Symbolic Robustness Analysis of Timed Automata

Conrado Daws[1,2,⋆] and Piotr Kordy[2]

[1] Department of Applied Mathematics
[2] Formal Methods Group
Faculty of Electrical Engineering, Mathematics and Computer Science,
University of Twente, The Netherlands
conrado.daws, piotr.kordy at ewi.utwente.nl

Abstract. We propose a symbolic algorithm for the analysis of the robustness of timed automata, that is the correctness of the model in presence of small drifts on the clocks or imprecision in testing guards. This problem is known to be decidable with an algorithm based on detecting strongly connected components on the region graph, which, for complexity reasons, is not effective in practice.

Our symbolic algorithm is based on the standard algorithm for symbolic reachability analysis using zones to represent symbolic states and can then be easily integrated within tools for the verification of timed automata models. It relies on the computation of the stable zone of each cycle in a timed automaton. The stable zone is the largest set of states that can reach and be reached from itself through the cycle. To compute the robust reachable set, each stable zone that intersects the set of explored states has to be added to the set of states to be explored.

1 Introduction

Timed automata [2] are an important formal model for the specification and analysis of real-time systems. They are a simple extension of automata with real-valued variables, called clocks, whose values increase at the same rate in the control locations, and can be reset to 0 when a discrete transition is taken. By adding a certain type of constraint on clocks to the locations and edges of the automaton, one can respectively specify the time a system is allowed to remain in a control location, and when a discrete transition can be taken. Many real-time systems have been modeled using timed automata and analyzed automatically with tools like UPPAAL [10] and KRONOS [6].

A fundamental form of system analysis is the verification of safety properties, which consists in checking whether any unsafe state is reachable. This kind of analysis is performed efficiently by the tools mentioned above with well known algorithms manipulating timed constraints, called *zones*, that can be represented as a square matrix of difference bounds (DBM). The reachability analysis is based on the *idealized* semantic assumption that all clocks advance with the same speed. However, in a real implementation of a system, clocks will be likely to drift and measure time only up to some precision.

Puri first addressed this concern in [12] where he considered drifting clocks and showed that timed automata models are not robust with respect to safety properties,

⋆ This work was originally carried out at the Nijmegen Institute for Computing and Information Sciences, Radboud University Nijmegen, and supported by the European Community Project IST-2001-35304 AMETIST meaning that a model proven to be safe under the standard ideal semantics might not be safe even if clocks drift by an arbitrarily small amount. De Wulf et al. consider a semantics, called the almost ASAP semantics [8], capturing certain notion of clock imprecision that can be translated into a syntactical enlargement of the guards. They later showed in [7] that the implementability of a model under their semantics can be decided with Puri's algorithm for robustness analysis.

Both results rely on an enlarged semantics of timed automata with either drift or imprecision of clocks. They consider the set of states that are reachable for *any* drift or imprecision. If this set contains some unsafe state, then the model is considered not to be robust or implementable. The robust reachability set can be computed with the algorithm from [12] in both cases, which are thus equivalent. The algorithm is based on the structure of the limit cycles of a timed automaton, i.e. the cyclic trajectories in the underlying timed transition system. The algorithm considers the strongly connected components of the region graph because they contain the limit cycles of the timed automaton. It adds every strongly connected component that intersects the reachability set, and its successors, to the reachability set. However, because the size of the region graph is exponential in the number of clock variables and the largest constant in the constraints, the algorithm is not effective in practice.

We propose a symbolic algorithm for computing the enlarged reachability set of a timed automaton based on the standard algorithm for symbolic reachability analysis using zones to represent symbolic states. Our algorithm relies on the computation of a *stable zone* W_σ of every progress cycle σ in the timed automaton, defined as the maximal set of clock values that have successors and predecessors through any number of iterations of σ in the timed automaton. That is, $W_\sigma = \bigcap_{i\geq 0} \mathrm{post}^i_\sigma(\mathsf{True}) \cap \bigcap_{i\geq 0} \mathrm{pre}^i_\sigma(\mathsf{True})$. The stable zone has the property of reaching and being reached from any cycle in the region graph, and hence any states in a limit cycle. We modify the standard reachability algorithm such that whenever the stable zone of a cycle intersects the standard reachable set, the whole stable zone is added to the set of states to be explored.

Related work. The robustness analysis has been extended to more general type of properties, like Büchi and LTL in [4]. Other notions of robustness have been considered in the literature, like [9,11] which impose a restriction to the type of accepted traces, as opposed to the enlargement we consider here. A different modelling based approach to implementability can be found in [1].

The remaining of the paper is organized as follows. Section 2 recalls the basic standard definitions of timed automata. The robustness problem arising from an enlarged semantics is presented in Section 3. Our contribution is the subject of Section 4, where we define the stable zone of a cycle and study its main properties, which we then use in our symbolic algorithm for robustness analysis. Finally, Section 5 concludes our presentation with a summary of the main results and a discussion on future work.

2 Timed Automata

This section briefly recalls the definitions of timed automata, their semantics, reachability analysis, and region graph.

2.1 Definitions

Definition 1 (Closed Zones). *A* closed zone *over the set of clocks $\mathcal{C}$ is a conjunction of simple constraints giving a positive lower and upper bound to the value of each clock, and a lower and upper bound to the difference between any pair of clocks. Formally,*

$$\mathcal{Z}(\mathcal{C}) = \bigwedge_{x \in \mathcal{C}} l_x \leq x \leq u_x \wedge \bigwedge_{x,y \in \mathcal{C}} x - y \leq d_{xy}$$

with $l_x, u_x \in \mathbb{N} \cup \{\infty\}$ and $d_{xy} \in \mathbb{Z} \cup \{-\infty, \infty\}$.

Zones without bounds for clock differences are called *rectangular* and denoted $\mathcal{Z}_R(\mathcal{C})$. Rectangular zones without lower bounds are called *upper* zones, and denoted $\mathcal{Z}_U(\mathcal{C})$. A *clock valuation* is a function v mapping $\mathcal{C}$ to non-negative real numbers. True denotes the zone $\bigwedge_{x \in \mathcal{C}} 0 \leq x$ satisfied by any valuation of the clocks. By $v \models z$ we will understand that if v is a clock valuation and z is a zone, then the clock values denoted by v satisfy the constraints of z.

Definition 2 (Timed Automaton). *A* timed automaton *is a tuple* $\mathsf{A} = \langle \mathcal{L}, \mathcal{C}, \mathcal{I}, \mathcal{T} \rangle$ *where*

1. *$\mathcal{L}$ is a finite set of locations representing the discrete control structure of the system*
2. *$\mathcal{C}$ is a finite set of nonnegative real-valued variables called clocks*
3. *$\mathcal{I} : \mathcal{L} \to \mathcal{Z}_U(\mathcal{C})$ are the location invariants*
4. *$\mathcal{T} \subseteq \mathcal{L} \times \mathcal{Z}_R(\mathcal{C}) \times 2^{\mathcal{C}} \times \mathcal{L}$ is a sef of edges. An edge (l, g, R, l') represents a transition from l to l' with a guard g and a set R of clocks to reset.*

2.2 Semantics

Definition 3 (Standard semantics). *Given a timed automaton* $\mathsf{A} = \langle \mathcal{L}, \mathcal{C}, \mathcal{I}, \mathcal{T} \rangle$ *its semantics is defined as the timed transition system* $[\![\mathsf{A}]\!] = (Q, \to_t \cup \to_e)$ *such that:*

- *$Q \subseteq \mathcal{L} \times \mathbb{R}^n_+$: $(l, v) \in Q$ iff $v \models \mathcal{I}(l)$*
- *$(l, v) \to_t (l, v + t)$ if $t \in \mathbb{R}_{\geq 0}$ and $v + t \models \mathcal{I}(l)$*
- *$(l, v) \to_e (l', v')$ if $e = (l, g, R, l') \in \mathcal{T}$ such that:*
  - *$v \models g$ and $v' \models I(l')$*
  - *$v'(x) = 0$ if $x \in R$, $v'(x) = v(x)$ otherwise.*

In the remaining of the paper we will use the following notations. Let $e \in \mathcal{T}$ be an edge of A, then $x \overset{e}{\Longrightarrow} y$ if and only if there exists $z, z' \in Q$ such that $x \to_t z \to_e z' \to_t y$. Let $\pi = e_1 e_2 \ldots e_n$ be a sequence of edges, then $x \overset{\pi}{\Longrightarrow} y$, meaning that there is a trajectory from x to y through π, if and only if there exist $z_1 \ldots z_{n-1} \in Q$ such that $x \overset{e_1}{\Longrightarrow} z_1 \ldots z_{n-1} \overset{e_n}{\Longrightarrow} y$.

The following lemma states that if we follow the same sequence of edges reaching two states then any linear combination of the states is also reachable. Lemma 1 follows from the convexity of the guards.

Lemma 1. *Let $\pi = e_1 e_2 \ldots e_n$ be be a sequence of edges. If $x \overset{\pi}{\Longrightarrow} x'$ and $y \overset{\pi}{\Longrightarrow} y'$, then for all $\lambda \in [0, 1]$, $\lambda x + (1 - \lambda) y \overset{\pi}{\Longrightarrow} \lambda x' + (1 - \lambda) y'$.*

2.3 Reachability Analysis

The most fundamental form of analysis of a timed automaton is the computation of its reachable state space from an initial state. The reachable state space of A from $q_0 \in Q$ under semantics$[\![\mathsf{A}]\!]$, denoted $\mathsf{Reach}([\![\mathsf{A}]\!], q_0)$, is the set of states $q \in Q$ such that $(q_0, q) \in (\rightarrow_t \cup \rightarrow_e)^\star$. We denote $y \in \mathsf{Reach}([\![\mathsf{A}]\!], x)$ by $x \Longrightarrow y$.

Safety properties can then be verified by checking if an undesired set of states Bad is reachable, i.e. if $\mathsf{Reach}([\![\mathsf{A}]\!], q_0) \cap \mathsf{Bad} = \emptyset$, which can be decided even when it is not possible to compute the reachable state space exactly. An interesting question first tackled by Puri in [12] is how robust a timed automaton model is if we relax the assumption that all clocks advance at the same speed. He showed that in some cases the verification results do not hold even for small drifts.

2.4 Region Graph

Given a timed automaton A, let k be a function, called a clock ceiling, mapping each clock $x \in \mathcal{C}$ to $k(x)$ - the largest integer c such that $(x \leq c)$ or $(c \leq x)$ is a subformula for some clock constraint appearing in A. We assume that every clock $x \in \mathcal{C}$ appears in some constraint. For a real number d, let $\langle d \rangle$ denote the fractional part of d, and $\lfloor d \rfloor$ denote its integral part. So $d = \lfloor d \rfloor + \langle d \rangle$.

Definition 4 (Clock regions). *A* clock region *is an equivalence class of the relation $\sim_k$. The equivalence relation $\sim_k$ is defined over the set of clock valuations. Two clock valuations are region equivalent denoted $v \sim_k v'$ iff all following conditions hold:*

1. *for all $x \in \mathcal{C}$ either $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ or $v(x) > k(x)$ and $v'(x) > k(x)$.*
2. *for all $x, y \in \mathcal{C}$ if $v(x) \leq k(x)$ and $v(y) \leq k(y)$ then $\langle v(x) \rangle \leq \langle v(y) \rangle$ iff $\langle v'(x) \rangle \leq \langle v'(y) \rangle$.*
3. *for all $x \in \mathcal{C}$ if $v(x) \leq k(x)$ then $\langle v(x) \rangle = 0$ iff $\langle v'(x) \rangle = 0$.*

We denote $[v]$ the smallest closure of the set of clock assignments region-equivalent to v. Such set is called a *closed region.*

Definition 5 (Region graph). *Given the timed transition system $[\![\mathsf{A}]\!] = (Q, \rightarrow)$ of a timed automaton A we define the corresponding (closed)* region graph *$G = (\mathsf{R}, \rightarrow_G)$ of A:*

- $\mathsf{R} = \{(l, [v]) \mid (l, v) \in Q\}$ *is a set of closed regions.*
- $\rightarrow_G \subseteq \mathsf{R} \times \mathsf{R}$*: $((l, [v]), (l', [v'])) \in \rightarrow_G$ if $(l, [v]) \neq (l', [v'])$ and either $(l, v) \rightarrow_t (l', v')$ or $(l, v) \rightarrow_e (l', v')$.*

If $q = (l, v)$ then by $[q]$ we understand closed region containing q, i.e. $[q] = (l, [v])$

3 Robustness Problem

In this section we consider the robustness or implementability of timed automata as studied in [12] and [7]. We first define a family of enlarged semantics parameterized by the drift in clocks ϵ and the imprecision in guards Δ. Following [12] we require that every cycle is a progress cycle, i.e. a cycle where each clock is reset at least once.

3.1 Enlarged Semantics

Given a timed automaton $\mathsf{A} = \langle \mathcal{L}, \mathcal{C}, \mathcal{I}, \mathcal{T} \rangle$ its enlarged semantics parameterized by $\epsilon, \Delta \in [0,1)$ is defined as the timed transition system $[\![\mathsf{A}]\!]^{\epsilon}_{\Delta} = (Q, \rightarrow_t \cup \rightarrow_e)$ such that:

Definition 6 (Enlarged semantics)

- $Q \subseteq \mathcal{L} \times \mathbb{R}^n_+$*:* $(l, v) \in Q$ *iff* $v \models \mathcal{I}(l)$
- $(l, v) \rightarrow_t (l, v')$ *if* $v' \models \mathcal{I}(l')$ *and* $v'(x_i) - v(x_i) \in [(1-\epsilon)t, (1+\epsilon)t]$ *for* $i = 1, \ldots, n$
- $(l, v) \rightarrow_e (l', v')$ *if* $e = (l, g, R, l') \in \mathcal{T}$ *such that:*
  - $v \models g_{\Delta}$[1] *and* $v' \models \mathcal{I}(l')$
  - $v'(x) = 0$ *if* $x \in R$*, otherwise* $v(x)$

Clearly, the enlarged semantics $[\![\mathsf{A}]\!]^{\epsilon}_{\Delta}$ has more reachable states than the standard semantics $[\![\mathsf{A}]\!] = [\![\mathsf{A}]\!]^0_0$. We are not interested in the reachable states for some particular ϵ or Δ, but in those states that are reachable for any ϵ or any Δ but are not reachable in the standard semantics.

Definition 7 (Robust reachability)
The robust set of states reachable from $q_0 \in Q$ *with some clock drift* R^*_{ϵ} *or guard imprecision* R^*_{Δ} *are given by*

$$R^*_{\epsilon}(\mathsf{A}, q_0) = \bigcap_{\epsilon > 0} \mathsf{Reach}([\![\mathsf{A}]\!]^{\epsilon}_0, q_0) \qquad R^*_{\Delta}(\mathsf{A}, q_0) = \bigcap_{\Delta > 0} \mathsf{Reach}([\![\mathsf{A}]\!]^0_{\Delta}, q_0).$$

The question is whether unsafe states Bad can be unreachable under the standard semantics but reachable under the robust semantics, i.e. $\mathsf{Reach}([\![\mathsf{A}]\!], q_0) \cap \mathsf{Bad} = \emptyset$ and $R^*_{\epsilon}(\mathsf{A}, q_0) \cap \mathsf{Bad} \neq \emptyset$. The following example, from [12], shows that this is indeed possible.

3.2 Example

We consider the timed automaton in Figure 1 and the initial state $(\mathsf{L1}, a = 1 \wedge b = 0)$. The parameter K is an integer taking the value 2 or 3. We want to check if the model is not safe, i.e. if location Err is reachable, which is only possible if the value of b can be larger or equal than K when entering location $\mathsf{L2}$.

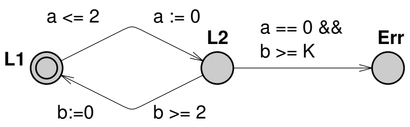

Fig. 1. A (robust?) timed automaton

Clearly, this is not possible under the standard semantics, regardless of the value of K. As can be seen from the reachable state space depicted in dark gray in Figure 2,

[1] g_{Δ} is the rectangular guard g extended by Δ, i.e. replacing $a \leq x \leq b$ by $a - \Delta \leq x \leq b + \Delta$.

upon entering location L2 we must have $a = 0$ and $b \leq 1$. However, lets consider the case of clock b advancing at speed $1 + \epsilon$ in location L1, and 1 in location L2, whilst a advances with speed 1 in both locations. Then there is a trajectory such that upon entering location L2 for the k-th time, $a = 0$ and $b = (1 + \epsilon)^k$. Moreover, because $a \geq 0$ in L1, we have that $b \leq 2(1 + \epsilon)/(1 - \epsilon)$ when entering L2. So for any ϵ and sufficiently many iterations, location Err becomes reachable if $K = 2$, but not if $K = 3$ and $\epsilon < 1/5$.

So, although the model for $K = 2$ is considered safe under the standard idealized semantics, it is not under the robust semantics because the unsafe location will be reachable for *any* drift or imprecision in the measurement of clocks. In other words, the model is not robust or implementable. On the other hand, the same model is robust or implementable for $K = 3$ provided that ϵ is small enough.

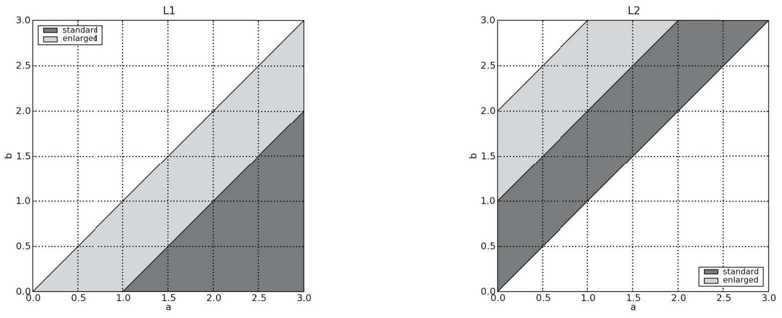

Fig. 2. Reachable state space from $(\mathsf{L1}, a = 1 \wedge b = 0)$ with the standard (dark) and enlarged (light) semantics

3.3 Algorithm

Algorithm 1 is a slightly modified version of the algorithm in [12] to compute iteratively the set J^* on the region graph of a timed automaton A. We first compute the reachable regions from a given initial state $[q_0]$. Then, for every region s in the strongly connected components of the region graph that intersects the current J^*, the regions reachable from s are added to J^*.

Theorem 1 (from [12] and [7]). *Algorithm 1 computes the robust reachable state space of* A *from* r_0 *under the enlarged semantics, with either drifting clocks [12] or enlarged guards [7].*

$$J^* = R^*_\epsilon(\mathsf{A}, r_0) = R^*_\Delta(\mathsf{A}, r_0)$$

The proof of this theorem (shown in [12] and [7]) relies on the structure of the limit cycles of the timed automaton.

4 Symbolic Robustness Analysis

The robust semantics of a timed automaton depends on the structure of its limit cycles. Algorithm 1 computes the robust reachable state space by adding the reachable cycles

Algorithm 1. Algorithm computing $R^*_\epsilon(\mathsf{A}, r_0)$

```
Input: A timed automaton $\mathsf{A}$ and initial region $r_0$
Output: The set $J^* = R^*_\epsilon(\mathsf{A}, r_0)$
ENLARGEDREACH($\mathsf{A}, r_0$)
(1)     Construct the region graph $G$ of timed automaton $\mathsf{A}$
(2)     $S \leftarrow$ STRONGLYCONNECTEDCOMPONENTS($G$)
(3)     $J^* \leftarrow$ REACH($G, r_0$)
(4)     while there exists $s \in S$ such that $s \not\subseteq J^*$ and $s \cap J^* \neq \emptyset$
(5)         $J^* \leftarrow J^* \cup$ REACH($G, s$)
(6)         $S \leftarrow S \setminus s$
(7)     return $J^*$
```

in the region graph (which contain the limit cycles). But this algorithm is not effective in practice because of the size of the region graph.

We propose a symbolic algorithm to compute the enlarged reachable state space using zones to represent symbolic states. For that, we consider in this section a cycle $\sigma = e_1 \dots e_n$ of a timed automaton, with $e_i = (l_i, g_i, r_i, l'_i)$, as a sequence of edges such that $l'_i = l_{i+1}$ for $i = 1 \dots n-1$ and $l_1 = l'_n$. We assume that the cycle is a progress cycle, meaning that each clock is reset at least once.

4.1 Limit Cycles

Limit cycles are cyclic trajectories in the underlying timed transition system of a timed automaton, without superfluous 0-time self loops. A state in a limit cycle can return to itself after one or more iterations of a cycle in the timed automaton.

Definition 8 (Return Map). *Let σ be a cycle in timed automaton A. The return map of a state q is the set of states reachable from q after one cycle σ*

$$R_\sigma(q) = \{q' \mid q \overset{\sigma}{\Longrightarrow} q'\}.$$

Definition 9 (Limit Cycles). *The set of states which can return back to themselves after $i > 0$ iterations of the cycle σ is $L^i_\sigma = \{q \mid q \in R^i_\sigma(q)\}$, and the set of states with limit cycles through σ is $L_\sigma = \bigcup_{i>0} L^i_\sigma$.*

Lemma 2 shows that the set of states in limit cycles is convex because the convex combination of two limit cycles is also a limit cycle. Let $\overset{\sigma^+}{\Longrightarrow}$ be the transitive closure of $\overset{\sigma}{\Longrightarrow}$.

Lemma 2. *Let σ be a cycle in A. If $x \overset{\sigma^+}{\Longrightarrow} x$ and $y \overset{\sigma^+}{\Longrightarrow} y$, then for all $\lambda \in [0, 1]$, $\lambda x + (1-\lambda)y \overset{\sigma^+}{\Longrightarrow} \lambda x + (1-\lambda)y$.*

Proof. If $x \overset{\sigma^+}{\Longrightarrow} x$ then there exists $m \geq 1$ such that $x \overset{\sigma^m}{\Longrightarrow} x$. Similarly, there exists $n \geq 1$ such that $y \overset{\sigma^n}{\Longrightarrow} y$. So $x \overset{\sigma^{mn}}{\Longrightarrow} x$ and $y \overset{\sigma^{mn}}{\Longrightarrow} y$. Therefore, from Lemma 1, for all $\lambda \in [0, 1]$ $\lambda x + (1-\lambda)y \overset{\sigma^{mn}}{\Longrightarrow} \lambda x + (1-\lambda)y$.

Several properties of limit cycles of timed automata are given in [12]. For instance, the set of states with limit cycles in a cycle of the region automaton is a non-empty region. It follows, by convexity of the limit cycles, that L_σ is a zone. An important property of limit cycles is that for any state in a cycle of the region graph, there is a state in the limit cycles of its region that can reach it, and a state that can be reached from it.

Lemma 3 (Lemma 11 in [12]). *Consider a cycle $c = c_0 c_1 \dots c_N$ in the region graph. Then, for any $x \in c_0$, there exist $u, v \in L_c$ such that $x \Longrightarrow u$ and $v \Longrightarrow x$.*

4.2 Region Graph Cycles

Definition 10 (Regions with Cycles). *Let σ be a cycle in timed automaton A. We define the set of regions with cycles through σ in the region graph of A as $C_\sigma = \{r \in \mathsf{R} \,|\, \exists q, q'.\, r = [q] = [q'] \text{ and } q \overset{\sigma^+}{\Longrightarrow} q'\}$.*

Lemma 4 says that for any two cycles in the region graph following a cycle in the timed automaton, we can find a sequence of regions touching each other and each of these regions is in a cycle of the region graph.

Lemma 4. *Given a cycle σ in the timed automaton, for any two regions $c, c' \in C_\sigma$ there exists a finite sequence of regions $c_0 \dots c_n \in C_\sigma$ such that for all $0 \leq i < n$, $c_i \cap c_{i+1} \neq \emptyset$, $c_0 \cap c \neq \emptyset$ and $c_n \cap c' \neq \emptyset$.*

Proof. We know that each region in a cycle of the region graph contains a state with a limit cycle. Let $x \in c$ and $y \in c'$ have limit cycles. Thus, state $\lambda x + (1-\lambda)y$ is in a limit cycle and its region in a cycle in the region graph, for any $\lambda \in [0,1]$, which implies the existence of regions with the required property.

Remark 1. The set of states in C_σ is not necessarily convex, as illustrated in the example in Section 4.5.

4.3 Stable Zone

Let $X \subseteq Q$ be a set of states and σ a cycle of A. We denote $\mathrm{pre}_\sigma(X) = \{q \,|\, \exists q' \in X. q \overset{\sigma}{\Longrightarrow} q'\}$ the set of predecessors of X through σ, and $\mathrm{post}_\sigma(X) = \{q \,|\, \exists q' \in X. q' \overset{\sigma}{\Longrightarrow} q\}$) the set of successors of X through σ.

Definition 11 (Stable Zone). *The stable zone of a cycle σ in a timed automaton is the zone*

$$W_\sigma = \nu X.\, \mathrm{post}_\sigma(X) \cap \nu X.\, \mathrm{pre}_\sigma(X)$$

The stable zone characterizes those states that have a predecessor and a successor after any number of σ iterations. In the finite lattice of zones defined using constants smaller than the largest constant appearing in the timed automaton, the greatest fixed points can be computed by iteration from True. Hence the σ-stable zone can be computed as

$$W_\sigma = \bigcap_{i \geq 0} \mathrm{post}^i_\sigma(\mathsf{True}) \cap \bigcap_{i \geq 0} \mathrm{pre}^i_\sigma(\mathsf{True})$$

Example 1. The stable zone of the cycle σ =L1-L2-L1 in the timed automaton in Figure 1 is:

$$\begin{aligned} W_\sigma &= (0 \le a \le 2 \wedge 0 \le b \le 3 \wedge -2 \le b - a \le 3) \cap \\ &\quad (0 \le a \le 3 \wedge 0 \le b \le 3 \wedge -3 \le b - a \le 0) \\ &= (0 \le a \le 2 \wedge 0 \le b \le 2 \wedge -2 \le b - a \le 0) \end{aligned}$$

where the fixpoints are computed in 2 iterations. It can be checked that those are also the states with a limit cycle through σ. However this is not always the case.

Moreover, for any state in W_σ there exist a successor and a predecessor through any number of iterations of σ that also belong to W_σ. This property follows from the following lemma that characterizes the stable zone as the maximal set of states that can reach and be reached from itself.

Lemma 5. $W_\sigma = \nu X.(\mathrm{post}_\sigma(X) \cap \mathrm{pre}_\sigma(X))$

Proof. Let $Y_1 = \nu X.\, \mathrm{post}_\sigma(X)$, $Y_2 = \nu X.\, \mathrm{pre}_\sigma(X)$ and $Z = \nu X.(\mathrm{post}_\sigma(X) \cap \mathrm{pre}_\sigma(X))$. $Z \subseteq \mathrm{post}_\sigma(Z)$, therefore $Z \subseteq Y_1$. Similarly, $Z \subseteq Y_2$. Hence $Z \subseteq W_\sigma$. On the other hand, for all $y \in Y_1 \cap Y_2$, there exist $y_1 \in Y_1$ and $y_2 \in Y_2$ such that $y_1 \overset{\sigma}{\Longrightarrow} y \overset{\sigma}{\Longrightarrow} y_2$. Since $y \in Y_2$, y has suctcessors through any number of σ iterations, and so does y_1, hence $y_1 \in Y_2$ and $y \in \mathrm{post}_\sigma(Y_1 \cap Y_2)$. Similarly, $y_2 \in Y_1$ and $y \in \mathrm{pre}_\sigma(Y_1 \cap Y_2)$. So $Y_1 \cap Y_2 \subseteq \mathrm{post}_\sigma(Y_1 \cap Y_2) \cap \mathrm{pre}_\sigma(Y_1 \cap Y_2)$, hence $W_\sigma \subseteq Z$.

The following lemma states that the stable zone of a cycle σ of a timed automaton contains the cycles in the region graph through σ, which we know from [12] contain the limit cycles in σ. Moreover, the inclusions can be strict. An example where the first inclusion is strict is given in [13], and an example showing that both inclusions can be strict can be found in Section 4.5. We will abuse notation and consider C_σ as the set of states in the regions of C_σ.

Lemma 6. $L_\sigma \subseteq C_\sigma \subseteq W_\sigma$

Proof. Let $c \in C_\sigma$ be a region.Then, for any $q \in c$ there exist $q_1, q_2 \in c$ such that $q_1 \overset{\sigma^+}{\Longrightarrow} q \overset{\sigma^+}{\Longrightarrow} q_2$ so q has successors and predecessors through any number of σ iterations. Hence, $q \in W_\sigma$ and $c \subseteq W_\sigma$.

An important property in Puri's theory is that even though not every state in a cycle in the region graph is itself in a limit cycle, it can always reach a state in a limit cycle, and be reached by a state in a limit cycle. In a similar way, not every state in the stable zone is in a cycle in the region graph, but it can always reach a state in a cycle in the regiont graph, and be reached by a state in a cycle in the region graph. Lemma 7 formalizes this property.

Lemma 7. *For all $q \in W_\sigma$, there exist q_1, q_2 such that $q_1 \overset{\sigma^+}{\Longrightarrow} q \overset{\sigma^+}{\Longrightarrow} q_2$ and $[q_1], [q_2] \in C_\sigma$.*

Proof. Let $q \in W_\sigma$ be a state in the stable zone of σ. From Lemma 5 there exist $x_1 \ldots x_n \in W_\sigma$ such that $x_n \xRightarrow{\sigma} \ldots x_1 \xRightarrow{\sigma} q$ for any $n \geq 1$. Since q can be reached in any number of iterations of σ, and the number of regions is finite, then, for n sufficiently large, there exist x_i, x_j such that $[x_i] = [x_j]$ and then we take $q_1 = x_i$. A similar reasoning shows the existence of q_2.

4.4 Symbolic Robustness Algorithm

Algorithm 1 computes the robust reachability set by adding iteratively to the reachable state space all regions reachable from a strongly connected component of the region graph, when some region in the SCC intersects the current reachable state space.

We propose an algorithm to compute symbolically the robust set of reachable states of a timed automaton. Algorithm 2 is based on the standard symbolic algorithm [5,3] to compute the reachable state space of a timed automaton as implemented in the real-time model checkers KRONOS and UPPAAL. A symbolic state is a pair $\langle l, z\rangle$ of a location l and a zone z. First the algorithm computes the stable zone of each cycle of the timed automaton. Each time a new symbolic state representing states not visited yet is chosen, its successors are added to the waiting list, until the latter is empty. The difference with the standard algorithm resides in lines 9–11 where we check if the zone of the currently visited symbolic state intersects any of the stable zones of a cycle starting in its location, in which case we add the stable zone and its time-successors to the waiting list.

Algorithm 2. Symbolic algorithm computing $R^*_\epsilon(\mathsf{A}, r_0)$

```
Input: A timed automaton $\mathsf{A}$ and initial region $r_0 \in R_{\mathsf{A}}$
Output: The symbolic state set Passed $= R^*_\epsilon(\mathsf{A}, r_0)$
SYMBENLARGEDREACH(A)
(1)   Compute $W_\sigma$ for every cycle $\sigma$ in A
(2)   Wait $= \{r_0\}$
(3)   Passed $= \emptyset$
(4)   while Wait $\neq \emptyset$
(5)       pop $\langle l, D\rangle$ from Wait
(6)       if $D \not\subseteq D'$ for all $\langle l', D'\rangle \in$ Passed
(7)           Passed $\leftarrow$ Passed $\cup \{\langle l, D\rangle\}$
(8)           foreach $\langle l', D'\rangle \in$ SUCC($\langle l, D\rangle$)
(9)               Wait $\leftarrow$ Wait $\cup \{\langle l', D'\rangle\}$
(10)          foreach $W_\sigma$ with $\sigma$ starting in $l$
(11)              if $D \cap W_\sigma \neq \emptyset$
(12)                  Wait $\leftarrow$ Wait $\cup$ {TIMESUCC($\langle l, W_\sigma\rangle$)}
(13)  return Passed
```

Theorem 2 states that Algorithm 2 computes the set R^*_ϵ. We prove this result by showing that this algorithm computes the same set of states J^* as Algorithm 1 from Puri.

Theorem 2. *Let* A *be a timed automaton,* $r_0 \in R_{\mathsf{A}}$ *an initial region, and* W^* *the set of states returned by Algorithm 2. Then* $W^* = R^*_\epsilon(\mathsf{A}, r_0)$.

Proof. Algorithm 1 adds cycles in the region graph to the reachable state space. From Lemma 6 we know that all cycles in the region graph are included in the stable zone and hence will also be added by Algorithm 2, so $J^* \subseteq W^*$.

On the other hand, if some W_σ is added in Algorithm 2, then $W_\sigma \cap D \neq \emptyset$ for some reachable $\langle l, D\rangle$. From Lemma 7 we know that there exists some $c \in C_\sigma$ that is reachable from $W_\sigma \cap D$. So $[c]$ will be added by Algorithm 1. From Lemma 4, the whole C_σ must be added. Finally, W_σ is also added because it is reachable from C_σ by Lemma 7. Hence, $W^* \subseteq J^*$.

4.5 Example with Strict Inclusions

The timed automaton in Figure 3 (left) shows that the inclusions $L_\sigma \subseteq C_\sigma \subseteq W_\sigma$ can be strict, and that the set C_σ is not necessarily convex (right).

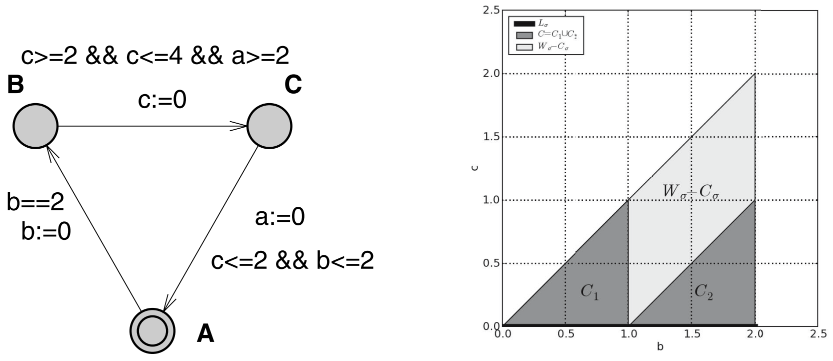

Fig. 3. Timed automaton (left) showing that $L_\sigma \subsetneq C_\sigma \subsetneq W_\sigma$ (right)

Strict inclusions and non convexity of C_σ. We only consider set of states when entering location A, that is, with $a = 0$ (the remaining reachable states can be reached from these states). The reader can check that the states with limit cycles L_σ, the states in cycles in the region graph C_σ, and the stable zone W_σ are given by the following equations, represented in Fig.3 (right):

$$\begin{aligned}
L_\sigma &= (A, a = 0 \wedge 0 \leq b \leq 2 \wedge c = 0)\\
C_1 &= (A, a = 0 \wedge 0 \leq b \leq 1 \wedge 0 \leq c \leq 1 \wedge 0 \leq b - c \leq 1)\\
C_2 &= (A, a = 0 \wedge 1 \leq b \leq 2 \wedge 0 \leq c \leq 1 \wedge 1 \leq b - c \leq 2)\\
C_\sigma &= C_1 \cup C_2\\
W_\sigma &= (A, a = 0 \wedge 0 \leq b \leq 2 \wedge 0 \leq c \leq 2 \wedge 0 \leq b - c \leq 2)
\end{aligned}$$

The fact that a set of states belongs or not to a cycle in the region graph can be observed computing the corresponding reachability set. In this case we have:

$$\begin{aligned}
\mathsf{Reach}(L_\sigma) &= W_\sigma & \mathsf{Reach}(C_1) &= W_\sigma\\
\mathsf{Reach}(W_\sigma) &= W_\sigma & \mathsf{Reach}(C_2) &= C_2\\
\mathsf{Reach}(C_\sigma) &= W_\sigma & \mathsf{Reach}(W_\sigma \setminus C_\sigma) &= C_2
\end{aligned}$$

Comparing both algorithms. Theorem 2 shows that both the original algorithm on the region graph, and our symbolic algorithm using the stable zone are equivalent. It is easy to see that any region added to the reachability set by Algorithm 1 will be also added by our algorithm because the stable zone contains all the cycles in the region graph. We illustrate with this example why the converse is also true.

Let's assume state $(A, (0, 2, 2))$ is reachable. Since it belongs to the stable zone, the whole stable zone will be added by our algorithm. Algorithm 1 will first only add the states reachable from $(A, (0, 2, 2))$, that is $(A, (0, 2, 0)) \in C_2$. So C_2 will be added next, then also C_1 because $C_1 \cap C_2 \neq \emptyset$ (a particular case of Lemma 4). Finally, $\mathsf{Reach}(C_1) = W_\sigma$, so the whole W_σ will be added.

5 Conclusions

We proposed a *symbolic algorithm* for computing the robust reachability set of a timed automaton based on the standard algorithm for symbolic reachability analysis of timed automata, using zones to represent symbolic states. Although the worst case complexity of symbolic algorithms is the same as for the region graph, symbolic algorithms are known to be more efficient in practice. Moreover, our symbolic algorithm is easy to implement and integrate within existing formal frameworks for the validation of real-time systems.

Our algorithm relies on the computation of the stable zone of each progress cycle in the timed automaton. The stable zone of a cycle is the maximal set of clock values that have a successor and a predecessor through any number of iterations of the cycle. All cycles in the region graph following the same cycle in the timed automaton are connected to each other through the limit cycles. Moreover, every point in the stable zone can reach a cycle in the region graph, and be reached from a cycle in the region graph. Based on these facts, we modified the standard reachability algorithm such that whenever the stable zone of a cycle intersects the set of computed reachable states, the stable zone is added to the set of states to be explored. We showed that our zone based algorithm is equivalent to the one from Puri operating on the region graph to compute the robust reachabilty set.

We implemented the computation of the stable zone of a cycle and applied it to the simple examples in [12] to validate our results. However, we need to implement the robust reachability algorithm in order to assess its effectiveness in handling complex timed automata models. The modifications to the standard reachability algorithm are quite straightforward to implement in tools like UPPAAL or KRONOS.

In the future we are interested in the study of sufficient conditions for a timed automaton model to be robust which can be checked more efficiently, as well as in devising meaningful model transformation techniques to make a model robust.

References

1. K. Altisen and S. Tripakis. Implementation of timed automata: an issue of semantics or modeling? Technical report, Verimag, Centre Équation, 38610 Gières, June 2005.
2. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

3. J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In W. Reisig and G. Rozenberg, editors, *In Lecture Notes on Concurrency and Petri Nets*, Lecture Notes in Computer Science vol 3098. Springer–Verlag, 2004.
4. P. Bouyer, N. Markey, and P.-A. Reynier. Robust model-checking of linear-time properties in timed automata. In J. R. Correa, A. Hevia, and M. Kiwi, editors, *Proceedings of the 7th Latin American Symposium on Theoretical Informatics (LATIN'06)*, volume 3887 of *Lecture Notes in Computer Science*, pages 238–249, Valdivia, Chile, Mar. 2006. Springer.
5. C. Daws. *Vérification de systèmes temporisés: de la théorie à la pratique*. PhD thesis, Institut National Polytechnique de Grenoble, 20 Oct. 1998.
6. C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool Kronos. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 208–219. Springer-Verlag, 1996.
7. M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. In *Formats - FTRTFT*, pages 118–133, 2004.
8. M. De Wulf, L. Doyen, and J. Raskin. Almost ASAP Semantics: from timed models to timed implementations, 2004.
9. V. Gupta, T. Henzinger, and R. Jagadeesan. Robust Timed Automata. In *Proc. Int. Work. Hybrid and Real-Time Systems (HART'97)*, volume 1201 of *LNCS*, pages 331–345. Springer, 1997.
10. K. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *Software Tools for Technology Transfer*, 1(1+2):134–152, 1997.
11. J. Ouaknine and J. Worrell. Revisiting digitization, robustness, and decidability for timed automata. In *LICS '03: Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science*, page 198, Washington, DC, USA, 2003. IEEE Computer Society.
12. A. Puri. Dynamical Properties of Timed Automata. In *FTRTFT '98: Proceedings of the 5th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 210–227, London, UK, 1998. Springer-Verlag.
13. A. Puri. Dynamical Properties of Timed Automata. *Discrete Event Dynamic Systems-Theory and Applications*, 10(1-2):87–113, Jan 2000.