Checking Timed Büchi Automata Emptiness on Simulation Graphs*

Stavros Tripakis

CNRS, Verimag Laboratory.
Address: Centre Equation, 2, avenue de Vignate, 38610 Gières, France.
Email: tripakis@imag.fr.
Web: http://www-verimag.imag.fr/~tripakis/.

Abstract. This paper completes the work of [5,13] on checking language emptiness of timed Büchi automata efficiently. In [5,13] we showed how to check emptiness on the region-closed simulation graph. However, the latter is not used in practice, since its nodes are non-convex, thus, not easily representable. Using recent results of Bouyer [6] on simulation-graph over-approximations that preserve convexity, we show that the main result of [5,13] carries over to the zone-closed simulation graph. The nodes of the latter are convex and can be efficiently represented. The zone-closed simulation graph is used in the tools Kronos and Uppaal for checking reachability. Our result shows that these tools can be also used to check emptiness of timed Büchi automata with small modifications.

Keywords: formal methods, specification languages, model checking, timed Büchi automata, property-preserving abstractions.

1 Introduction

It is well-known that model-checking problems involving timed automata (TA) and timed Büchi automata (TBA), such as language emptiness, can be solved using the *region graph* [1]. In practice, model-checking tools such as Kronos [7] or Uppaal [10] use a *simulation graph*, a graph that can be constructed on-the-fly, in a forward manner, and is usually much smaller than the region graph.

Different versions of the simulation graph exist. The *exact* simulation graph is based on the successor operator Post which gives the precise set of successor states for a given set of states, with respect to time-elapse followed by a discrete transition. Apart from being exact, Post has the advantage to *preserve convexity*: if $S = (q, Z)$ is a node in the graph such that the set of clock states Z is convex (in this case Z is called a *zone*) then the successor of S using Post is also convex. This is important since it allows efficient data-structures to be used, in particular, DBMs [9,4].

Unfortunately, the exact simulation graph may be infinite [8]. For this reason, finite versions of the exact graph have been developed, using an abstraction operator. One choice of an abstraction operator is the *region-closure* operator which returns, given a zone Z, the union of all regions intersecting Z. This operator can be used to define the *region-closed* simulation graph. The latter is finite, since the number of possible regions is finite.

* Work partially supported by CNRS STIC project "CORTOS" and by IST Network of Excellence "ARTIST2".

The region-closed simulation graph can be used to check language emptiness of TBA: a *strongly non-zeno* TBA[1] is non-empty iff its region-closed simulation graph has an accepting cycle [5,13]. Moreover, every TBA A can be effectively transformed into a strongly non-zeno TBA A' containing one extra clock, such that the language of A is empty iff the language of A' is empty [13]. Unfortunately, the region-closure of a zone is not generally a zone (i.e., it is not convex). Thus, the region-closed simulation graph is not used in practice.

Another choice of an abstraction operator is a *zone-closure* operator (called *maximization* in [11], *extrapolation* in [8] and *k-approximation* in [6]), which abstracts Z by another zone Z' such that all constraints defined in Z' are bounded by the maximal constant appearing in the guards and invariants of the automaton. This operator can be used to define the *zone-closed* simulation graph. This graph is finite since the number of zones with bounded constraints are bounded. This is the graph typically used in Kronos and Uppaal for checking *reachability*.

In [6], Bouyer showed that the zone-closed simulation graph is correct for reachability, provided the automaton does not have *diagonal* constraints. These are constraints of the form $x - y \leq c$, where x and y are clocks and c is a constant. Correct means that a location q is reachable in a diagonal-free TA A iff the zone-closed simulation graph of A contains a reachable node (q, Z). If A has diagonal constraints, its zone-closed simulation graph is generally an over-approximation, that is, it may contain unreachable locations [3,6].

The above work settled the question of language emptiness of "plain" TA. What about emptiness of timed Büchi automata? Can the zone-closed simulation graph be used for checking emptiness of TBA? We answer this question positively, thus completing the work of [5,13] on checking timed Büchi automata emptiness efficiently.

More precisely, we show that the language of a *strongly non-zeno* TBA A is empty iff the exact simulation graph of A (provided it is finite) contains no accepting cycle. We also show that the language of A is empty iff the zone-closed simulation graph of A contains no accepting cycle. These results are obtained by "lifting" cycles from the exact and zone-closed simulation graphs to the region-closed simulation graph, and then using the proof of [5,13] for the latter graph. The "lifting" is possible because of commutation properties of the approximation operators involved, which are proved using the results of Bouyer [6].

The rest of this paper is organized as follows. In Section 2 we recall timed Büchi automata. In Section 3 we present the three versions of the simulation graphs. In Section 4 we provide the results. Section 5 concludes this paper.

2 Timed Büchi Automata

Let $\mathbb{R}$ be the set of non-negative real numbers. Let $\mathcal{X}$ be a finite set of variables, called *clocks*, taking values in $\mathbb{R}$. A *valuation* on $\mathcal{X}$ is a function $\mathbf{v} : \mathcal{X} \to \mathbb{R}$ that assigns to each variable in $\mathcal{X}$ a value in $\mathbb{R}$. $\mathbf{0}$ denotes the valuation assigning 0 to all variables in $\mathcal{X}$. Given a valuation $\mathbf{v}$ and $\delta \in \mathbb{R}$, $\mathbf{v} + \delta$ is defined to be the valuation $\mathbf{v}'$ such that $\mathbf{v}'(x) = \mathbf{v}(x) + \delta$

[1] A TBA is strongly non-zeno if it has no accepting zeno run, i.e., every accepting run is guaranteed to be non-zeno.

for all $x \in \mathcal{X}$. Given a valuation $\mathbf{v}$ and $X \subseteq \mathcal{X}$, $\mathbf{v}[X := 0]$ is defined to be the valuation $\mathbf{v}'$ such that $\mathbf{v}'(x) = 0$ if $x \in X$ and $\mathbf{v}'(x) = \mathbf{v}(x)$ otherwise.

An *atomic constraint* on $\mathcal{X}$ is a constraint of one of the forms $x \# c$ or $x - y \# c$, where $x, y \in \mathcal{X}$, c is an integer constant and $\# \in \{<, \leq, =, \geq, >\}$. A boolean expression on atomic constraints defines a set of valuations, the ones satisfying the expression, called an *$\mathcal{X}$-polyhedron*. A conjunction of atomic constraints defines a *convex* polyhedron, or *zone*. $\mathcal{X}$-polyhedra using atomic constraints of the form $x - y \# c$ are called *diagonal*, otherwise, they are called *diagonal-free* [6].

Definition 1 (Timed Büchi Automata [1]). A *timed Büchi automaton* is a tuple $A = (\mathcal{X}, Q, q_0, E, I, F)$, where:

- $\mathcal{X}$ is a finite set of *clocks*.
- Q is a finite set of *locations* and $q_0 \in Q$ is the *initial* location.
- $F \subseteq Q$ is a finite set of *accepting* locations.
- E is a finite set of *edges* of the form $e = (q, Z, X, q')$, where $q, q' \in Q$ are the *source* and *target* locations, Z is a convex $\mathcal{X}$-polyhedron, called the *guard* of e, and $X \subseteq \mathcal{X}$ is a set of clocks to be *reset* upon crossing the edge.
- I is a function associating with each location q a convex $\mathcal{X}$-polyhedron, called the *invariant* of q.

A is said to be diagonal-free if all its guards and invariants are diagonal-free.

A *state* of A is a pair $s = (q, \mathbf{v})$, where $q \in Q$ and $\mathbf{v} \in I(q)$. The *initial state* of A is $s_0 = (q_0, \mathbf{0})$. Given two states $s = (q, \mathbf{v})$ and $s = (q', \mathbf{v}')$, and an edge $e = (q, Z, X, q')$, there is a *discrete transition* $s \xrightarrow{e} s'$ iff $\mathbf{v} \in Z$ and $\mathbf{v}' = \mathbf{v}[X := 0] \in I(q')$. Given $\delta \in \mathbb{R}$, there is a *time transition* $s \xrightarrow{\delta} s'$ iff $q = q'$ and $\mathbf{v}' = \mathbf{v} + \delta \in I(q)$. We write $s \xrightarrow{\delta} \xrightarrow{e} s'$ if there exists s'' such that $s \xrightarrow{\delta} s''$ and $s'' \xrightarrow{e} s'$.

An infinite run of A starting at state s is an infinite sequence

$$(s_0, \delta_0, e_0), (s_1, \delta_1, e_1), ...,$$

where $s_0 = s$ and for all $i = 0, 1, ...$, $s_i = (q_i, \mathbf{v}_i)$ is a state, $\delta_i \in \mathbb{R}$, $e_i \in E$, and $s_i \xrightarrow{\delta_i} \xrightarrow{e_i} s_{i+1}$. The run is called *accepting* if there exists an infinite set of indices i such that $q_i \in F$. The run is called *non-zeno* if $\forall t \in \mathbb{R}, \exists k, \sum_{i=0,...,k} \delta_i > t$, otherwise, it is called *zeno*.

Definition 2 (Language and emptiness problem). The *language* of A, denoted $Lang(A)$, is defined to be the set of all non-zeno accepting runs of A starting at the initial state s_0. The *emptiness problem* for A is to check whether $Lang(A) = \emptyset$.

The emptiness problem for timed Büchi automata is known to be PSPACE-complete [1].

Definition 3 (Strong non-zenoness). A timed Büchi automaton A is called *strongly non-zeno* if all accepting runs starting at the initial state of A are non-zeno.

Theorem 4 ([13]). *Any timed Büchi automaton A can be transformed into a strongly non-zeno timed Büchi automaton* $\mathsf{snz}(A)$*, such that $Lang(A) = \emptyset$ iff $Lang(\mathsf{snz}(A)) = \emptyset$.*

3 Simulation Graphs

Consider a TBA $A = (\mathcal{X}, Q, q_0, E, I, F)$. A *symbolic state* S is a pair (q, Z) where $q \in Q$ and Z is an $\mathcal{X}$-polyhedron. S is called *convex* iff Z is convex (i.e., Z is a zone). S represents a set of states of A, namely, $(q, Z) = \{(q, \mathbf{v}) \mid \mathbf{v} \in Z\}$.

We first provide a generic definition of a simulation graph, using a generic successor operator for symbolic states, $\mathsf{Succ}(\cdot, \cdot)$. We will then specialize this definition using different instances of Succ.

Given a symbolic state S and an edge e, $\mathsf{Succ}(S, e)$ returns a symbolic state S'. Given an initial symbolic state S_0, a simulation graph of A with respect to Succ and S_0, denoted $\mathsf{SG}_{\mathsf{Succ}}(A, S_0)$, is a labeled graph $(\mathcal{S}, S_0, \rightarrow)$, where $\mathcal{S}$ is the set of nodes, S_0 the initial node, and $\rightarrow$ the set of edges. $\mathcal{S}$ is defined to be the least set of non-empty symbolic states, such that:

1. $S_0 \in \mathcal{S}$ and
2. if $e \in E$, $S \in \mathcal{S}$ and $S' = \mathsf{Succ}(S, e)$ is non-empty, then $S' \in \mathcal{S}$.

$\mathsf{SG}_{\mathsf{Succ}}(A, S_0)$ has an edge $S \xrightarrow{e} S'$ iff $S, S' \in \mathcal{S}$ and $S' = \mathsf{Succ}(S, e)$. S' is called the e-successor of S. Notice that, given S and e, the e-successor of S is unique.

3.1 Exact simulation graph

A natural definition of the simulation graph is obtained using the following symbolic successor operator:

$$\mathsf{Post}(S, e) = \{s' | \exists s \in S. \exists \delta \in \mathbb{R}. s \xrightarrow{\delta} \xrightarrow{e} s'\}$$

Then, the *exact simulation graph* of A is the graph

$$\mathsf{SG}(A) = \mathsf{SG}_{\mathsf{Post}}(A, \{(q_0, \mathbf{0})\}).$$

The nodes of $\mathsf{SG}(A)$ contain all reachable states of A and nothing but the reachable states. Also, for every node (q, Z) of $\mathsf{SG}(A)$, Z is a zone, that is, Z is convex. This is an important feature, since it allows efficient data structures for representation of zones, such as DBMs [9], to be used. On the other hand, $\mathsf{SG}(A)$ can be infinite [8]. Thus, it is not appropriate for fully-automatic reachability checking. Different remedies to this problem exist, two of which are discussed in the sequel.

3.2 Region-closed simulation graph

A first possibility is to define the simulation graph as an abstraction of the *region graph* [1]. In particular, a symbolic state S can be replaced by the union of regions that S intersects. Since the number of regions is finite, there is a finite number of such unions, thus, finiteness of the simulation graph is guaranteed. This approach is taken in [5,13]. We briefly recall it here. We use the presentation of [6] in order to be uniform with the section that follows.

Let $\alpha = ((max_x)_{x \in \mathcal{X}}, (max_{x,y})_{x,y \in \mathcal{X}})$ be a tuple of maximal constants (or infinity) for each single clock x and each pair of clocks x, y, as defined in [6]. α defines a finite set of regions,

$\mathcal{R}_\alpha$ [6]. A region is a special case of a convex $\mathcal{X}$-polyhedron. A union of regions is also an $\mathcal{X}$-polyhedron, however, it is generally not convex (see [6] for examples). The set of convex unions of regions in $\mathcal{R}_\alpha$ is denoted $\mathcal{Z}_\alpha$ [6]. Note that $\mathcal{Z}_\alpha$ is a finite set.

Given an $\mathcal{X}$-polyhedron Z, define $\mathsf{Closure}_\alpha(Z)$ to be the union of all regions that intersect Z [6]:

$$\mathsf{Closure}_\alpha(Z) = \cup\{R \in \mathcal{R}_\alpha \mid R \cap Z \neq \emptyset\}.$$

We lift the definition to symbolic states as follows:

$$\mathsf{Closure}_\alpha((q, Z)) = (q, \mathsf{Closure}_\alpha(Z))$$

and define the composite successor operator:

$$\mathsf{Clo\_Post}(S, e) = \mathsf{Closure}_\alpha(\mathsf{Post}(S, e)).$$

Then, the *region-closed simulation graph* of A is the graph

$$\mathsf{SG}_{\mathcal{R}_\alpha}(A) = \mathsf{SG}_{\mathsf{Clo\_Post}}(A, \mathsf{Closure}_\alpha(\{(q_0, \mathbf{0})\})).$$

$\mathsf{SG}_{\mathcal{R}_\alpha}(A)$ is guaranteed to be finite and is also exact with respect to reachability of locations,[2] meaning that there is a node (q, Z) in $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$ iff there is a reachable state $(q, \mathbf{v})$ in A. On the other hand, since $\mathsf{Closure}_\alpha(Z)$ is not always convex, efficient data structures such as DBMs cannot be used.

3.3 Zone-closed simulation graph

In practice, tools such as Kronos or Uppaal use an over-approximation operator for $\mathcal{X}$-polyhedra, which ensures both convexity and finiteness. Such an operator was proposed in [8] and claimed to be exact with respect to reachability of locations. Later, Bouyer proved that this over-approximation is indeed exact for diagonal-free timed automata, but is not always exact for timed automata with diagonal constraints [6]. For the rest of this section, we assume that A is a diagonal-free TBA.

In the case of diagonal-free timed automata, $\mathcal{R}_\alpha$ can be defined as a "diagonal-free" set of regions, where α is such that $max_{x,y} = -\infty$ for all $x, y \in \mathcal{X}$ [6]. From this tuple α, we obtain a new tuple $\beta = ((max_x)_{x\in\mathcal{X}}, (max'_{x,y})_{x,y\in\mathcal{X}})$, by setting $max'_{x,y} = max_x$, for all $x, y \in \mathcal{X}$. Now, given a convex $\mathcal{X}$-polyhedron Z, define $\mathsf{Approx}_\beta(Z)$ to be the smallest zone in $\mathcal{Z}_\beta$ containing Z [6].

As previously, we lift the definition to symbolic states as follows:

$$\mathsf{Approx}_\beta((q, Z)) = (q, \mathsf{Approx}_\beta(Z))$$

and define the composite successor operator:

$$\mathsf{Apx\_Post}(S, e) = \mathsf{Approx}_\beta(\mathsf{Post}(S, e)).$$

[2] Reachability of general states can be reduced to reachability of locations by adding an extra location (the one to be reached) and annotating the edges to this location with a guard that encodes the states to be reached.

Then, the *zone-closed simulation graph* of A is the graph

$$\mathsf{SG}_{\mathcal{Z}_\beta}(A) = \mathsf{SG}_{\mathsf{Apx\_Post}}(A, \mathsf{Approx}_\beta(\{(q_0, \mathbf{0})\})).$$

$\mathsf{SG}_{\mathcal{Z}_\beta}(A)$ is finite since $\mathcal{Z}_\beta$ is finite. Bouyer shows that $\mathsf{SG}_{\mathcal{Z}_\beta}(A)$ is also exact with respect to reachability of locations [6].

3.4 Lassos

Any simulation graph is a discrete graph, thus, notions such as paths or cycles in this graph are easy to define. For our purpose, we will define a *lasso* as a path starting at the initial node followed by a cycle. More precisely, given a graph $(\mathcal{S}, S_0, \rightarrow)$, a lasso is a sequence

$$S_0 \overset{e_0}{\rightarrow} S_1 \overset{e_1}{\rightarrow} \cdots \overset{e_{n-1}}{\rightarrow} S_n \overset{e_n}{\rightarrow} \cdots \overset{e_{n+l-1}}{\rightarrow} S_{n+l} \overset{e_{n+l}}{\rightarrow} S_n$$

such that $n \geq 0$ and $l \geq 0$. That is, $S_0 \overset{e_0}{\rightarrow} S_1 \overset{e_1}{\rightarrow} \cdots \overset{e_{n-1}}{\rightarrow} S_n$ is a path from the initial node S_0 to a node S_n, and $S_n \overset{e_n}{\rightarrow} \cdots \overset{e_{n+l-1}}{\rightarrow} S_{n+l} \overset{e_{n+l}}{\rightarrow} S_n$ is a cycle from S_n to itself. We say that the lasso is accepting if its cycle visits some accepting node, that is, there is some $i \in \{0, ..., l\}$ such that $S_{n+i} = (q, Z)$ and $q \in F$.

4 Checking Timed Büchi Automata Emptiness

In this section we show how checking emptiness for strongly non-zeno timed Büchi automata (TBA for short) can be reduced to finding accepting cycles in the different types of simulation graphs, that is, to the problem of checking language emptiness for Büchi automata.

In all three theorems that follow, the direction "$Lang(A) \neq \emptyset$ implies simulation graph has an accepting lasso" (when finite) is easy to show and is based on the following "post-stability" property.

Lemma 5. *Let A be a TBA and let $s_0 \overset{\delta_0}{\rightarrow}\overset{e_0}{\rightarrow} s_1 \overset{\delta_1}{\rightarrow}\overset{e_1}{\rightarrow} \cdots$ be an infinite run of A. Then, in each of $\mathsf{SG}(A)$, $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$, $\mathsf{SG}_{\mathcal{Z}_\beta}(A)$, there exists an infinite path $S_0 \overset{e_0}{\rightarrow} S_1 \overset{e_1}{\rightarrow} \cdots$, such that for all $i = 0, 1, ...$, $s_i \in S_i$.*

The following two lemmata relate the edges of $\mathsf{SG}(A)$ and $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$ to those of the region graph. Given regions $R, R' \in \mathcal{R}_\alpha$, we use the notation $(q, R) \overset{\epsilon^*}{\rightarrow}_{rg} (q, R')$ to denote that (q, R') can be reached from (q, R) in the region graph by taking time-elapsing transitions. Given an edge e of A, we use the notation $(q, R) \overset{e}{\rightarrow}_{rg} (q', R')$ to denote that (q, R') can be reached from (q, R) in the region graph by taking the discrete transition e. We also write $(q, R) \overset{\epsilon^*}{\rightarrow}_{rg} \overset{e}{\rightarrow}_{rg} (q', R')$ if there exists $R_t \in \mathcal{R}_\alpha$ such that $(q, R) \overset{\epsilon^*}{\rightarrow}_{rg} (q, R_t)$ and $(q, R_t) \overset{e}{\rightarrow}_{rg} (q', R')$.

Lemma 6. *Let $S = (q_1, Z_1)$ and $S_2 = (q_2, Z_2) = \mathsf{Post}(S_1, e)$. Then:*

1. *For all regions $R_1, R_2 \in \mathcal{R}_\alpha$, if $R_1 \cap Z_1 \neq \emptyset$ and $(q_1, R_1) \overset{\epsilon^*}{\rightarrow}_{rg} \overset{e}{\rightarrow}_{rg} (q_2, R_2)$, then $R_2 \cap Z_2 \neq \emptyset$.*

2. *For any region* $R_2 \in \mathcal{R}_\alpha$ *such that* $R_2 \cap Z_2 \neq \emptyset$*, there exists a region* $R_1 \in \mathcal{R}_\alpha$ *such that* $R_1 \cap Z_1 \neq \emptyset$ *and* $(q_1, R_1) \xrightarrow{\epsilon^*}_{rg} \xrightarrow{e}_{rg} (q_2, R_2)$.

Proof. Part 1: Let $\mathbf{v}_1 \in R_1 \cap Z_1$. By definition of the region graph and the fact that $(q_1, R_1) \xrightarrow{\epsilon^*}_{rg} \xrightarrow{e}_{rg} (q_2, R_2)$, there exist $\delta \in \mathbb{R}$ and $\mathbf{v}_2 \in R_2$ such that $(q_1, \mathbf{v}_1) \xrightarrow{\delta} \xrightarrow{e} (q_2, \mathbf{v}_2)$. By definition of Post, $\mathbf{v}_2 \in Z_2$. Thus, $R_2 \cap Z_2 \neq \emptyset$.

Part 2: Let $\mathbf{v}_2 \in R_2 \cap Z_2$. By definition of Post, there exists $\delta \in \mathbb{R}$ and $\mathbf{v}_1 \in Z_1$ such that $(q_1, \mathbf{v}_1) \xrightarrow{\delta} \xrightarrow{e} (q_2, \mathbf{v}_2)$. Let $R_1 \in \mathcal{R}_\alpha$ be the region where $\mathbf{v}_1$ belongs. Clearly, $R_1 \cap Z_1 \neq \emptyset$. Also, by definition of the region graph, $(q_1, R_1) \xrightarrow{\epsilon^*}_{rg} \xrightarrow{e}_{rg} (q_2, R_2)$. □

Lemma 7. *Let* $S_1 \xrightarrow{e} S_2$ *be an edge of* $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$*, with* $S_i = (q_i, Z_i)$*, for* $i = 1, 2$*. For any region* $R_2 \subseteq Z_2$*, there exists a region* $R_1 \subseteq Z_1$ *such that* $(q_1, R_1) \xrightarrow{\epsilon^*}_{rg} \xrightarrow{e}_{rg} (q_2, R_2)$.

Proof. By definition of $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$, $S_2 = \mathsf{Closure}_\alpha(\mathsf{Post}(S_1, e))$. Let $S = (q_2, Z) = \mathsf{Post}(S_1, e)$. By definition of $\mathsf{Closure}_\alpha$, $R_2 \cap Z \neq \emptyset$. Thus, by part 2 of Lemma 6, there exists a region R_1 such that $(q_1, R_1) \xrightarrow{\epsilon^*}_{rg} \xrightarrow{e}_{rg} (q_2, R_2)$ and $R_1 \cap Z_1 \neq \emptyset$. By definition of $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$, S_1 is region-closed, that is, $\mathsf{Closure}_\alpha(S_1) = S_1$. Thus, $R_1 \subseteq Z_1$. □

We now recall an important result from [6].

Lemma 8 ([6]). *For any zone* Z*, the following holds:*

$$Z \subseteq \mathsf{Approx}_\beta(Z) \subseteq \mathsf{Closure}_\alpha(Z).$$

Based on the above, we can show the following:

Lemma 9. *For any zone* Z*,* $\mathsf{Closure}_\alpha(\mathsf{Approx}_\beta(Z)) = \mathsf{Closure}_\alpha(Z)$.

Proof. Indeed, by Lemma 8 and the monotonicity of $\mathsf{Closure}_\alpha$, we have

$$\mathsf{Closure}_\alpha(Z) \subseteq \mathsf{Closure}_\alpha(\mathsf{Approx}_\beta(Z)) \subseteq \mathsf{Closure}_\alpha(\mathsf{Closure}_\alpha(Z)).$$

The result follows from the fact that $\mathsf{Closure}_\alpha$ is idempotent:

$$\mathsf{Closure}_\alpha(\mathsf{Closure}_\alpha(Z)) = \mathsf{Closure}_\alpha(Z).$$

□

Lemma 10. *For any convex symbolic state* S *and any edge* e

$$\mathsf{Closure}_\alpha(\mathsf{Post}(S, e)) = \mathsf{Closure}_\alpha(\mathsf{Post}(\mathsf{Closure}_\alpha(S), e)).$$

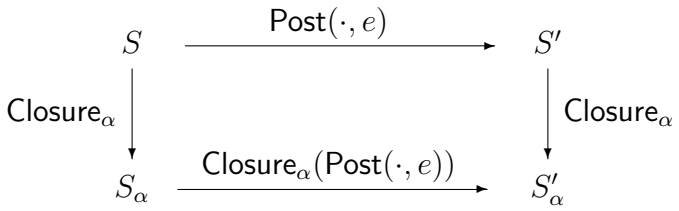

Fig. 1. Commutation diagram proved in Lemma 10.

Proof. Let $S = (q, Z)$. By Lemma 8 and the fact that $\mathsf{Closure}_\alpha(S) = (q, \mathsf{Closure}_\alpha(Z))$, we have

$$S \subseteq \mathsf{Closure}_\alpha(S).$$

By monotonicity of Post and $\mathsf{Closure}_\alpha$ operators, we have:

$$\mathsf{Closure}_\alpha(\mathsf{Post}(S, e)) \subseteq \mathsf{Closure}_\alpha(\mathsf{Post}(\mathsf{Closure}_\alpha(S), e)).$$

For the other direction, we will use the notation of Figure 1, namely,

$$S' = (q', Z') = \mathsf{Post}(S, e),$$

$$S_\alpha = (q, Z_\alpha) = \mathsf{Closure}_\alpha(S)$$

and

$$S'_\alpha = (q', Z'_\alpha) = \mathsf{Closure}_\alpha(\mathsf{Post}(S_\alpha, e)).$$

Using this notation, the proof objective becomes

$$Z'_\alpha \subseteq \mathsf{Closure}_\alpha(Z').$$

Let R' be a region contained in Z'_α. By definition of $\mathsf{Closure}_\alpha$,

$$(q', R') \cap \mathsf{Post}(S_\alpha, e) \neq \emptyset.$$

By part 2 of Lemma 6, there exists a region R such that $R \cap Z_\alpha \neq \emptyset$ and $(q, R) \xrightarrow{\epsilon^*}_{rg} \xrightarrow{e}_{rg} (q', R')$. Since $Z_\alpha = \mathsf{Closure}_\alpha(Z)$, it must be that $R \cap Z \neq \emptyset$. By part 1 of Lemma 6, $R' \cap Z' \neq \emptyset$. Thus, $R' \subseteq \mathsf{Closure}_\alpha(Z')$. □

Lemma 11. *For any convex symbolic state S and any edge e*

$$\mathsf{Closure}_\alpha(\mathsf{Approx}_\beta(\mathsf{Post}(S, e))) = \mathsf{Closure}_\alpha(\mathsf{Post}(\mathsf{Closure}_\alpha(S), e)).$$

Proof. By Lemma 9, we have

$$\mathsf{Closure}_\alpha(\mathsf{Approx}_\beta(\mathsf{Post}(S, e))) = \mathsf{Closure}_\alpha(\mathsf{Post}(S, e)).$$

The result follows from Lemma 10. □

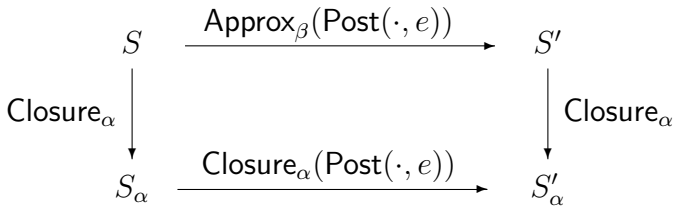

Fig. 2. Commutation diagram proved in Lemma 11.

4.1 Checking emptiness on the region-closed simulation graph

In order for the paper to be self-contained, we recall one of the main results of [5,13].

Theorem 12 ([5,13]). *Let A be a strongly non-zeno timed Büchi automaton. $Lang(A) = \emptyset$ iff $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$ contains no accepting lasso.*

Proof. The direction "$Lang(A) \neq \emptyset$ implies ..." is proven using Lemma 5 as mentioned above. We now illustrate the main idea of the proof of the converse, which is more involved. This is because simulation graphs are not generally *pre-stable* [12], which means that, given an edge $S \xrightarrow{e} S'$, it is not guaranteed that every state in S has a successor in S'. Thus, when we have a cycle, we cannot guarantee that, starting from an arbitrary state s_1 at some node in the cycle, we can find a successor s_2 of s_1, then s_3 of s_2, and so on, ad infinitum, in order to form an infinite run. That is, we cannot extract an infinite run from a cycle in a *forward* manner.

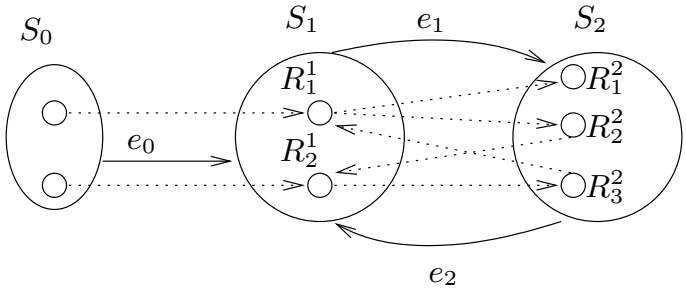

Fig. 3. Every lasso of $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$ contains a lasso of the region graph of A.

Instead, we proceed *backwards*, as illustrated in Figure 3. Let $S_i = (q_i, Z_i)$. We pick an arbitrary node in the cycle, say, S_2. Z_2 is a union of regions (the small circles drawn inside the ellipsis). We pick one such region, say R_1^2, arbitrarily. By Lemma 7, there exists some region $R_1^1 \subseteq Z_1$ such that

$$(q_1, R_1^1) \xrightarrow{\epsilon^*}_{rg} \xrightarrow{e_1}_{rg} (q_2, R_1^2).$$

Similarly, there exists $R_3^2 \subseteq Z_2$ such that

$$(q_2, R_3^2) \xrightarrow{\epsilon^*}_{rg} \xrightarrow{e_2}_{rg} (q_1, R_1^1),$$

and so on. Since the number of regions contained in any Z_i is finite, sooner or later the same region will be encountered, that is, a cycle will be found. In the case of the drawing of Figure 3 this cycle is

$$(q_1, R_1^1) \xrightarrow{\epsilon^*}_{rg} \xrightarrow{e_1}_{rg} (q_2, R_2^2) \xrightarrow{\epsilon^*} \xrightarrow{e_2}_{rg} (q_1, R_2^1) \xrightarrow{\epsilon^*}_{rg} \xrightarrow{e_1}_{rg} (q_2, R_3^2) \xrightarrow{\epsilon^*} \xrightarrow{e_2}_{rg} (q_1, R_1^1).^{3}$$

The above cycle can be extended backwards until the initial node S_0, so that a lasso is found. This lasso corresponds to a lasso in the region graph of A. Moreover, the lasso is accepting, since all regions in a node are associated with the same location, and the simulation-graph cycle is accepting. Then, using the pre-stability of the region graph we can extract an infinite accepting run from the lasso. Since A is strongly non-zeno, the run is also non-zeno, thus $Lang(A) \neq \emptyset$. □

4.2 Checking emptiness on the exact simulation graph

Theorem 13. *Let A be a strongly non-zeno timed Büchi automaton. If $Lang(A) = \emptyset$ then $\mathsf{SG}(A)$ contains no accepting lasso. If $Lang(A) \neq \emptyset$ and $\mathsf{SG}(A)$ is finite then $\mathsf{SG}(A)$ contains an accepting lasso.*

Proof. The direction "$Lang(A) \neq \emptyset$ implies ..." is proven using Lemma 5 as mentioned above. For the converse, consider an accepting lasso of $\mathsf{SG}(A)$:

$$S_0 \xrightarrow{e_0} S_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{n-1}} S_n \xrightarrow{e_n} \cdots \xrightarrow{e_{n+l-1}} S_{n+l} \xrightarrow{e_{n+l}} S_{n+l+1}, \text{ with } S_{n+l+1} = S_n.$$

Define $S'_i = \mathsf{Closure}_\alpha(S_i)$, for all $i = 0, ..., n+l$. We claim that

$$S'_0 \xrightarrow{e_0} S'_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{n-1}} S'_n \xrightarrow{e_n} \cdots \xrightarrow{e_{n+l-1}} S'_{n+l} \xrightarrow{e_{n+l}} S'_{n+l+1}$$

is an accepting lasso of $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$. The result follows from Theorem 12.

We now prove the claim. First, note that

$$S'_0 = \mathsf{Closure}_\alpha(S_0) = \mathsf{Closure}_\alpha(\{(q_0, \mathbf{0})\}),$$

which is indeed the initial node of $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$. Second, by Lemma 10, we have

$$S'_1 = \mathsf{Closure}_\alpha(S_1) = \mathsf{Closure}_\alpha(\mathsf{Post}(S_0, e_0)) = \mathsf{Closure}_\alpha(\mathsf{Post}(S'_0, e_0)).$$

Thus, S'_1 is indeed the e_0-successor of S'_0 in $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$. We can continue the same way, showing that S'_2 is the e_1-successor of S'_1 in $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$, etc. Since $S'_{n+l+1} = \mathsf{Closure}_\alpha(S_{n+l+1})$ and $S_{n+l+1} = S_n$, we have $S'_{n+l+1} = S'_n$, that is, we have a lasso. □

[3] Notice that R_1^2 is left out because it has no successors in S_1. This does not mean R_1^2 is a deadlock since there might be other successor nodes to S_2.

4.3 Checking emptiness on the zone-closed simulation graph

Theorem 14. *Let A be a strongly non-zeno timed Büchi automaton. $Lang(A) = \emptyset$ iff $\mathsf{SG}_{\mathcal{Z}_\beta}(A)$ contains no accepting lasso.*

Proof. The direction "$Lang(A) \neq \emptyset$ implies ..." is proven using Lemma 5 as mentioned above. For the converse, consider an accepting lasso of $\mathsf{SG}_{\mathcal{Z}_\beta}(A)$:

$$S_0 \xrightarrow{e_0} S_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{n-1}} S_n \xrightarrow{e_n} \cdots \xrightarrow{e_{n+l-1}} S_{n+l} \xrightarrow{e_{n+l}} S_{n+l+1}, \text{ with } S_{n+l+1} = S_n.$$

Define $S'_i = \mathsf{Closure}_\alpha(S_i)$, for all $i = 0, ..., n+l$. We claim that

$$S'_0 \xrightarrow{e_0} S'_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{n-1}} S'_n \xrightarrow{e_n} \cdots \xrightarrow{e_{n+l-1}} S'_{n+l} \xrightarrow{e_{n+l}} S'_{n+l+1}$$

is an accepting lasso of $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$. The result follows from Theorem 12.

We now prove the claim. First, note that

$$S'_0 = \mathsf{Closure}_\alpha(S_0) = \mathsf{Closure}_\alpha(\mathsf{Approx}_\beta(\{(q_0, \mathbf{0})\})),$$

which, by Lemma 9, is equal to $\mathsf{Closure}_\alpha(\{(q_0, \mathbf{0})\})$, which is indeed the initial node of $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$. Second, by Lemma 11, we have

$$S'_1 = \mathsf{Closure}_\alpha(S_1) = \mathsf{Closure}_\alpha(\mathsf{Approx}_\beta(\mathsf{Post}(S_0, e_0))) = \mathsf{Closure}_\alpha(\mathsf{Post}(S'_0, e_0)).$$

Thus, S'_1 is indeed the e_0-successor of S'_0 in $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$. We can continue the same way, showing that S'_2 is the e_1-successor of S'_1 in $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$, etc. Since $S'_{n+l+1} = \mathsf{Closure}_\alpha(S_{n+l+1})$ and $S_{n+l+1} = S_n$, we have $S'_{n+l+1} = S'_n$, that is, we have a lasso. □

5 Conclusions and Perspectives

This paper completes the work of [5,13] on checking language emptiness of timed Büchi automata efficiently. In [5,13] we showed how to check emptiness on the region-closed simulation graph. However, the latter is not used in practice, since its nodes are non-convex, thus, not easily representable. Using recent results of Bouyer [6] on simulation-graph over-approximations that preserve convexity, we show that the main result of [5,13] carries over to the zone-closed simulation graph, which is the graph used in the tools Kronos and Uppaal for checking reachability. Our result implies that these tools can be used not only for reachability, but also to check emptiness of timed Büchi automata. This can be done with small modifications to the tools, namely, implementing an algorithm to find cycles or strongly connected components in a graph. Our result also proves the correctness of timed Büchi automata emptiness algorithms implemented in the tool Open-Kronos.[4]

Perspectives of this work include studying other classes of properties, apart from reachability and Büchi emptiness, that are preserved in the zone-closed simulation graph. It would also be interesting to study whether other, coarser, zone-based abstractions, such as those proposed in [2], can be used to check timed Büchi automata emptiness.

[4] See `http://www-verimag.imag.fr/ tripakis/openkronos.html`.

References

1. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994. 1, 3, 4
2. G. Behrmann, P. Bouyer, K. Larsen, and R. Pelánek. Lower and upper bounds in zone based abstractions of timed automata. In *TACAS'04*, volume 2988 of *LNCS*. Springer, 2004. 11
3. J. Bengtsson and W. Yi. On clock difference constraints and termination in reachability analysis of timed automata. In *ICFEM'03*, volume 2885 of *LNCS*. Springer, 2003. 2
4. B. Berthomieu and M. Menasche. An enumerative approach for analyzing time Petri nets. *IFIP Congress Series*, 9:41–46, 1983. 1
5. A. Bouajjani, S. Tripakis, and S. Yovine. On-the-fly symbolic model checking for real-time systems. In *18th IEEE Real-Time Systems Symposium (RTSS'97)*, pages 25–34. IEEE, December 1997. 1, 2, 4, 9, 11
6. P. Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320, 2004. 1, 2, 3, 4, 5, 6, 7, 11
7. C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool Kronos. In *Hybrid Systems III, Verification and Control*, volume 1066 of *LNCS*, pages 208–219. Springer-Verlag, 1996. 1
8. C. Daws and S. Tripakis. Model checking of real-time reachability properties using abstractions. In *Tools and Algorithms for the Construction and Analysis of Systems '98, Lisbon, Portugal*, volume 1384 of *LNCS*. Springer-Verlag, 1998. 1, 2, 4, 5
9. D. Dill. Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *LNCS*, pages 197–212. Springer–Verlag, 1989. 1, 4
10. K. Larsen, P. Petterson, and W. Yi. Uppaal in a nutshell. *Software Tools for Technology Transfer*, 1(1/2), October 1997. 1
11. S. Tripakis and C. Courcoubetis. Extending Promela and Spin for real time. In *TACAS'96, Passau, Germany*, volume 1055 of *LNCS*. Springer-Verlag, 1996. 2
12. S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, January 2001. 9
13. S. Tripakis, S. Yovine, and A. Bouajjani. Checking timed Büchi automata emptiness efficiently. *Formal Methods in System Design*, 26(3):267–292, May 2005. 1, 2, 3, 4, 9, 11