of (a finite prefix of) a constraint tree in a particular order. The following definition of a non-deterministic procedure abstracts away from that specific order.

Definition 3 (Symbolic Forward Analysis). *A symbolic forward analysis of an infinite state system $\mathcal{S}$ is a procedure that enumerates constraints φ_i labeling the nodes w_i of the constraint tree of $\mathcal{S}$ in a tree order such that the disjunction of the enumerated constraints represents all reachable states of $\mathcal{S}$. Formally,*

- *$\varphi_i = [\![w_i]\!](\varphi^0)$ for $0 \leq i < B$ where the bound B is either a natural number or ω,*
- *if w_i is a prefix of w_j then $i \leq j$,*
- *the disjunction $\bigvee_{0 \leq i < B} \varphi_i$ is equivalent to the disjunction $\bigvee_{0 \leq i < \omega} \varphi_i$.*

The number i is a leaf of a symbolic forward analysis if the node w_i is a leaf of the tree formed by all the nodes w_i where $0 \leq i \leq B$. We say that a symbolic forward analysis terminates if its bound B is finite. We define that a symbolic forward analysis terminates with local entailment if for all its leaves i there exists a $j < i$ such that the constraint φ_i entails the constraint φ_j (as a passing remark, we note that by changing the notion of local entailment, we can get a model checking procedure for liveness properties; we can change the notion of local entailment by requiring that for all leaves i, there exists a $j < i$ such that such that the constraint φ_j entails the constraint φ_i). In contrast, a symbolic forward analysis terminates with global entailment if for all its leaves i, the constraint φ_i entails the disjunction of the constraints φ_j where $j < i$. As discussed in the Introduction, model checking is more efficient with local entailment than with global entailment, both theoretically and practically. Many model checking tools for infinite state systems use local entailment (e.g., UPPAAL [BLL+96], which uses identity; the model checker for infinite state systems with integer-valued variables described in [DP99] also uses local entailment).

Proposition 1. *If every simple cycle of an infinite state system $\mathcal{S}$ is initializing wrt. all variables then symbolic forward analysis for the system terminates with local entailment.*

Proof. We show that the constraint transformer function associated with each initializing string w is either a constant function or unsatisfiable. Suppose that symbolic forward analysis for $\mathcal{S}$ does not terminate with local entailment. Hence, there must be an infinite path p along the constraint tree. Since we have only finitely many locations in $\mathcal{S}$, some of them must occur infinitely often along p, and thus p contains infinitely many occurrences of some simple cycle w; i.e., p is an element of the language $(\mathcal{E}^*.w)^\omega$. Now consider any two nodes $s_1 = w_1.w$ and $s_2 = w_2.w$ of p such that $s_1 < s_2$. Since the constraint transformer function labeling w is a constant function, the constraints labeling s_1 and s_2 are the same and the local entailment check between constraints at these two nodes terminates the analysis along path p, which contradicts the assumption that p is infinite. Hence symbolic forward analysis for $\mathcal{S}$ terminates with local entailment. □

As an immediate application of this proposition we get the following result.

Theorem 1. *Symbolic forward analysis of an o-minimal hybrid system terminates with local entailment.*

Proof. In an o-minimal system every edge and thus every cycle is initializing. □

It is worth noting that the same reasoning gives that backward analysis for o-minimal systems terminates, which was first proved in [LPY99,LPY00].

Proposition 2. *If every simple cycle of a rectangular hybrid system $\mathcal{S}$ is weakly initializing wrt. all variables and is guarded, then symbolic forward analysis for the system $\mathcal{S}$ terminates with local entailment.*

Proof. The reasoning here is similar to that of Proposition 1. In an infinite path p either we have two consecutive occurrences of the same simple cycle which is a constant function wrt. some variables and identity wrt. the others and thus the two constraints corresponding to the two occurrences are the same, or the cycle together with its entry or exit is a constant function. In the letter case, there are only finitely many different constant functions, and thus there must be two occurrences giving the same constraint. □

The propositions above give a termination criteria for forward analysis for infinite-state systems, but in order to apply them one needs an explicit representation of the system. However, usually systems are not represented explicitly but in form of parallel composition of several components. The computation of the explicit representation of a composed system gives an exponential blowup, which we want to avoid. In the following, we look for criteria on the components that allow the use of this proposition.

4 Compositional Reasoning about Termination

In this section, we show how to reason compositionally about sufficient termination conditions in our framework. We provide sufficient conditions on the individual components under which symbolic forward analysis of the parallel composition of n infinite state systems $\mathcal{S}_1, \ldots, \mathcal{S}_n$ terminates.

First notice that just proving termination for individual components is not enough. Consider Figure 1. The figure shows two hybrid systems; each of them is o-minimal and hence symbolic forward analysis for each terminates. The first system consists of two locations ℓ_0 and ℓ_1 and one program variable x which increases with derivative 1 in each location. There is an edge from ℓ_0 to ℓ_1 labeled a. The second system consists of a single location m_0 and an edge from m_0 to itself labeled b. The variable y is the only program variable; it increases with derivative 1 at the location m^0. The initial states (constraints) for both systems are respectively $L = \ell_0 \wedge x = 0$ and $L = m_0 \wedge y = 0$. The asynchronous parallel composition of the two systems is not o-minimal. To see this, consider the transition, where the first system stays (only the variable x increases as the time passes) and the second system moves. Since the value of the variable x after the transition depends on its value before the transition, the composed system is not o-minimal. In fact, symbolic forward analysis does not terminate here: in every iteration of the transition described above the variable x increases by at most two time units which is a non-terminating process.

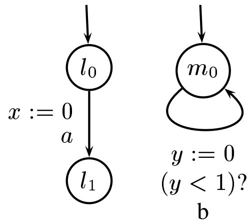

Fig. 1. Parallel composition of two o-minimal hybrid systems is not o-minimal

Theorem 2. *Let $\mathcal{S}_1, \ldots, \mathcal{S}_k$ be k infinite state systems with synchronizing alphabet sets $\Sigma_1, \ldots, \Sigma_k$. If*

- *each simple cycle $w = e_1 \ldots e_m$ ($m \geq 1$) of each $\mathcal{S}_i$ contains an e_j ($1 \leq j \leq m$) such that $lab(e_j) \in \Sigma_1 \cap \ldots \cap \Sigma_k$*
- *and for each $e \in w$ such that $lab(e) \in \Sigma_1 \cap \ldots \cap \Sigma_k$, e is an initializing edge*

then symbolic forward analysis for $\mathcal{S} = \mathcal{S}_1 || \ldots || \mathcal{S}_k$ terminates with local entailment.

Proof. We show that the constraint transformer function associated with each simple cycle in the composed is either a constant function or unsatisfiable.

Let $w = e_1 . \ldots . e_m$ be any simple cycle of $\mathcal{S}$. Then, there exists an edge e such that every component $\mathcal{S}_i$ "moves" on that edge: $lab(e) \in \bigcap_{i=1}^{k} \Sigma_i$. Let $e = e_j$. Consider the projection of e_j on any component $\mathcal{S}_i$. The projection will be an edge e' in this component and also $lab(e') \in \bigcap_{i=1}^{k} \Sigma_i$. Hence, by the assumption of the theorem, e is an initializing edge and the constraint transformer function associated with e is either a constant function or unsatisfiable. Let $w' = e_1 \ldots e_{j-1}$ and $w'' = e_{j+1} \ldots e_j$. Then the constraint transformer function associated with w is given by $[\![w]\!] = [\![w']\!] \circ [\![e_j]\!] \circ [\![w'']\!]$. Hence $[\![w]\!]$ is either a constant function or unsatisfiable. Therefore by Proposition 1, symbolic forward analysis with local entailment terminates. $\square$

To see the applicability of our results, consider the two-process real time Fischer's mutual exclusion protocol given in Figure 2. The critical section is denoted by cs. The processes do not share real variables — the communication is through the synchronization labels. The set of synchronization labels Σ_1 of process P_1 is the set $\{a, b, g, p, t_1\}$ and that for process P_2 the set $\Sigma_2 = \{a, b, g, q, t_2\}$. Each process P_i has only one clock x_i. It can be seen that the protocol satisfies the conditions of Theorem 2: every cycle in each of the systems has an edge with a label in $\{a, b, g\}$. Hence, symbolic forward analysis for the composed system terminates. Note that to find it out we do not have to compute the composition of the two systems (which might be quite big) explicitly. Note also that the composed system is not o-minimal (for example in the transition labeled t_1 the clock x_2 is not reset) and hence the termination results from [LPY99,LPY00] do not apply here.

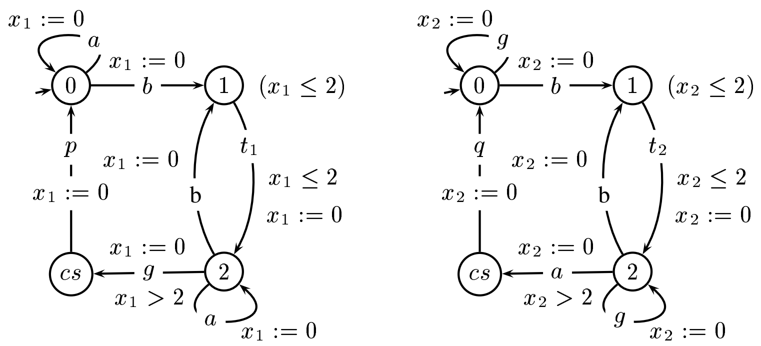

Fig. 2. Fischer's protocol for mutual exclusion of two timed processes

Note that Fischer's protocol is a parallel composition of timed automata. Theorem 2 does not use this fact. It is formulated for arbitrary hybrid systems (where the continuous evolution of particular components over the time might be completely different and thus time could be used as a source of additional communication between components), and can be used in particular to reason about parallel composition of o-minimal systems. The assumptions of the theorem are quite restrictive and are not enough to prove termination for the railroad-crossing example from [AD94,LS85]. It consists of the parallel composition of three components—the train, the gate and the controller. The transition systems (timed automata) for the three components are given in Figure 3. Although every simple cycle in every component is initializing, the assumptions of Theorem 2 are not satisfied since the intersection of $\{lower, down, raise, up\}$ with $\{approach, in, out, exit\}$ is empty.

In the case of timed automata, and more generally of rectangular hybrid systems, due to the uniform evolution of each automaton over the time, we can relax these restrictions. The termination of the forward analysis for the railroad-crossing example follows from Theorem 3 and Observation 1 below: it is enough to choose the controller as the system $\mathcal{A}_1$.

We say that a location ℓ in a rectangular hybrid system is *fixing* if the continuous change of all variables in ℓ satisfies $\dot{x} = 0$. A fixing location ℓ is *guarded* if all edges entering ℓ (or all edges leaving ℓ) initialize all variables of the system. We say that a location ℓ is time-bounded if either ℓ has an invariant $x \leq c$ (or $x < c$) and $\dot{x}$ is positive or it has an invariant $x \geq c$ (or $x > c$) and $\dot{x}$ is negative.

Theorem 3. *Let S be a parallel composition of rectangular non-zeno*[1] *hybrid systems, such that every simple cycle of every component is initializing and every fixing location is guarded. If in every simple cycle of the composed system every component either moves along some cycle or remains in a fixing or time-bounded location, then symbolic forward analysis terminates with local entailment.*

Proof. Suppose that there exists an infinite path p in the constraint tree. Some simple cycle C of the composed system must occur infinitely often along this path. Consider the projection of C on any of the components of the system: it is either a cycle of the component (and then it is initializing wrt. the variables of this component) or a single location that is time-bounded or fixing. If it is a time-bounded location, the system cannot stay at this location forever, therefore C must be a part of a bigger cycle C' in which every component moves or stays in a fixing location. The reasoning then follows the one of Proposition 2. □

The condition that every component moves in every simple cycle of the composed system is still not compositional, but in many cases it is not difficult to find sufficient compositional conditions implying this one—see the two observations below. The first of them applies e.g., to both Fischer's protocol and to timed-automaton version of the railroad crossing; the second to a hybrid version of railroad crossing present in the HYTECH distribution. Together with Theorem 3 above, these observation give sufficient compositional conditions for termination of the forward analysis.

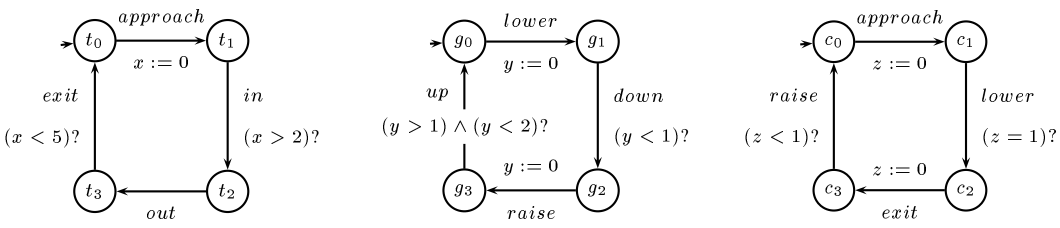

Fig. 3. Railroad crossing: Train, Gate and Controller as timed automata

Observation 1 *Let $\mathcal{A}_1, \ldots, \mathcal{A}_k$ be rectangular hybrid systems. If for all $i = 2, \ldots, k$ each simple cycle in $\mathcal{A}_1$ contains a synchronizing label from Σ_i, and each simple*

[1] Intuitively, a system is non-zeno [AH97] if it "allows" time to grow forever.

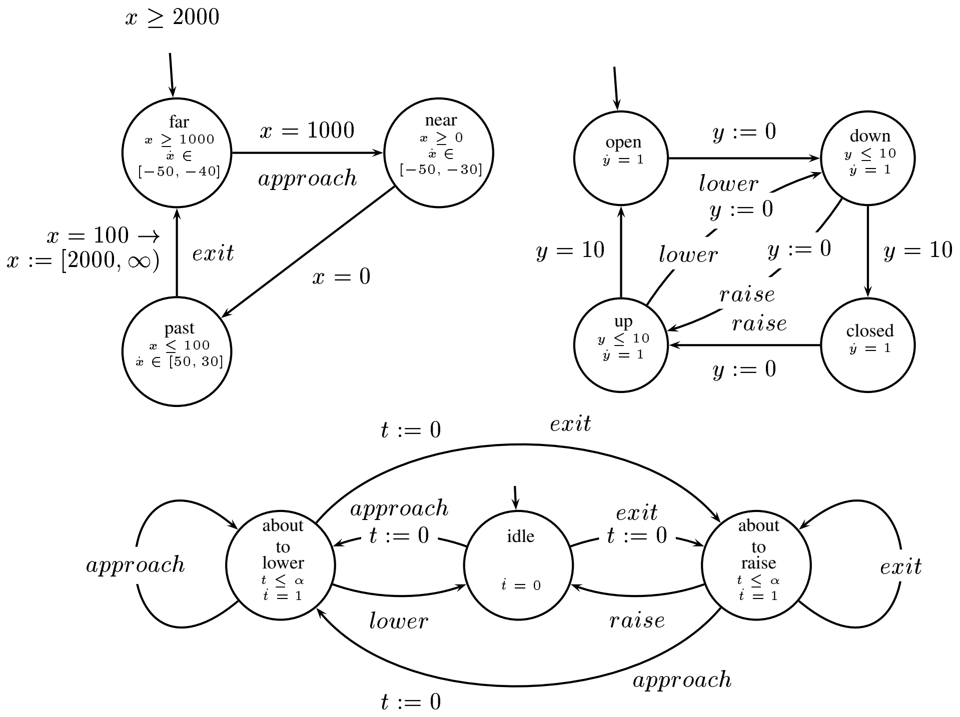

Fig. 4. Railroad crossing: Train, Gate and Controller as rectangular systems

cycle in $\mathcal{A}_i$ contains a synchronizing label from Σ_1 then in every simple cycle of $\mathcal{S} = \mathcal{A}_1 || \ldots || \mathcal{A}_k$ every component $\mathcal{A}_i$ moves.

Proof. The projection of any simple cycle C in the composed system on some of the components is a cycle. This cycle contains a synchronizing label from Σ_1, hence the projection of C on $\mathcal{A}_1$ is a cycle in $\mathcal{A}_1$, which for all i contains a synchronizing label from Σ_i. Therefore the projection of C on every component $\mathcal{A}_i$ is a cycle in $\mathcal{A}_i$ (in contrast, the projection of the only simple cycle of the composed system from Figure 1 on the first component is empty and thus not a cycle). $\square$

Observation 2 *Let $A_1, \ldots, A_k$ be non-zeno rectangular hybrid systems. Suppose that every location is fixing time-bounded. Then in every simple cycle of the composed system every component either moves along some cycle or remains in a fixing or time-bounded location.*

The observation above applies e.g., to the hybrid version of the railroad crossing that can be found in the HYTECH distribution. Of course for this example the symbolic forward analysis terminates, but a very subtle change in the system may lead to non-termination. On Figure 4 we have modified this example by simply modeling the gate as a timed automaton; the other components are not changed at all, and the gate itself behaves essentially in the same way as the original one in HYTECH distribution. Forward analysis for this example does not terminate, because for big enough values of the parameter α the gate may stay forever in the location open or closed, while the controller switches between about-to-lower and about-to-raise; every iteration of this cycle increases the values for the variable y by at most 2α, which is a nonterminating process. The observation above of course does not apply to this system (time is not bounded in locations open and closed), but it gives a hint how to improve the system such that the analysis terminates: changing $\dot{y}$ from 1 to 0 in both these locations forces the analysis to terminate without any essential changes in the behavior of the system (the variable y after leaving these locations is reset and thus its value is not needed; in particular both safety and liveness properties remain unchanged). Note that both these lacations are guarded (in fact they are doubly guarded: the entering edges initialize y to 10 and exit edges initialize y to 0).

We are implementing the static tests based on the observations above on the top of the model checker described in [DP99].

5 Integer-valued systems

The composed program $\mathcal{S} = \mathcal{S}_1 \parallel \mathcal{S}_2$ consists of all guarded commands of the form

$$e \;\equiv\; \langle \ell_1, \ell_2 \rangle : \; \gamma_e(\boldsymbol{x}, \boldsymbol{y}) \,[\!]\, \alpha_e(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{x'}, \boldsymbol{y'}); \;\mathsf{goto}\; \langle \ell_1', \ell_2' \rangle$$

such that either there exists an edge $e_1 \;\equiv\; \ell_1 \;:\; \gamma_{e_1}(\boldsymbol{x}) \,[\!]\, \alpha_{e_1}(\boldsymbol{x}, \boldsymbol{x'});\; \mathsf{goto}\; \ell_1'$ in $\mathcal{S}_1$ where $\gamma_e = \gamma_{e_1}$, $\ell_2' = \ell_2$ and $\alpha_e = \alpha_{e_1} \wedge \bigwedge_{y \in \boldsymbol{y} - \boldsymbol{x}} y' = y$, or the symmetrical condition with the roles of $\mathcal{S}_1$ and $\mathcal{S}_2$ reversed holds.

Integer-valued systems in general are not well-suited for symbolic forward analysis (without acceleration techniques such as widening) as the following observations show. We say that an edge e increments a variable x if the action α_e contains a conjunct $x' = x + 1$.

Proposition 3. *Suppose that a variable x has initially value 0 in a system $\mathcal{S} = \mathcal{S}_1 \parallel \ldots \parallel \mathcal{S}_n$, the component S_1 contains a simple cycle C with an edge e incrementing the variable x, and e is the only edge changing the value of x in $\mathcal{S}$. Then either C is executed only finitely many times in every infinite execution of $\mathcal{S}$ or forward analysis does not terminate.*

Sketch of proof. Suppose that C is executed infinitely many times. Then the constraint tree T for $\mathcal{S}$ contains an infinite path corresponding to this infinite execution. The nodes corresponding to the i-th execution of the cycle C in T are constraints containing a conjunct $x = i$, thus T contains infinitely many inequivalent constraints. $\square$

The proposition above applies to many integer-valued protocols including the ticket protocol and bounded-buffer producer/consumer protocol. It does not, however apply to bakery protocol.

Observation 3 *Suppose that a variable x has initially value 0 in a system $\mathcal{S} = \mathcal{S}_1 \parallel \ldots \parallel \mathcal{S}_n$, the component S_i contains a simple cycle C_i with an edge with the action $x_{i+1 \bmod n} := x_i + 1$ and C_i does not modify $x_{i+2 \bmod n}$. Then forward analysis is not likely to terminate.*

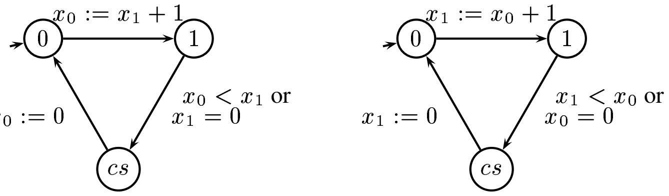

Fig. 5. Bakery protocol for mutual exclusion

The reason for possible non-termination of the forward analysis is an infinite execution of the system that consecutively increases the values of variables, thus creating infinitely many nonequivalent constraints in the constraint tree.

The observation above gives an idea of changing the protocol a little bit to make the analysis terminate: one can replace the action of the form $x := y+1$ by $x := [y+1, \infty)$, where the new value of x is any number greater than the value of y instead of concrete value $y+1$. The modified version of bakery protocol preserves its main property of mutual exclusion and starvation-freeness, but the forward analysis terminates. The above transformation can be applied on-the-fly as a widening operation.

6 Related Work

Reachability analysis for infinite state systems with integer valued variables has been considered by Berard and Fribourg [BF99] as well as by Fribourg and Olsen [FO97], but the authors do not provide any sufficient conditions for termination of their model checking procedure on any interesting class of systems.

Abdulla, Cerans, Jonsson and Tsay [ACJT96] as well as Finkel and Schnoebelen [FS98] gave a unifying framework for deriving decidability results for model checking for infinite state systems. However, their framework requires finding a well quasi-ordering on the states, which in many practical situations is not feasible. Besides, their method of deriving sufficient termination conditions for reachability analysis is monolithic; one has to consider the state-space of the composed system to show the termination of reachability analysis.

Comon and Jurski [CJ98] obtained decidability results for reachability analysis for a fragment of the class of multiple counter automata. They showed that the fixpoint of iterating transitions for this subclass of multiple counter automata is expressible in Presburger arithmetic. Again, their framework does not provide any means of reasoning about sufficient termination conditions compositionally.

Boigelot [Boi98] obtained sufficient conditions for termination of reachability analysis for infinite state systems with integer-valued variables based on graph-theoretic properties of the underlying control graphs. However, like the works mentioned above, his work does not provide a compositional way of reasoning about sufficient termination conditions.

Bultan, Gerber and Pugh [BGP97] presented a model checker for infinite state systems with integer-valued variables based on the Presburger solver from the Omega library [Pug92]. While [BGP97] provided model checking procedures for both safety and liveness properties, no sufficient conditions for termination of the procedures were provided.

Wong-Toi [WT95] has identified a subclass of linear hybrid systems called skewed clock automata that can be translated to timed safety automata. The subclass of skewed clock automata is closed under parallel composition. While symbolic backward analysis is guaranteed to terminate for skewed clock automata, symbolic forward analysis is possibly non-terminating for this subclass.

Non-linear hybrid systems have been considered by Lafferriere, Pappas and Yovine [LPY99,LPY00]. For the class of o-minimal hybrid systems, they proved the termination of symbolic backward analysis by showing that this class admits finite bisimulations. The reasoning about termination of symbolic backward analysis in [LPY99,LPY00] is not compositional.

Henzinger, Kopke, Puri and Varaiya [HKPV95] considered initialized rectangular automata, a subclass of linear hybrid systems, for which symbolic backward analysis is guaranteed to terminate. Henzinger [Hen95] considered hybrid automata with finite bisimulations for which symbolic backward analysis is guaranteed to terminate. But none of these works addressed the issue of compositional reasoning about sufficient termination conditions.

Lam and Brayton [LB93] considered alternating RQ timed automata closed under I/O composition. The class of alternating RQ automata is restrictive in the sense that it allows exactly one reset and exactly one query for each clock in an entire automaton. Moreover the notion of I/O composition that they used is much more restrictive than the notion of parallel composition used in this paper. It is also not known whether symbolic forward analysis for alternating RQ timed automata is guaranteed to terminate.

Namjoshi [Nam98] considered model checking for parameterized systems in which each process is finite state. In contrast, in this paper, we considered finite families of possibly infinite state systems.

In [MP99], we gave a framework for reasoning about syntactic sufficient termination conditions for timed automata. The present work is an extension of that framework to the more general context of hybrid systems as well as augmenting the framework with compositional reasoning.

References

[ACHH93] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, editors, *Hybrid Systems I*, LNCS 736, pages 209–229. Springer-Verlag, 1993.

[ACJT96] P. Abdulla, K. Cerans, B. Jonsson, and T. K. Tsay. General decidability theorems for infinite state systems. In *LICS*, pages 313–321, 1996.

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–236, 1994.

[AH97] R. Alur and T. A. Henzinger. Modularity for timed and hybrid systems. In A. Mazurkiewicz and J. Winkowski, editors, *CONCUR'97: Concurrency Theory*, volume 1243 of *LNCS*, pages 74–88. Springer-Verlag, 1997.

[BF99] B. Berard' and L. Fribourg. Reachability analysis of (timed) petri nets using real arithmetic. In J. C. M. Baeten and S. Mauw, editors, *CONCUR: Concurrency Theory*, volume 1664 of *LNCS*, pages 178–193. Springer-Verlag, 1999.

[BGP97] T. Bultan, R. Gerber, and W. Pugh. Symbolic model checking of infinite state systems using presburger arithmetics. In Orna Grumberg, editor, *the 9th International Conference on Computer Aided Verification (CAV'97)*, LNCS 1254, pages 400–411. Springer, Haifa, Israel, July 1997.

[BLL+96] Johan Bengtsson, Kim. G. Larsen, Fredrik Larsson, Paul Petersson, and Wang Yi. Uppaal in 1995. In T. Margaria and B. Steffen, editors, *TACAS*, LNCS 1055, pages 431–434. Springer-Verlag, 1996.

[Boi98] Bernard Boigelot. *Symbolic Methods for Exploring Infinite State Spaces*. PhD thesis, Universite De Liege, Montefiore, Belgium, 1998.

[CJ98] H. Comon and Y. Jurski. Multiple Counters Automata, Safety Analysis, and Presburger Arithmetics. In Alan J. Hu and M. Y. Vardi, editors, *CAV'98: Computer Aided Verification*, volume 1427 of *LNCS*, pages 268–279. Springer-Verlag, 1998.

[DP99] G. Delzanno and A. Podelski. Model Checking in CLP. In R. Cleaveland, editor, *Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'99)*, volume 1579 of *LNCS*, pages 223–239. Springer-Verlag, March 1999.

[Eil76] S. Eilenberg. *Automata, Languages and Machines*, volume B. Academic Press, 1976.

[FO97] L. Fribourg and H. Olsen. A Decompositional Approach for Computing Least Fixed Point of Datalog Programs with Z-counters. *Journal of Constraints*, 2(3-4):305–336, 1997.

[FS98] A. Finkel and P. Schnoebelen. Well-structured Transition Systems Everywhere! Technical Report LSV-98-4, Laboratoire Spécification et Vérification, Ecole Normale Supérieure de Cachan, 1998.

[Hen95] T.A. Henzinger. Hybrid automata with finite bisimulations. In Z. Fülöp and F. Gécseg, editors, *ICALP 95: Automata, Languages, and Programming*, LNCS 944, pages 324–335. Springer-Verlag, 1995.

[HHWT95] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HyTech. In E. Brinksma, W.R. Cleaveland, K.G. Larsen, T. Margaria, and B. Steffen, editors, *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 1019, pages 41–71. Springer-Verlag, 1995.

[HKPV95] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *the 27th Annual Symposium on Theory of Computing*, pages 373–382. ACM Press, 1995.

[LB93] W. K. C. Lam and R. K. Brayton. Alternating RQ timed automata. In Costas Courcoubetis, editor, *the 5th International Conference on Computer-Aided Verification*, LNCS 697, pages 236–252. Springer-Verlag, June/July 1993.

[Lib00] L. Libkin. Variable independence, quantifier elimination, and constraint representations. In *ICALP: International Colloqium on Automata Languages and Programming*, 2000.

[LPY95] K.G. Larsen, P. Pettersson, and W. Yi. Compositional and symbolic model checking of real-time systems. In *Proceedings of the 16th Annual Real-time Systems Symposium*, pages 76–87. IEEE Computer Society Press, 1995.

[LPY99] G. Lafferriere, G. J. Pappas, and S. Yovine. A new class of decidable hybrid systems. In F. W. Vaandrager and J. H. van Schuppen, editors, *Hybrid Systems, Computation and Control*, volume 1569 of *LNCS*, pages 137–151, 1999.

[LPY00] G. Lafferriere, G. J. Pappas, and S. Yovine. O-minimal hybrid systems. *Mathematics of Control, Signals and Systems*, 13(1):1–21, March 2000.

[LS85] N. G. Leveson and J. L. Stolzy. Analyzing safety and fault tolerance using time petri nets. In H. Uhrig, C. Floyd, M. Nivat, and J. W. Thatcher, editors, *TAPSOFT: Theory and Practice of Software*, volume 186 of *LNCS*, pages 339–355. Springer, 1985.

[MP99] S. Mukhopadhyay and A. Podelski. Beyond region graphs: Symbolic forward analysis of timed automata. In C. Pandurangan, V. Raman, and R. Ramanujam, editors, *19th International Conference on the Foundations of Software Technology and Theoretical Computer Science*, volume 1738 of *LNCS*, pages 233–245, December 1999.

[Nam98] K. S. Namjoshi. *Ameliorating the State Explosion Problem*. PhD thesis, The Graduate School of the University of Texas at Austin, 1998.

[Pug92] W. Pugh. The omega test: a fast and practical integer programming algorithm for dependence analysis. *Communications of the ACM*, 8:102–104, 1992.

[WT95] H. Wong-Toi. *Symbolic Approximations for Verifying Real-Time Systems*. PhD thesis, Stanford University, 1995.