Beyond Region Graphs: Symbolic Forward Analysis of Timed Automata

Supratik Mukhopadhyay and Andreas Podelski

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

Abstract. Theoretical investigations of infinite-state systems have so far concentrated on decidability results; in the case of timed automata these results are based on region graphs. We investigate the specific procedure that is used practically in order to decide verification problems, namely symbolic forward analysis. This procedure is possibly non-terminating. We present basic concepts and properties that are useful for reasoning about sufficient termination conditions, and then derive some conditions. The central notions here are constraint transformers associated with sequences of automaton edges and *zone trees* labeled with successor constraints.

1 Introduction

A *timed automaton* [AD94] models a system whose transitions between finitely many control locations depend on the values of clocks. The clocks advance continuously over time; they can individually be reset to the value 0. Since the clocks take values over reals, the state space of a timed automaton is infinite.

The theoretical and the practical investigations on timed automata are recent but already quite extensive (see e.g. [AD94,HKPV95,LPY95,Bal96,DT98]). Many decidability results are obtained by designing algorithms on the *region graph*, which is a finite quotient of the infinite state transition graph [AD94]. Practical experiments showing the feasibility of model checking for timed automata, however, employ *symbolic forward analysis*. We do not know of any practical tool that constructs the region graph. Instead, symbolic model checking is extended directly from the finite to the infinite case; logical formulas over reals are used to 'symbolically' represent infinite sets of tuples of clock values and are manipulated by applying the same logical operations that are applied to Boolean formulas in the finite state case.

If model checking is based on *backward* analysis (where one iteratively computes sets of predecessor states), termination is guaranteed [HNSY94]. In comparison, symbolic forward analysis for timed automata has the theoretical disadvantage of possible non-termination. Practically, however, it has the advantage that it is amenable to on-the-fly local model checking and to partial-order reduction techniques (see [HKQ98] for a discussion of forward vs. backward analysis).

In symbolic forward analysis applied to the timed automata arising in practical applications (see e.g. [LPY95]), the theoretical possibility of non-terminating does not seem to play a role. Existing versions that exclude this possibility (through built-in runtime checks [DT98] or through a static preprocessing step [HKPV95]) are not used in practice.

This situation leads us to raising the question whether there exist 'interesting' sufficient conditions for the termination of symbolic model checking procedures for timed automata based on forward analysis. Here, 'interesting' means applicable to a large class of cases in practical applications. The existence of a practically relevant class of infinite-state systems for which the practically employed procedure is actually an algorithm would be a theoretically satisfying explanation of the success of the ongoing practice of using this procedure, and it may guide us in designing practically successful verification procedures for other classes of infinite-state systems.

As a first step towards answering the question that we are raising, we build a kind of 'toolbox' consisting of basic concepts and properties that are useful for reasoning about sufficient termination conditions. The central notions here are constraint transformers associated with sequences of automaton edges and *zone trees* labeled with successor constraints. The constraint transformer associated with the sequences of edges $e_1, \ldots, e_n$ of the timed automaton assigns a constraint φ another constraint that 'symbolically' represents the set of the successor states along the edges $e_1, \ldots, e_n$ of the states in the set represented by φ. We prove properties for constraint transformers associated with edge sequences of a certain form; these properties are useful in termination proofs as we then show. The zone tree is a vehicle that can be used to investigate sufficient conditions for termination without having to go into the algorithmic details of symbolic forward analysis procedures. It captures the fact that the constraints enumerated in a symbolic forward analysis must respect a certain tree order.

We show how the zone tree can characterize termination of (various versions of) symbolic forward analysis. A combinatorial reasoning is then used to derive sufficient termination conditions for symbolic forward analysis. We prove that symbolic forward analysis terminates for three classes of timed automata. These classes are not relevant practically; the goal is merely to demonstrate how the presented concepts and properties of the successor constraint function and of the zone tree can be employed to prove termination. Termination proofs can be quite tedious, as the third case shows; the proof here distinguishes many cases.

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

A *timed automaton* $\mathcal{U}$ can, for the purpose of reachability analysis, be defined as a set $\mathcal{E}$ of guarded commmands e (called edges) of the form below. Here L is a variable ranging over the finite set of *locations*, and $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$ are the variables standing for the clocks and ranging over nonnegative real numbers. As usual, the primed version of a variable stands for its value after the transition.

The 'time delay' variable z ranges over nonnegative real numbers.

$$e \;\equiv\; L = \ell \wedge \gamma_e(\boldsymbol{x}) \;\|\; L' = \ell' \wedge \alpha_e(\boldsymbol{x}, \boldsymbol{x'}, z).$$

The guard formula $\gamma_e(\boldsymbol{x})$ over the variables $\boldsymbol{x}$ is built up from conjuncts of the form $x_i \sim k$ where x_i is a clock variable, $\sim$ is a comparison operator (i.e., $\sim \in \{=, <, \leq, >, \geq\}$) and k is a natural number.

The action formula $\alpha_e(\boldsymbol{x}, \boldsymbol{x'}, z)$ of e is defined by a subset Reset_e of $\{1, \ldots, n\}$ (denoting the clocks that are *reset*); it is of the form

$$\alpha_e(\boldsymbol{x}, \boldsymbol{x'}, z) \;\equiv\; \bigwedge_{i \in \mathrm{Reset}_e} x'_i = z \wedge \bigwedge_{i \notin \mathrm{Reset}_e} x'_i = x_i + z.$$

We write ψ_e for the logical formula corresponding to e (with the free variables $\boldsymbol{x}$ and $\boldsymbol{x'}$; we replace the guard symbol $\|$ with conjunction).

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

The states of $\mathcal{U}$ (called *positions*) are tuples of the form $\langle \ell, \boldsymbol{v} \rangle$ consisting of values for the location and for each clock. The position $\langle \ell, \boldsymbol{v} \rangle$ can make a *time transition* to any position $\langle \ell, \boldsymbol{v} + \delta \rangle$ where $\delta \geq 0$ is a real number.

The position $\langle \ell, \boldsymbol{v} \rangle$ can make an *edge transition* (followed by a time transition) to the position $\langle \ell', \boldsymbol{v'} \rangle$ using the edge e if the values ℓ for L, $\boldsymbol{v}$ for $\boldsymbol{x}$, ℓ' for L' and $\boldsymbol{v'}$ for $\boldsymbol{x'}$ define a solution for ψ_e. (An edge transition by itself is defined if we replace the variable z in the formula for α by the constant 0.)

We use *constraints* φ in order to represent certain sets of positions (called *zones*). A constraint is a conjunction of the equality $L = \ell$ with a conjunction of formulas of the form $x_i - x_j \sim c$ or $x_i \sim c$ where c is an integer (i.e. with a *zone constraint* as used in [DT98]). We identify solutions of constraints with positions $\langle \ell, \boldsymbol{v} \rangle$ of the timed automaton.

We single out the *initial constraint* φ^0 that denotes the time successors of the initial position $\langle \ell^0, \boldsymbol{0} \rangle$.

$$\varphi^0 \;\equiv\; L = \ell^0, x_1 \geq 0, x_2 = x_1, \ldots, x_n = x_1$$

A constraint φ is called *time-closed* if its set of solutions is closed under time transitions. Formally, $\varphi(\boldsymbol{x})$ is equivalent to $(\exists \boldsymbol{x} \exists z(\varphi \wedge x'_1 = x_1 + z \wedge \ldots \wedge x'_n = x_n + z))[\boldsymbol{x}/\boldsymbol{x'}]$. For example, the initial constraint is time-closed. In the following, we will be interested only in time-closed constraints.

In the definition below, $\varphi'[\boldsymbol{x'}/\boldsymbol{x}]$ denotes the constraint obtained from φ' by α-renaming (replace each x'_i by x_i).

We write $e_1 \ldots . e_m$ for the word w obtained by concatenating the 'letters' $e_1, \ldots, e_m$; thus, w is a word over the set of edges $\mathcal{E}$, i.e. $w \in \mathcal{E}^\star$.

Definition 1 (Constraint Transformer $[\![w]\!]$). *The constraint transformer wrt. to an edge e is the 'successor constraint function' $[\![w]\!]$ that assigns a constraint φ the constraint*

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

The successor constraint function $[\![w]\!]$ *wrt. a string* $w = e_1 \ldots . e_m$ *of length* $m \geq 0$ *is the functional composition of the functions wrt. the edges* $e_1, \ldots, e_m$, *i.e.* $[\![w]\!] = [\![e_1]\!] \circ \ldots \circ [\![e_m]\!]$.

Thus, $[\![\varepsilon]\!](\varphi) = \varphi$ and $[\![w.e]\!](\varphi) = [\![e]\!]([\![w]\!](\varphi))$. The solutions of $[\![w]\!](\varphi)$ are exactly the ("edge plus time") successors of a solution of φ by taking the sequence of transitions via the edges $e_1, \ldots, e_m$ (in that order).

We will next consider constraint transformers $[\![w]\!]$ for strings w of a certain form. In the next definition, the terminology 'a clock x_i is queried in the edge e' means that x_i is a variable occurring in the guard formula γ of e; 'x_i is reset in e' means that $i \in \mathrm{Reset}_e$.

Definition 2 (Stratified Strings). *A string* $w = e_1 \ldots . e_m$ *of edges is called stratified if*

- *each clock* $x_1, \ldots, x_n$ *is reset at least once in* w, *and*
- *if* x_i *is reset in* e_j *then* x_i *is not queried in* $e_1, \ldots, e_j$.

Proposition 1. *The successor constraint function wrt. a stratified string* w *is a constant function over satisfiable constraints (i.e. there exists a unique constraint* φ_w *such that* $[\![w]\!](\varphi) = \varphi_w$ *for all satisfiable constraints* φ*).*

Proof. We express the successor constraint of the constraint φ wrt. the stratified string $w = e_1 \ldots e_m$ equivalently by

$$[\![w]\!](\varphi) \;\equiv\; (\exists \boldsymbol{x} \exists \boldsymbol{x^1} \ldots \exists \boldsymbol{x^{m-1}} \exists z^1 \ldots \exists z^m (\varphi \wedge \psi_1 \wedge \ldots \wedge \psi_m))[\boldsymbol{x}/\boldsymbol{x^m}]$$

where ψ_k is the formula that we obtain by applying α-renaming to the (quantifier-free) conjunction of the guard formula $\gamma_{e_k}(\boldsymbol{x})$ and the action formula $\alpha_{e_k}(\boldsymbol{x}, \boldsymbol{x'}, z)$ for the edge e_k; i.e.

$$\psi_k \;\equiv\; \gamma_{e_k}(\boldsymbol{x^{k-1}}) \wedge \alpha_{e_k}(\boldsymbol{x^{k-1}}, \boldsymbol{x^k}, z^k).$$

Thus, in the formula for e_k, we rename the clock variable x_i to x_i^{k-1}, its primed version x_i' to x_i^k, and the 'time delay' variable z to z^k.

We identify the variables x_i (applying in φ) with their "0-th renaming" x_i^0 (appearing in ψ_1); accordingly we can write $\boldsymbol{x^0}$ for the tuple of variables $\boldsymbol{x}$.

We will transform $\exists \boldsymbol{x^1} \ldots \exists \boldsymbol{x^{m-1}} (\psi_1 \wedge \ldots \wedge \psi_m)$ equivalently to a constraint ψ containing only conjuncts of the form $x_i^m = z^l + \ldots + z^m$ and of the form $z^l + \ldots + z^m \sim c$ where $l > 0$; i.e. ψ does not contain any of the variables x_i of φ. Thus, we can move the quantifiers $\exists \boldsymbol{x}$ inside; formally, $\exists \boldsymbol{x}(\varphi \wedge \psi)$ is equivalent to $(\exists \boldsymbol{x} \varphi) \wedge \psi$. Since φ is satisfiable, the conjunct $\exists \boldsymbol{x} \varphi$ is equivalent to *true*. Summarizing, $[\![w]\!](\varphi)$ is equivalent to a formula that does not depend on φ, which is the statement to be shown.

The variable x_i^k (the "k-th renaming of the i-th clock variable") occurs in the action formula of ψ_k, either in the form $x_i^k = z^k$ or in the form $x_i^k = x_i^{k-1} + z^k$, and it occurs in the guard formula of ψ_{k+1}, in the form $x_i^k \sim c$.

If the i-th clock is not reset in the edges $e_1, \ldots, e_{k-1}$, then we replace the conjunct $x_i^k = x_i^{k-1} + z^k$ by $x_i^k = x_i + z^1 + \ldots z^k$.

Otherwise, let l be the largest index of an edge e_l with a reset of the i-th clock. Then we replace $x_i^k = x_i^{k-1} + z^k$ by $x_i^k = z^l + \ldots + z^k$.

If $k = m$, the first case cannot arise due to the first condition on stratified strings (the i-th clock must be reset at least once in the edges $e_1, \ldots, e_m$). That is, we replace $x_i^m = x_i^{m-1} + z^k$ always by a conjunct of the form $x_i^k = z^l + \ldots + z^k$.

If the conjunct $x_i^k \sim c$ appears in ψ_{k+1}, then, by assumption on w (the second condition for stratified strings), the i-th clock is reset in an edge e_l where $l \leq k$. Therefore, we can replace the conjunct $x_i^k \sim c$ by $z_l + \ldots + z_k \sim c$.

Now, each variable x_i^k (for $0 < k < m$) has exactly one occurrence, namely in a conjunct C of the form $x_i^k = x_i + z^1 + \ldots z^k$ or $x_i^k = z^l + \ldots z^k$. Hence, the quantifier $\exists x_i^k$ can be moved inside, before the conjunct C; the formula $\exists x_i^k\ C$ can be replaced by *true*.

After the above replacements, all conjuncts are of the form $x_i^m = z^l + \ldots + z^m$ or of the form $z^l + \ldots + z^m \sim c$; as explained above, this is sufficient to show the statement. ∎

We say that an edge e is *reset-free* if $\mathrm{Reset}_e = \emptyset$, i.e., its action is of the form $\alpha_e \equiv \bigwedge_{i=1,\ldots,n} x_i' = x_i$. A string w of edges is reset-free if all its edges are.

Proposition 2. *If the string w is reset-free, and the successor constraint of a time-closed constraint of the form $L = \ell \wedge \varphi$ is of the form $L = \ell' \wedge \varphi'$, then φ' entails φ, formally $\varphi' \models \varphi$.*

Proof. It is sufficient to show the statement for w consisting of only one reset-free edge e. Since φ is time-closed, it is equivalent to $(\exists \boldsymbol{x} \exists z(\varphi \wedge \boldsymbol{x'} = \boldsymbol{x} + z))[\boldsymbol{x}/\boldsymbol{x'}]$.

Then $[\![w]\!](L = \ell \wedge \varphi)$ is equivalent to $(\exists \ldots (L = \ell' \wedge \varphi \wedge \boldsymbol{x'} = \boldsymbol{x} + z' \wedge \gamma_e(\boldsymbol{x'}) \wedge \boldsymbol{x''} = \boldsymbol{x'} + z')[\boldsymbol{x}/\boldsymbol{x''}]$. This constraint is equivalent to $L = \ell' \wedge \varphi(\boldsymbol{x}) \wedge \gamma(\boldsymbol{x})$. This shows the statement. ∎

3 Zone Trees and Symbolic Forward Analysis

Definition 3 (Zone Tree). *The zone tree of a timed automaton $\mathcal{U}$ is an infinite tree with domain $\mathcal{E}^\star$ (i.e., the nodes are the strings over $\mathcal{E}$) that labels the node w by the constraint $[\![w]\!](\varphi^0)$.*

That is, the root ε is labeled by the initial constraint φ^0. For each node w labeled φ, and for each edge $e \in \mathcal{E}$ of the timed automaton, the successor node $w.e$ is labeled by the constraint $[\![e]\!](\varphi)$. Clearly, the (infinite) disjunction of all constraints labeling a node of the zone tree represents all reachable positions of $\mathcal{U}$.

We are interested in the termination of various versions of symbolic forward analysis of a timed automaton $\mathcal{U}$. All versions have in common that they traverse (a finite prefix of) its zone tree, in a particular order. The following definition of a non-deterministic procedure abstracts away from that specific order.

Definition 4 (Symbolic Forward Analysis). *A symbolic forward analysis of a timed automaton $\mathcal{U}$ is a procedure that enumerates constraints φ_i labeling the nodes w_i of the zone tree of $\mathcal{U}$ in a tree order such that the enumerated constraints together represent all reachable positions. Formally,*

– $\varphi_i = [\![w_i]\!](\varphi^0)$ *for* $0 \leq i < B$ *where the bound* B *is 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* $\bigvee_{0 \leq i < \omega} \varphi_i$.

We assume that the constraint φ_i is computed by applying any of the known quantifier elimination algorithms (see e.g. [MS98]) to a conjunction of constraints.

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 the bound B is finite (i.e. not ω). We define that symbolic forward analysis terminates *with local subsumption* if for all its leafs i there exists $j < i$ such that the constraint φ_i entails the constraint φ_j. In contrast, it terminates with *global* subsumption if for all its leafs i there the constraint φ_i entails the disjunction of all constraints φ_j where $j < i$. Model checking is more efficient with local subsumption than with global subsumption, both practically and theoretically [DP99].

A depth-first symbolic forward analysis depends on a chosen order of edges. Symbolic forward analysis terminates if and only if the depth-first symbolic forward analysis of $\mathcal{U}$ terminates for *every* order chosen.

If the symbolic depth-first forward analysis of $\mathcal{U}$ terminates for at least one order of edges, then also the breadth-first version terminates. The converse need not be true, as the counterexample of Figure 1 shows.

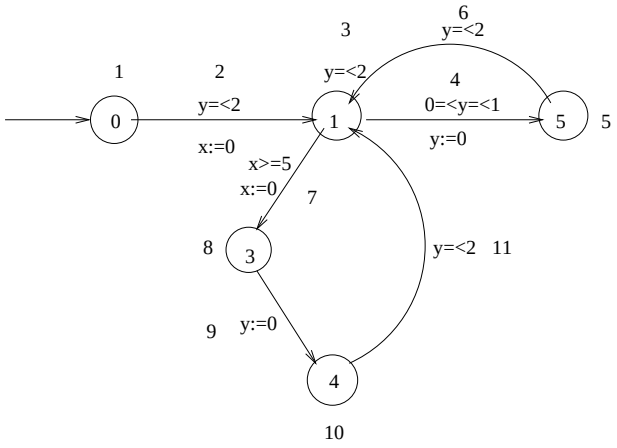

Fig. 1. Example of a timed automaton for which the breadth-first version of symbolic forward analysis terminates but the depth-first version does not, if the edge numbered 4 is followed before the edge numbered 7.

A *path* p in a zone tree is an infinite string over $\mathcal{E}$, i.e., $p \in \mathcal{E}^\omega$; p contains a node w if the string w is a prefix of p, written $w < p$. A node v precedes a node w if v is a prefix of w, written $v < p$.

Definition 5 (Local finiteness). *A path p of a zone tree is locally finite if and only if it contains a node w labeled by a constraint that entails the constraint labeling some node v preceding w (formally, there exist v and w such that $v < w < p$ and $[\![w]\!](\varphi^0) \models [\![v]\!](\varphi^0)$). A zone tree is locally finite if every path is.*

Proposition 3. Every *symbolic forward analysis of a timed automaton $\mathcal{U}$ terminates with local subsumption if and only if the zone tree of $\mathcal{U}$ is locally finite.*

We will next investigate the special class of *strings* (that we call *cycles*) that correspond to cycles in the control graph of the given timed automaton. Each cycle in the graph-theoretic sense corresponds to finitely many cycles in the sense defined here (as strings), depending on the entry location.

We say that an edge e of the form $L = \ell \ldots \parallel L' = \ell' \ldots$ leads from the location ℓ to the location ℓ'. This terminology refects the fact that there exists a directed edge from ℓ to ℓ' labeled by the corresponding guarded command in the control graph of the given timed automaton (we will not formally introduce the control graph). Semantically, all transitions using such an edge go from a position with the location ℓ to a position with the location ℓ'. We canonically extend the terminology 'leads to' from edges e to strings w of edges.

Definition 6 (Cycle). *The string $w = e_1 \ldots . e_m$ of length $m \geq 1$ is a cycle if the sequence of edges e_1, ..., e_m lead from a location ℓ to the same location ℓ such that there exists a sequence of edges that leads from the initial location ℓ^0 to ℓ whose last edge is different from e_m.*

The last condition above expresses that ℓ is an entry point to the corresponding cycle in the control graph of the given timed automaton $\mathcal{U}$. The next notion is used in effective sufficient termination conditions.

Definition 7 (Simple Cycle). *A cycle $w = e_1 \ldots . e_m$ is called simple if it does not contain a proper subcycle; formally, no string $e_i \ldots . e_j$ where $1 \leq i < j \leq m$ is also a cycle.*

Proposition 4. *A locally infinite path $p \in \mathcal{E}^\omega$ in the zone tree of the timed automaton $\mathcal{U}$ contains infinitely many occurrences of a simple cycle w; formally, p is an element of the omega-language $(\mathcal{E}^\star.w)^\omega$.*

Proof. Let p be a locally infinite path. Then there exists a location ℓ such that infinitely many nodes on this path are labeled by ℓ (i.e. a constraint of the form $L = \ell \wedge \ldots$. The strings formed by the edges connecting two nodes labeled by ℓ must all contain a simple cycle. Since the number of simple cycles is finite, some simple cycles must be repeated infinitely often. ∎

A string is *stratifiable* if contains a stratified substring (a substring of a string $e_1 \ldots . e_m$ is any string of the form $e_i \ldots . e_j$ where $1 \leq i \leq j \leq m$).

Proposition 5. *If every simple cycle of the timed automaton $\mathcal{U}$ is either reset-free or stratifiable, the zone tree of $\mathcal{U}$ is locally finite.*

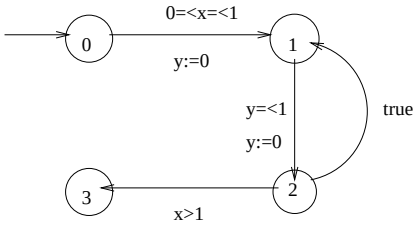

Fig. 2. Example of a timed automaton showing that the property: "Every reachable location is reachable through a simple path" does *not* entail termination of depth-first symbolic forward analysis.

Proof. Follows from Propositions 1, 2 and 4. ∎

We apply the above results to obtain our first sufficient termination condition.

Theorem 1. *Symbolic depth-first forward analysis of a timed automaton $\mathcal{U}$ terminates if all simple cycles of $\mathcal{U}$ are either reset-free or stratifiable.*

Proof. Follows from Propositions 3 and 5. ∎

4 RQ Automata

A timed automaton $\mathcal{U}$ is called RQ [LB93] if for each clock x, $\mathcal{U}$ contains exactly one edge with a reset of x and exactly one edge with a query of x, and moreover, for every transition sequence of $\widetilde{\mathcal{U}}$ starting from the initial position, the sequence of resets and queries of x is alternating, with a reset before the first query; here, $\widetilde{\mathcal{U}}$ refers to the timed automatonfrom $\mathcal{U}$ obtained by replacing all conjuncts $x \sim c$ in the guard formulas by the conjunct $x \geq 0$. We may require wlog. that no edge e of a timed automaton $\mathcal{U}$ contains both a reset of a clock and a query of a clock.

RQ automata have the following interesting property: if a location is reachable then it is reachable through a *simple path*, i.e. a sequence of edges that form a string not containing a cycle [LB93]. So it is possible to derive specialized terminating graph algorithms for reachability for RQ automata. Moreover, a cycle is traversable infinitely often if it is traversable once [LB93]. We will now investigate how a generic model checker based on symbolic forward analysis behaves on RQ automata. We do not know whether we obtain termination for this special case. We know that the distinguished property of RQ automata (that reachability is equivalent to reachability through a simple path) by itself is not sufficient for termination; Figure 2 gives a counterexample.

We will consider two special classes of RQ automata. The first one is characterized by the cut condition.

A timed automaton $\mathcal{U}$ satisfies the *cut condition* if any two simple cycles w and w' are either identical or their sets of edges are disjoint. Graph-theoretically,

every simple cycle in the control graph has exactly one entry point (which is then called the 'cut vertex').

Theorem 2. *Symbolic depth-first forward analysis of an RQ timed automaton $\mathcal{U}$ terminates if it satisfies the cut condition and in every simple cycle, either all or no clock is reset.*

Proof. A simple cycle containing a reset for each clock in an RQ automaton satisfying the cut condition is stratified. Hence, Theorem 1 yields the statement. ∎

The second class of RQ automata is obtained by restricting the number of clocks to two.

Theorem 3. *Symbolic depth-first forward analysis of an RQ timed automaton with two clocks terminates.*

Proof. We name the two clock variables of the automaton x and y. We note R_x the unique edge of the time automaton where x is reset, and Q_y the one where x is queried; similarly we define R_y and Q_y. By our non-proper restriction, $R_x \neq Q_x$ etc..

A *segment* S of a path p in a zone tree is a sequence of nodes $n_1, \ldots, n_m$ of the zone tree. The string $w = e_1 \ldots e_{m-1}$ *labels* the segment S if n_m is reached from n_1 by following the edges $e_1, \ldots, e_m$ in the zone tree.

For a proof by contradiction, assume that p is an infinite branch of the zone tree. By Proposition 4, there exists a simple cycle w (leading, say, from the location ℓ to ℓ) that repeats infinitely often on p. We write S_1, S_2, ... for the segments that are labeled by w (in consecutive order). We write L_i for the segment between S_i and S_{i+1}. We note v^i the string labeling the segment L_i; each string v^i is a cycle (leading also from the location ℓ to ℓ). Below we will use the terminology 'w labels S_i' and 'v^i labels L_i'.

We first distinguish between the cases whether the edge R_x is part of the string w ("$R_x \in w$") or not.

Case 1 $R_x \in w$.

The edge Q_x must then also be an element of w (if the cycle w can be executed once then even infinitely often [LB93]; if it contained R_x but not Q_x then the RQ condition would be violated).

Case 1.1 $R_y \in w$.

Again, we must have that $Q_y \in w$.

We distinguish between the cases that the edge R_y appears strictly before the edge Q_y in the strings w ("$R_y < Q_y$") or after ("$Q_y < R_y$").

Case 1.1.1 $R_y < Q_y$.

Repeating the above reasoning for x instead of y, we distinguish between the cases "$R_y < Q_y$" and "$Q_y < R_y$".

Case 1.1.1.1 $R_x < Q_x$.

The two assumptions $R_x < Q_x$ and $R_y < Q_y$ mean that the string w is *stratified*. Hence, by Proposition 1, the successor constraint function wrt. w is constant. Hence, the constraint labeling the last node of S_2 entails the constraint labeling

the last node of S_1. Thus, the path p is locally finite, which achieves the contradiction.

Case 1.1.1.2 $Q_x < R_x$.

We distinguish the cases whether the edge Q_x appears before the edge R_y or strictly after.

Case 1.1.1.2.1 $Q_x < R_y$.

Combining the assumptions leading to this case, namely $R_x \in w$ (and hence also $Q_x \in w$) and $R_y \in w$ (and hence also $Q_y \in w$) and $R_y < Q_y$ and $Q_x < R_x$ and $Q_x < R_y$, we know that the string w is of the form $w = w_1.Q_x.w_2$ such that w_2 contains R_x and R_y. Hence, the substring w_2 of w *stratified*. By Proposition 1, the successor constraint function wrt. w_2 is constant, and hence also the one wrt. w. As in the case above, we achieve a contradiction.

Case 1.1.1.2.2 $R_y < Q_x$.

Again we combine the assumptions leading to this case: namely $R_x, Q_x, R_y, Q_y \in w$ and $R_y < Q_y$ and $Q_x < R_x$ and $R_y < Q_x$.

Only using that $R_y < R_x$, we know that the string w is of the form $w = w_1.R_y.w_2.R_x.w_3$.

One of the two cases, namely $R_x \notin L_i$ or $R_x \in L_i$, will hold for infinitely many segments L_i's.

Case 1.1.1.2.2.1 $R_x \notin L_i$.

Then also $Q_x \notin L_i$ (because of the RQ-condition and since L_i is a cycle).

We then distinguish between the analogue cases for y instead of x.

Case 1.1.1.2.2.1.1 $R_y \notin L_i$.

Again, then $Q_y \notin L_i$.

We are assuming that $R_x, Q_x, R_y, Q_y \notin L_i$ for infinitely many L_i. We take two such segments, calling them L and L'. Let v and v' be the string labeling (the edge linking the nodes in) L and L'. Then, the successor constraint functions wrt. v and v' are the identity.

We form the *stratified* strings $V = R_x.w_3.v.w_1.R_y$ and $V' = R_x.w_3.v'.w_1.R_y$. Since the successor constraint functions wrt. v and v' are the identity, the successor constraint functions wrt. V and V' are the same *constant* function. The same reasoning as above leads to a contradiction.

Case 1.1.1.2.2.1.2 $R_y \in L_i$.

Then also $Q_y \in L_i$. Because of the RQ-condition and since the edge R_y precedes Q_y in S_i, the first occurrence of R_y precedes the first occurrence of Q_y in L_i. Hence, the strings v and v' (defined as above, labeling of some L_i's) is of the form $v = v_1.R_y.v_2$ or $v = v'_1.R_y.v'_2$ where v_1, v_2, v'_1 and v'_2 do not contain any reset or any query of a clock variable (and hence, yield the identity as the successor constraint function). We form the *stratified* substrings $V = R_x.w_3.v_1.R_y$ and $V' = R_x.w_3.v'_1.R_y$, which yield the same constant successor constraint function for the same reason as above. Again, this leads to a contradiction.

Case 1.1.1.2.2.2 $R_x \in L_i$.

Again, then $Q_x \in L_i$. Now we are assuming that $R_x, Q_x, R_y, Q_y \in L_i$ for infinitely many L_i.

As in Case 1.1.1.2.2.1.2, the first occurrence of R_y must precede the first occur-

rence of Q_y in L_i.
Assume that there is a reset of x in L_i before the first reset of y. We form the string $R_x.w_2.v_1, R_x$ where $w = w_1.R_x.w_2$ is such that w_2 does not contain any reset (by the assumptions for the cases 1.1.1.2 and 1.1.1.2.2) and $v^i = v_1.R_x.v_2$ (the string labeling L_i) is such that v_1 does not contain any reset. Following the lines of the proof for Proposition 2 one can show that for any constraint φ, $[\![R_x.w_2.v_1.R_x]\!](\varphi)$ entails $[\![R_x]\!](\varphi)$. This is a contradiction (to the fact that the path p is locally infinite).
Assume that there is no reset of x in L_i before the first reset of y. Then the string formed by the edges leading from the reset of x in S_i to the first reset of y in L_i is stratified. We can then apply the same reasoning as in Case 1.1.1.2.1 to derive a contradiction.
Case 1.1.2 $Q_y < R_y$.
Thus now $R_x \in w$ (and hence $Q_x \in w$), $R_y \in w$ (and hence $Q_y \in w$) and $Q_y < R_y$. Now we consider the following subcases of this case.
Case 1.1.2.1 $R_x < Q_x$
This case is symmetric to Case 1.1.1.2.1 where $R_x, R_y \in w$, $Q_x < R_y$ and $R_y < Q_y$.
Case 1.1.2.2 $Q_x < R_x$.
The assumption of the case is that the reset occurs after the query for both clocks. Due to the RQ condition, there cannot be any query between the two resets. Therefore, $R_x.w_1.R_y$ (or, symmetrically, $R_y.w_1.R_x$) forms a *stratified* substring of w. As before, we obtain a contradiction.
We refer to the full version of this paper [MP99] for the remaining cases. ‖

5 Future Work

The presented work targets theoretical investigations of timed automata not at the verification problem itself but, instead, at the termination behavior of the procedure solving it in practice, namely symbolic forward analysis. This work is a potential starting point for deriving interesting sufficient termination conditions. There are, however, other open questions along these lines.

Our setup may also be used to derive *necessary* termination conditions. These are useful obviously in the cases when their test is negative. Another question is whether there exist decidable necessary and sufficient conditions.

We may also consider logical equivalence instead of local subsumption for a practically more efficient, but theoretically weaker fixpoint test (used in tools such as Uppaal [LPY95]). We observe that Proposition 1 is still directly applicable in the new context, but Proposition 2 is not. The comparison of the different fixpoint tests (equivalence, local and global subsumption) is an interesting subject of research.

We may be able to derive natural and less restrictive sufficient termination conditions when we consider the enhancement of symbolic forward analysis with techniques from [Boi98] to compute the effect of loops, i.e. essentially the constraint transformer $[\![w^\omega]\!]$ for simple cycles w.

The constraint transformers $[\![w]\!]$ form a 'symbolic version' of the *syntactic monoid* [Eil76] for timed automata. This notion may be of intrinsic interest and deserve further study.

Acknowledgement. We thank Tom Henzinger and Jean-Francois Raskin for discussions.

References

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

[Bal96] F. Balarin. Approximate reachability analysis of timed automata. In *Proceedings of 17th IEEE Real-Time Systems Symposium*, pages 52–61. IEEE Computer Society Press, 1996.

[Boi98] B. Boigelot. *Symbolic Methods for Exploring Infinite State Spaces.* PhD thesis, Université de Liège, 1998.

[DP99] G. Delzanno and A. Podelski. Model checking in CLP. In Rance Cleaveland, editor, *Proceedings of TACAS'99, the Second International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1579 of *Springer LNCS*. Springer-Verlag, 1999.

[DT98] C. Daws and S. Tripakis. Model checking of real-time reachability properties using abstractions. In B. Steffen, editor, *Proceedings of the 4th International Conference on Tools and Algorithms for the Construction of Systems*, LNCS 1384, pages 313–329. Springer-Verlag, 1998.

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

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

[HKQ98] T. A. Henzinger, O. Kupferman, and S. Qadeer. From pre-historic to post-modern symbolic model checking. In *Proceedings of the International Conference on Computer-Aided Verification*, pages 195–206. Springer, 1998.

[HNSY94] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994. Special issue for LICS 92.

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

[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.

[MP99] S. Mukhopadhyay and A. Podelski. Beyond region graphs: Symbolic forward analysis of timed automata, 1999. Full Version. Available at http://www.mpi-sb.mpg.de/∼podelski.

[MS98] K. Marriott and P. J. Stuckey. *Programming with Constraints: An Introduction.* MIT Press, 1998.