Checking Timed Büchi Automata Emptiness on Simulation Graphs

STAVROS TRIPAKIS
University of California, Berkeley

Timed automata [Alur and Dill 1994] comprise a popular model for describing real-time and embedded systems and reasoning formally about them. Efficient model-checking algorithms have been developed and implemented in tools such as Kronos [Daws et al. 1996] or Uppaal [Larsen et al. 1997] for checking safety properties on this model, which amounts to reachability. These algorithms use the so-called zone-closed simulation graph, a finite graph that admits efficient representation and has been recently shown to preserve reachability [Bouyer 2004]. Building upon Bouyer [2004] and our previous work [Bouajjani et al. 1997; Tripakis et al. 2005], we show that this graph can also be used for checking liveness properties, in particular, emptiness of timed Büchi automata.

Categories and Subject Descriptors: D.2.4 [**Software Engineering**]: Software/Program Verification—*Model checking*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs

General Terms: Verification, Design, Languages

Additional Key Words and Phrases: Formal methods, specification languages, model checking, timed Büchi automata, property-preserving abstractions

ACM Reference Format:

Tripakis, S. 2009. Checking timed Büchi automata emptiness on simulation graphs. ACM Trans. Comput. Logic, 10, 3, Article 15 (April 2009), 19 pages. DOI = 10.1145/1507244.1507245 http://doi.acm.org/10.1145/1507244.1507245

1. INTRODUCTION

Advances in hardware have been steady over the past decades, and have resulted in execution platforms with impressive computing power. Availability of computing power makes possible the development of applications that are of increasing complexity. Design methods, however, are having a hard time keeping up with this trend. In particular, disproportional testing efforts are often

This work was done while S. Tripakis was with the Verimag Laboratory.

This research has been partially supported by the CNRS STIC project "CORTOS" and by the European IST Network of Excellence "ARTIST2".

Author's address: S. Tripakis, 545Q Cory Hall, University of California, Berkeley, CA 94720; email: stavros@eecs.berkeley.edu.

DOI 10.1145/1507244.1507245 http://doi.acm.org/10.1145/1507244.1507245

devoted to ensure that the applications function correctly, or that systems meet their requirements.

Formal verification is a methodology that aims to improve design quality with the use of formal models and proof techniques. In particular, it proposes to use such models to describe both the system and the requirements and then attempt to formally prove that the system meets the requirements. *Model checking* takes this further by proposing to use models where this proof can be carried out automatically, at least in principle. Different model checking techniques and tools have been developed in the past three decades, for different types of models: For instance, see Queille and Sifakis [1981] and Clarke et al. [1986, 2000].

In this article, we are interested in model checking for *timed automata* [Alur and Dill 1994]. Timed automata (TA) are finite automata equipped with real-valued variables, called *clocks*, that measure time. This feature makes the TA model particularly suitable for modeling systems where timing constraints are important. This is especially the case for real-time and embedded systems. In turn, these types of systems are often used in safety-critical applications, which makes model checking for TA an important problem.

TA model checking has been well studied. Even though the semantics of TA is infinite state (because of the clocks), many model-checking problems for TA are decidable [Alur et al. 1993; Alur and Dill 1994]. The reason is that the infinite state-space admits a finite abstraction, called the *region graph*, which preserves most properties of interest. This abstraction is defined by an equivalence relation on clock configurations, which induces a partition of the state space into a finite set of regions: Two states in a region are equivalent, and equivalent states have equivalent future behaviors, thus also satisfy the same sets of properties. Using the region graph, we can in principle reduce model checking for a TA into model checking for a finite-state automaton. The latter problem can be solved by an exhaustive exploration of the state space of the finite automaton.

In practice, the region graph is not used because it is too large: The number of nodes in the graph grows exponentially with the number of clocks and the size of constants used in the clock constraints. Even small examples can generate a huge number of regions, making exhaustive exploration intractable.

To remedy this problem, *symbolic* model-checking methods for TA have been developed, which attempt to reduce the size of the state space to be explored by operating on sets of states coarser than regions. Among these symbolic methods are the *fixpoint iteration* method of Henzinger et al. [1994] and the *partition refinement* method of Tripakis and Yovine [2001], both implemented in the tool Kronos [Daws et al. 1996]. Unfortunately, these methods are still limited by state explosion problems. One of the reasons is that they operate "backwards" and end up exploring unreachable parts of the state space (states that cannot be reached from the set of initial states). Another reason is that they operate on a single TA which needs to be constructed a priori as the product of a set of communicating timed automata: Building such an automaton often fails in practice, again due to state explosion.

In practice, TA model checkers such as Kronos or Uppaal [Larsen et al. 1997] use another type of graph, called the *simulation graph*. The latter has the advantage that it can be built on-the-fly, in a "forward" manner, without having to build the product of automata in advance. Moreover, the simulation graph is much smaller than the region graph in practice.

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 of *preserving 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 to represent Z, in particular, DBMs [Dill 1989; Berthomieu and Menasche 1983].

Unfortunately, the exact simulation graph may be infinite [Daws and Tripakis 1998], thus exhaustive exploration of this graph is not possible in general. To remedy this, 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. Unfortunately, the region closure of a zone is not generally a zone (i.e., it is not convex), therefore DBMs cannot be used. For this reason, the region-closed simulation graph is not used in practice.

Another choice of an abstraction operator is a *zone closure* operator (called *maximization* in Tripakis and Courcoubetis [1996], *extrapolation* in Daws and Tripakis [1998], and *k-approximation* in Bouyer [2004]), 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 is bounded. This is the graph typically used in Kronos and Uppaal for checking *reachability*, that is, whether a given state of the TA is reachable from a given set of initial states.

Bouyer [2004] 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 overapproximation, that is, it may contain unreachable locations [Bengtsson and Yi 2003; Bouyer 2004].

Reachability is important since it allows to express *safety* properties, which informally state that something "bad" will never happen. Another set of important properties are *liveness* properties, which state that something "good" will eventually happen [Lamport 1977]. Liveness properties cannot be stated in terms of reachability. Instead, among other possibilities, liveness properties can be expressed in terms of ω-automata, that is, automata on infinite words, such as Büchi automata. The model-checking problem then becomes checking whether the Büchi automaton modeling the composition of the system with the property (or its complement) is empty (i.e., accepts no infinite word). Büchi versions of timed automata exist and are called *timed Büchi automata* [Alur and Dill 1994].

The work of Bouyer settled the question of reachability, that is, language emptiness of "plain" TA. What about emptiness of timed Büchi automata (TBA)? Can the zone-closed simulation graph be used for checking emptiness of TBA? We answer this question positively for diagonal-free TBA. This completes our previous work [Bouajjani et al. 1997; Tripakis et al. 2005] on checking timed Büchi automata emptiness efficiently.

More precisely, we show that the language of a *strongly nonzeno* TBA A is empty iff the exact simulation graph of A (provided it is finite) contains no accepting cycle. A TBA is strongly nonzeno if it has no accepting *zeno* run, that is, an accepting run where time cannot progress. The strong nonzenoness assumption is not restrictive: as shown in Tripakis et al. [2005], every TBA A can be transformed into a strongly nonzeno TBA A' containing one extra clock, such that A is empty iff A' is empty. We also show that if A is diagonal free, then 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 Bouajjani et al. [1997] and Tripakis et al. [2005] 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 [2004].

The rest of this article 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 article.

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 $\mathbb{R}$. $\vec{0}$ 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 $\mathbf{v}'(x) = \mathbf{v}(x) + \delta$ 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.

Let $\mathbb{N}$ be the set of natural numbers, including 0. An *atomic constraint* on $\mathcal{X}$ is a constraint of one of the forms $x\#c$ or $x - y\#c$, where $x, y \in \mathcal{X}$, $c \in \mathbb{N}$ 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* [Bouyer 2004].

Definition 1 (*Timed Büchi Automata*) [*Alur and Dill* 1994]. 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.

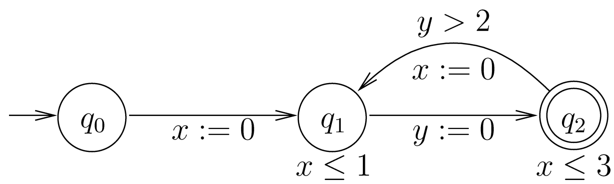

Fig. 1. A timed Büchi automaton.

—$F \subseteq Q$ 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* (to zero) upon crossing the edge.
—I is a function associating with each location $q \in Q$ 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.

An example of a timed Büchi automaton is shown in Figure 1. This automaton has two clocks x and y, three locations q_0, q_1, q_2, and three edges. There is one accepting location, q_2. The clock constraints annotating the locations are the invariants. The invariants of q_1 and q_2 are $x \leq 1$ and $x \leq 3$, respectively. The invariant of q_0 is omitted in the figure: When this is done we assume that the invariant is the trivial one which leaves all clocks unconstrained, namely, $\bigwedge_{x \in \mathcal{X}} x \geq 0$. The edges are annotated with clock resets and guards; for instance, the edge from q_2 to q_1 resets clock x and has a guard $y > 2$. When guards are omitted they are assumed trivial, as with invariants.

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 $s_0 = (q_0, \vec{0})$. 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* $s \stackrel{e}{\to} s'$ 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* $s \stackrel{\delta}{\to} s'$ iff $q = q'$ and $\mathbf{v}' = \mathbf{v} + \delta \in I(q)$. Notice that since $I(q)$ is assumed convex, the latter condition implies that $\forall \delta' \in [0, \delta], \mathbf{v} + \delta' \in I(q)$. We write $s \stackrel{\delta}{\to}\stackrel{e}{\to} s'$ if there exists s'' such that $s \stackrel{\delta}{\to} s''$ and $s'' \stackrel{e}{\to} s'$.

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

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

where $s_0 = s$ and for all $i = 0, 1, \ldots, s_i = (q_i, \mathbf{v}_i)$ is a state, $\delta_i \in \mathbb{R}$, $e_i \in E$, and $s_i \stackrel{\delta_i}{\to}\stackrel{e_i}{\to} s_{i+1}$. The run is called *accepting* if there exists an infinite set of indices i such that $q_i \in F$. The run is called *nonzeno* if $\forall t \in \mathbb{R}, \exists k, \sum_{i=0,\ldots,k} \delta_i > t$, 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 nonzeno accepting runs of A starting at the initial state s_0. The *emptiness problem* for A is to check whether *Lang*$(A) = \emptyset$.

For example, the automaton shown in Figure 1 has a nonempty language: Indeed, its has a nonzeno accepting run which visits locations $q_0, q_1, q_2, q_1, q_2, \ldots$, always spending zero time in q_0 and q_1, and 2.1 units of time in q_2.

The emptiness problem for timed Büchi automata is known to be PSPACE-complete [Alur and Dill 1994].

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

As an example, the automaton shown in Figure 1 is strongly nonzeno. This is because clock y is reset to zero and then lower bounded by 2 in the single loop that visits the accepting location q_2. This fact guarantees that at least 2 time units will be spent every time this location is visited.

It is shown in Tripakis et al. [2005] that any timed Büchi automaton A with n clocks, l locations, and m edges can be transformed into a strongly nonzeno timed Büchi automaton $\mathsf{snz}(A)$ with $n+1$ clocks and at most $2l$ locations and $(2m+1)n$ edges, such that $Lang(A) = \emptyset$ iff $Lang(\mathsf{snz}(A)) = \emptyset$. This result allows us, without loss of generality, to focus our attention on checking emptiness of strongly nonzeno timed Büchi automata.

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'.

Definition 4 (*Generic Simulation Graph*). Given an initial symbolic state S_0 and a successor operator $\mathsf{Succ}(\cdot, \cdot)$, the 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, defined to be the least set of nonempty 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 nonempty, then $S' \in \mathcal{S}$.

—S_0 is the initial node.

—$\rightarrow$ is the set of edges, defined as follows. $\mathsf{SG}_{\mathsf{Succ}}(A, S_0)$ 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.

Definition 5 (*Exact Symbolic Successors*).

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

Definition 6 (*Exact Simulation Graph*). The *exact simulation graph* of A is the graph

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

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 [Dill 1989], to be used. On the other hand, $\mathsf{SG}(A)$ can be infinite [Tripakis and Courcoubetis 1996; Daws and Tripakis 1998]. 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. 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 Bouajjani et al. [1997] and Tripakis et al. [2005]. We briefly recall it here.

Let us first introduce some notation. Let $\delta \in \mathbb{R}$. We define $\mathit{fract}(\delta)$ and $\lfloor\delta\rfloor$ to be the fractional and integral parts of δ, respectively; that is, $\delta = \mathit{fract}(\delta) + \lfloor\delta\rfloor$. For example, $\mathit{fract}(1.3) = 0.3$ and $\lfloor 1.3 \rfloor = 1$.

Let A be a TBA with a set of clocks $\mathcal{X}$. For each clock $x \in \mathcal{X}$, we define $c_x \in \mathbb{N}$ to be the greatest constant appearing in a guard or invariant of A that involves clock x, that is

$$c_x = \max\{c \mid x\#c \text{ or } x - y\#c \text{ or } y - x\#c \text{ is a constraint in a guard or invariant of } A\}.$$

For example, for the automaton shown in Figure 1, we have $c_x = 3$ and $c_y = 2$.

Definition 7 (*Regions*) [*Alur and Dill* 1994]. Let A be a TBA with set of clocks $\mathcal{X}$. Let $\alpha = (c_x)_{x\in\mathcal{X}}$ be the tuple of maximal constants for each clock in $\mathcal{X}$, as defined Previously. We define two equivalence relations between valuations on $\mathcal{X}$, the *region equivalence* and the *diagonal-free region equivalence*. Two valuations $\mathbf{v}, \mathbf{v}'$ are region equivalent, denoted $\mathbf{v} \sim \mathbf{v}'$, iff all the following conditions hold.

(1) For all $x \in \mathcal{X}$, either $\lfloor\mathbf{v}(x)\rfloor = \lfloor\mathbf{v}'(x)\rfloor$, or $\mathbf{v}(x) > c_x$ and $\mathbf{v}'(x) > c_x$.
(2) For all $x, y \in \mathcal{X}$ with $\mathbf{v}(x) \le c_x$ and $\mathbf{v}(y) \le c_y$, $\mathit{fract}(\mathbf{v}(x)) \le \mathit{fract}(\mathbf{v}(y))$ iff $\mathit{fract}(\mathbf{v}'(x)) \le \mathit{fract}(\mathbf{v}'(y))$.
(3) For all $x \in \mathcal{X}$ with $\mathbf{v}(x) \le c_x$ or $\mathbf{v}'(x) \le c_x$, $\mathit{fract}(\mathbf{v}(x)) = 0$ iff $\mathit{fract}(\mathbf{v}'(x)) = 0$.
(4) For all $x, y \in \mathcal{X}$, for any interval I in the set $\{(-\infty, -c_y), \{-c_y\}, (-c_y, -c_y + 1), \ldots, (-1, 0), \{0\}, (0, 1), \ldots, (c_x - 1, c_x), \{c_x\}, (c_x, +\infty)\}$ we have $\mathbf{v}(x) - \mathbf{v}(y) \in I$ iff $\mathbf{v}'(x) - \mathbf{v}'(y) \in I$.

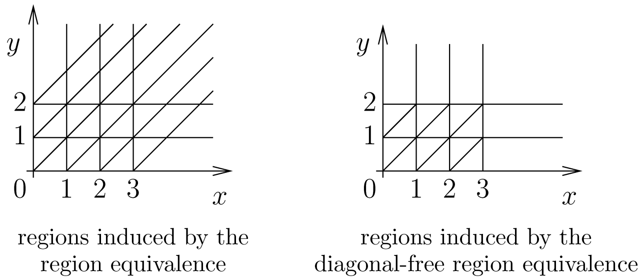

Fig. 2. Example of regions over two clocks x and y.

Two valuations $\mathbf{v}, \mathbf{v}'$ are diagonal-free region-equivalent, denoted $\mathbf{v} \sim_{df} \mathbf{v}'$, iff the preceding conditions 1–3 hold (but not necessarily condition 4).

Each region equivalence induces a partition of the space of valuations, $\mathbb{R}^{\mathcal{X}}$, into a finite number of equivalence classes, called regions. The set of regions induced by the region equivalence is denoted $\mathcal{R}_\alpha$. The set of regions induced by the diagonal-free region equivalence is denoted $\mathcal{R}_\alpha^{df}$. Notice that $\sim$ is stronger than $\sim_{df}$ thus $\mathcal{R}_\alpha$ is a finer partition than $\mathcal{R}_\alpha^{df}$. Also notice that since in each partition regions are disjoint and cover the entire valuation space, each valuation belongs to a unique region.

An example of the set of regions over two clocks x, y induced by constants $c_x = 3$ and $c_y = 2$ is provided in Figure 2. A valuation can be viewed as a point in the n-dimensional space $\mathbb{R}^n$, where n is the number of clocks. In this case there are two clocks, thus a valuation is a point in $\mathbb{R}^2$. The partition $\mathcal{R}_\alpha$ is illustrated to the left of the figure and the partition $\mathcal{R}_\alpha^{df}$ is illustrated to the right of the figure. Some examples of regions are given next.

—The singleton $\{(0, 1)\}$ is a region in both $\mathcal{R}_\alpha$ and $\mathcal{R}_\alpha^{df}$.
—The open triangle $0 < x < y < 1$ is a region in both $\mathcal{R}_\alpha$ and $\mathcal{R}_\alpha^{df}$.
—The unbounded subspace $x > 3 \wedge y > 2$ is a region in $\mathcal{R}_\alpha^{df}$ but not in $\mathcal{R}_\alpha$.
—The open line $2 < x = y < 3$ is a region in $\mathcal{R}_\alpha$ but not in $\mathcal{R}_\alpha^{df}$.

Definition 8 (*Region-Closure Operator*). Given an $\mathcal{X}$-polyhedron Z, define $\mathsf{Closure}_\alpha(Z)$ to be the union of all regions that intersect Z.

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

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.

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

Definition 9 (*Region-Closed Simulation Graph*). The *region-closed simulation graph* of A is the graph

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

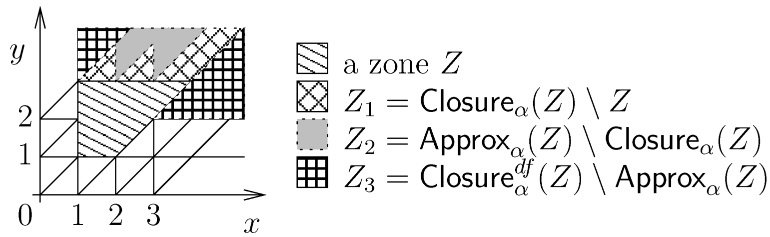

Fig. 3. Illustration of overapproximation operators.

$\mathsf{SG}_{\mathcal{R}_\alpha}(A)$ is guaranteed finite and is also exact with respect to reachability of locations,[1] 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.

$\mathsf{Closure}_\alpha$, $\mathsf{Clo\_Post}_\alpha$, and $\mathsf{SG}_{\mathcal{R}_\alpha}$ have been defined with respect to the set of regions $\mathcal{R}_\alpha$, induced by the region equivalence. In the same manner, we define $\mathsf{Closure}^{df}_\alpha$, $\mathsf{Clo\_Post}^{df}_\alpha$, and $\mathsf{SG}_{\mathcal{R}^{df}_\alpha}$ with respect to the set of regions $\mathcal{R}^{df}_\alpha$, induced by the diagonal-free region equivalence.

$$\begin{aligned}
\mathsf{Closure}^{df}_\alpha(Z) &= \cup\{R \in \mathcal{R}^{df}_\alpha \mid R \cap Z \neq \emptyset\} \\
\mathsf{Closure}^{df}_\alpha((q, Z)) &= (q, \mathsf{Closure}^{df}_\alpha(Z)) \\
\mathsf{Clo_Post}^{df}_\alpha(S, e) &= \mathsf{Closure}^{df}_\alpha(\mathsf{Post}(S, e)) \\
\mathsf{SG}_{\mathcal{R}^{df}_\alpha}(A) &= \mathsf{SG}_{\mathsf{Clo_Post}^{df}_\alpha}(A, \mathsf{Closure}^{df}_\alpha(\{(q_0, \vec{0})\}))
\end{aligned}$$

Notice that, by definition, $Z \subseteq \mathsf{Closure}_\alpha(Z)$, for any Z. Also, since $\mathcal{R}_\alpha$ is a finer partition than $\mathcal{R}^{df}_\alpha$, it follows that for any Z,

$$Z \subseteq \mathsf{Closure}_\alpha(Z) \subseteq \mathsf{Closure}^{df}_\alpha(Z). \tag{1}$$

Illustrations of the $\mathsf{Closure}_\alpha$ and $\mathsf{Closure}^{df}_\alpha$ operators are provided in Figure 3. It is assumed that $\mathcal{R}_\alpha$ and $\mathcal{R}^{df}_\alpha$ are as shown in Figure 2. The figure shows a zone Z defined by the constraints $x \geq 1 \wedge 1 \leq y \leq 3 \wedge x - y \leq 1$. Three successively larger overapproximations of Z are also shown.

$$\begin{aligned}
\mathsf{Closure}_\alpha(Z) &= Z \cup Z_1 \\
\mathsf{Approx}_\alpha(Z) &= Z \cup Z_1 \cup Z_2 \\
\mathsf{Closure}^{df}_\alpha(Z) &= Z \cup Z_1 \cup Z_2 \cup Z_3
\end{aligned}$$

Notice that both $\mathsf{Closure}_\alpha(Z)$ and $\mathsf{Closure}^{df}_\alpha(Z)$ are nonconvex in this example. For instance, $Z_3 = x \geq 1 \wedge y \geq 2$. The Approx_α operator is defined next.

3.3 Zone-Closed Simulation Graph

As illustrated in the preceding example, the $\mathsf{Closure}_\alpha$ and $\mathsf{Closure}^{df}_\alpha$ operators may yield nonconvex polyhedra: This is problematic since this kind of polyhedra does not admit efficient data structure representations. In practice, tools such as

[1]Reachability of 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. This assumes that the target states are definable by guards.

Kronos or Uppaal use a *zone-closure* overapproximation operator which ensures both convexity and finiteness. Such an operator was first proposed in Tripakis and Courcoubetis [1996] and Daws and Tripakis [1998] and claimed in Daws and Tripakis [1998] to be exact with respect to reachability of locations. Later, Bouyer proved that this overapproximation is indeed exact for diagonal-free timed automata, but is not always exact for timed automata with diagonal constraints [Bouyer 2004]. For the rest of this section, we assume that A is a diagonal-free TBA.

For the definition that follows, let A be a diagonal-free TBA with set of clocks $\mathcal{X}$. Let $\alpha = (c_x)_{x\in\mathcal{X}}$ be the tuple of maximal constants for each clock in $\mathcal{X}$, and $\mathcal{R}_\alpha$ be the set of regions induced by the region equivalence, as defined earlier. A union of regions in $\mathcal{R}_\alpha$ is an $\mathcal{X}$-polyhedron; however, it is generally not convex. Let $\mathcal{Z}_\alpha$ be the set of all convex unions of regions in $\mathcal{R}_\alpha$. By definition, every element of $\mathcal{Z}_\alpha$ is a zone. Note that $\mathcal{Z}_\alpha$ is a finite set, since $\mathcal{R}_\alpha$ is finite. Also note that $\mathcal{Z}_\alpha$ is closed by intersection, that is, if $Z_1 \in \mathcal{Z}_\alpha$ and $Z_2 \in \mathcal{Z}_\alpha$ then $Z_1 \cap Z_2 \in \mathcal{Z}_\alpha$.

Definition 10 (*Zone-Closure Operator*). Let Z be a convex $\mathcal{X}$-polyhedron. $\mathsf{Approx}_\alpha(Z)$ is defined to be the smallest zone in $\mathcal{Z}_\alpha$ containing Z.

Note that $\mathsf{Approx}_\alpha(Z)$ is well defined since $\mathcal{Z}_\alpha$ is finite and closed by intersection. Also note that, even though A is diagonal free, $\mathsf{Approx}_\alpha(Z)$ is defined with respect to the set of regions $\mathcal{R}_\alpha$ and not with respect to the set of regions $\mathcal{R}_\alpha^{df}$. Finally, note that, by definition, for any zone Z,

$$\mathsf{Closure}_\alpha(Z) \subseteq \mathsf{Approx}_\alpha(Z). \tag{2}$$

The Approx_α operator is illustrated in Figure 3. It is assumed that $\mathcal{R}_\alpha$ and $\mathcal{R}_\alpha^{df}$ are as shown in Figure 2. Zone Z is defined by the constraints $x \geq 1 \wedge 1 \leq y \leq 3 \wedge x - y \leq 1$ as explained before. $\mathsf{Approx}_\alpha(Z)$ is defined by the constraints $x \geq 1 \wedge y \geq 1 \wedge -2 \leq x - y \leq 1$.

An important result from Bouyer [2004] is the following.

LEMMA 1 [BOUYER 2004]. *For any zone Z:*

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

Combining Lemma 1 with Properties (1) and (2), we obtain

$$Z \subseteq \mathsf{Closure}_\alpha(Z) \subseteq \mathsf{Approx}_\alpha(Z) \subseteq \mathsf{Closure}_\alpha^{df}(Z). \tag{3}$$

An example is provided in Figure 3, where we have a sequence of strict inclusions: $Z \subset \mathsf{Closure}_\alpha(Z) \subset \mathsf{Approx}_\alpha(Z) \subset \mathsf{Closure}_\alpha^{df}(Z)$.

To proceed with the definition of the zone-closed simulation graph, we lift the definition of Approx_α to symbolic states:

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

and define the composite successor operator.

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

Definition 11 (*Zone-Closed Simulation Graph*). The *zone-closed simulation graph* of a diagonal-free timed Büchi automaton A is the graph

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

$\mathsf{SG}_{\mathcal{Z}_\alpha}(A)$ is finite since $\mathcal{Z}_\alpha$ is finite. Bouyer shows that $\mathsf{SG}_{\mathcal{Z}_\alpha}(A)$ is also exact with respect to reachability of locations [Bouyer 2004].

From now on, when we refer to the zone-closed simulation graph of a TBA A, we will assume that A is a diagonal-free TBA.

3.4 Lassos

Any simulation graph is a discrete graph; thus, notions such as paths or cycles in this graph are defined in the standard way. A *lasso* is 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

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

such that $n \geq 0$ and $l \geq 0$. In other words, $S_0 \xrightarrow{e_0} S_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{n-1}} S_n$ is a path from the initial node S_0 to a node S_n, and $S_n \xrightarrow{e_n} \cdots \xrightarrow{e_{n+l-1}} S_{n+l} \xrightarrow{e_{n+l}} S_n$ 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 $i \in \{0, \ldots, l\}$ 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 nonzeno timed Büchi automata (TBA for short) can be reduced to finding accepting cycles in the different types of simulation graphs.

In all three theorems that follow, the direction "$Lang(A) \neq \emptyset$ implies that the simulation graph has an accepting lasso" (provided the simulation graph is finite) is easy to show and is based on the following "poststability" property, provided here without a proof.

LEMMA 2. *Let A be a TBA and let $s_0 \xrightarrow{\delta_0} \xrightarrow{e_0} s_1 \xrightarrow{\delta_1} \xrightarrow{e_1} \cdots$ be an infinite run of A. Then, in each of $\mathsf{SG}(A)$, $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$, $\mathsf{SG}_{\mathcal{R}_\alpha^{df}}(A)$, $\mathsf{SG}_{\mathcal{Z}_\alpha}(A)$, there exists an infinite path $S_0 \xrightarrow{e_0} S_1 \xrightarrow{e_1} \cdots$, such that for all $i = 0, 1, \ldots, s_i \in S_i$.*

The following two lemmata relate the edges of $\mathsf{SG}(A)$, $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$, and $\mathsf{SG}_{\mathcal{R}_\alpha^{df}}(A)$ to those of the region graph. We first recall the definition of the region graph.

Definition 12 (*Region Graph*) [*Alur and Dill* 1994]. Let A be a TBA with set of locations Q and tuple of clock constants α. Let $\mathcal{R}$ be the set of regions $\mathcal{R}_\alpha$ or $\mathcal{R}_\alpha^{df}$. The region graph with respect to $\mathcal{R}$ is a graph whose nodes are of the form (q, R), where $q \in Q$ and $R \in \mathcal{R}$. The initial node of the graph is $(q_0, \{\vec{0}\})$. The graph has two types of transitions: time-elapsing transitions of the form $(q, R) \xrightarrow{\epsilon}_{rg} (q, R')$ iff there exist valuations $\mathbf{v} \in R$ and $\mathbf{v}' \in R'$, and $\delta \in \mathbb{R}$, such that $(q, \mathbf{v}) \xrightarrow{\delta} (q, \mathbf{v}')$; and discrete transitions of the form $(q, R) \xrightarrow{e}_{rg} (q', R')$, where e is an edge of A, iff there exist valuations $\mathbf{v} \in R$ and $\mathbf{v}' \in R'$, such that $(q, \mathbf{v}) \xrightarrow{e} (q', \mathbf{v}')$. We use $\xrightarrow{\epsilon^*}_{rg}$ to denote the reflexive, transitive closure of $\xrightarrow{\epsilon}_{rg}$.

We also write $(q, R)\xrightarrow{\epsilon^*}_{rg}\xrightarrow{e}_{rg}(q', R')$ if there exists R_t such that $(q, R)\xrightarrow{\epsilon^*}_{rg}(q, R_t)$ and $(q, R_t)\xrightarrow{e}_{rg}(q', R')$.

LEMMA 3. *Let $S_1 = (q_1, Z_1)$ and $S_2 = (q_2, Z_2) = \mathsf{Post}(S_1, e)$. Let $\mathcal{R}$ be either $\mathcal{R}_\alpha$ or $\mathcal{R}_\alpha^{df}$ and $\rightarrow_{rg}$ denote the transitions of the corresponding region graphs, respectively. Then:*

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

(2) *For any region $R_2 \in \mathcal{R}$ such that $R_2 \cap Z_2 \neq \emptyset$, there exists a region $R_1 \in \mathcal{R}$ 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 $(q_1, \mathbf{v}_1) \xrightarrow{\delta}\xrightarrow{e} (q_2, \mathbf{v}_2)$. Then, 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 $R_1 \in \mathcal{R}$ be the region where $\mathbf{v}_1$ belongs. Clearly, $R_1 \cap Z_1 \neq \emptyset$. Also, by definition of the region graph, $(q_1, R_1)\xrightarrow{\epsilon^*}_{rg}\xrightarrow{e}_{rg}(q_2, R_2)$. $\square$

LEMMA 4. *Let $S_1 \xrightarrow{e} S_2$ be an edge of $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$ or $\mathsf{SG}_{\mathcal{R}_\alpha^{df}}(A)$, 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 $(q_1, R_1)\xrightarrow{\epsilon^*}_{rg}\xrightarrow{e}_{rg}(q_2, R_2)$ in the corresponding region graph.*

PROOF. Consider first the case of $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$. 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 3, there exists a region $R_1 \in \mathcal{R}_\alpha$ 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 $R_1 \subseteq Z_1$. The case of $\mathsf{SG}_{\mathcal{R}_\alpha^{df}}(A)$ is similar, with the difference that $S_2 = \mathsf{Closure}_\alpha^{df}(\mathsf{Post}(S_1, e))$. Again we use part 2 of Lemma 3. $\square$

Based on Lemma 1, we can show the following.

LEMMA 5. *For any zone Z:*

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

PROOF. Indeed, by property (3) and the monotonicity of $\mathsf{Closure}_\alpha^{df}$, we have

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

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

$$\mathsf{Closure}_\alpha^{df}(\mathsf{Closure}_\alpha^{df}(Z)) = \mathsf{Closure}_\alpha^{df}(Z) \quad \square$$

The example shown in Figure 3 can be used to illustrate the aforementioned result. Indeed, the zone Z shown in the figure verifies the equality $\mathsf{Closure}_\alpha^{df}(\mathsf{Approx}_\alpha(Z)) = \mathsf{Closure}_\alpha^{df}(Z)$ as expected. Notice that this equality does not hold in general if we replace $\mathsf{Closure}_\alpha^{df}$ by $\mathsf{Closure}_\alpha$. This is to be expected since $\mathsf{Approx}_\alpha(Z)$ can be strictly larger than $\mathsf{Closure}_\alpha(Z)$, and $\mathsf{Closure}_\alpha(\mathsf{Approx}_\alpha(Z))$ is larger than $\mathsf{Approx}_\alpha(Z)$. Indeed, this situation occurs in Figure 3.

LEMMA 6. *For any convex symbolic state S and any edge e*

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

and

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

PROOF. Let $S = (q,Z)$. By definition, $\mathsf{Closure}_\alpha(S) = (q,\mathsf{Closure}_\alpha(Z))$ and $\mathsf{Closure}^{df}_\alpha(S) = (q,\mathsf{Closure}^{df}_\alpha(Z))$. From $Z \subseteq \mathsf{Closure}^{df}_\alpha(Z) \subseteq \mathsf{Closure}_\alpha(Z)$, we obtain

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

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

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

and

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

This proves the $\subseteq$-direction for both equalities.

For the other direction, we will use the notation of Figure 4, 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 first 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 3, there exists a region $R \in \mathcal{R}_\alpha$ 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 3, $R' \cap Z' \neq \emptyset$. Thus, $R' \subseteq \mathsf{Closure}_\alpha(Z')$. The proof is similar for $\mathsf{Closure}^{df}_\alpha$. Again we use Lemma 3. □

LEMMA 7. *For any convex symbolic state S and any edge e*

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

PROOF. By Lemma 5, we have

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

The result follows from Lemma 6 (also see Figure 5). □

$$
\begin{tikzcd}
S \arrow[r, "\mathsf{Post}(\cdot, e)"] \arrow[d, "\mathsf{Closure}_\alpha"'] & S' \arrow[d, "\mathsf{Closure}_\alpha"] \\
S_\alpha \arrow[r, "\mathsf{Closure}_\alpha(\mathsf{Post}(\cdot, e))"] & S'_\alpha
\end{tikzcd}
$$

$$
\begin{tikzcd}
S \arrow[r, "\mathsf{Post}(\cdot, e)"] \arrow[d, "\mathsf{Closure}^{df}_\alpha"'] & S'_{df} \arrow[d, "\mathsf{Closure}^{df}_\alpha"] \\
S^{df}_\alpha \arrow[r, "\mathsf{Closure}^{df}_\alpha(\mathsf{Post}(\cdot, e))"] & S'_{\alpha,df}
\end{tikzcd}
$$

Fig. 4. Commutation diagrams proved in Lemma 6.

$$
\begin{tikzcd}
S \arrow[r, "\mathsf{Approx}_\alpha(\mathsf{Post}(\cdot, e))"] \arrow[d, "\mathsf{Closure}^{df}_\alpha"'] & S' \arrow[d, "\mathsf{Closure}^{df}_\alpha"] \\
S_\alpha \arrow[r, "\mathsf{Closure}^{df}_\alpha(\mathsf{Post}(\cdot, e))"] & S'_\alpha
\end{tikzcd}
$$

Fig. 5. Commutation diagram proved in Lemma 7.

4.1 Checking Emptiness on the Region-Closed Simulation Graphs

In order for the article to be self-contained, we recall one of the main results of Bouajjani et al. [1997] and Tripakis et al. [2005].

THEOREM 1 [BOUAJJANI ET AL. 1997; TRIPAKIS ET AL. 2005]. *Let A be a strongly nonzeno timed Büchi automaton. $Lang(A) \neq \emptyset$ iff $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$ contains an accepting lasso. If A is diagonal free then $Lang(A) \neq \emptyset$ iff $\mathsf{SG}_{\mathcal{R}^{df}_\alpha}(A)$ contains an accepting lasso.*

The direction "$Lang(A) \neq \emptyset$ implies ... " in the previous theorem can be proven using Lemma 2. We now illustrate the main idea of the proof of the converse, which is more involved. This is because simulation graphs are not generally *prestable* [Tripakis and Yovine 2001], which means that, given an edge $S \stackrel{e}{\to} S'$, 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. In other words, we cannot extract an infinite run from a cycle in a *forward* manner.

Instead, we proceed *backwards*, as illustrated in Figure 6. The idea applies to both region graphs with respect to $\mathcal{R}_\alpha$ or $\mathcal{R}^{df}_\alpha$; therefore, in the discussion that

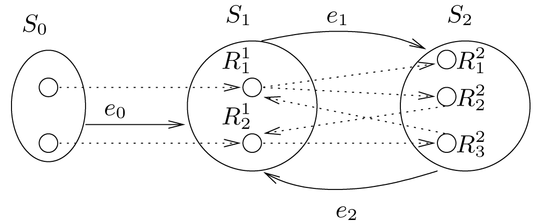

Fig. 6. Every lasso of a region-closed simulation graph contains a lasso of the corresponding region graph.

follows we simply refer to "regions" and "region graphs" without specifying which of the two.

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 4, 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 6 this cycle is

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

The aforesaid 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 results of Alur and Dill [1994] and the prestability property of the region graphs we can extract an infinite accepting run from the lasso: In the case of the region graph with respect to $\mathcal{R}_\alpha^{df}$ we also use the fact that A is diagonal free (otherwise it is not guaranteed that such a run exists). Since A is strongly nonzeno, the run is also nonzeno, thus $Lang(A) \neq \emptyset$.

4.2 Checking Emptiness on the Exact Simulation Graph

THEOREM 2. *Let A be a strongly nonzeno timed Büchi automaton. If $Lang(A) = \emptyset$ then* SG(A) *contains no accepting lasso. If $Lang(A) \neq \emptyset$ and* SG(A) *is finite then* SG(A) *contains an accepting lasso.*

PROOF. The direction "$Lang(A) \neq \emptyset$ implies ... " is proven using Lemma 2 as mentioned previously. For the converse, suppose SG(A) has an accepting lasso.

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

[2] 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.

Define $S'_i = \mathsf{Closure}_\alpha(S_i)$, for all $i = 0, \ldots, n+l$. 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 1.

We now prove the claim. First, we have

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

thus, S'_0 is indeed the initial node of $\mathsf{SG}_{\mathcal{R}_\alpha}(A)$.

Second, we have

$$\begin{aligned} S'_1 = \mathsf{Closure}_\alpha(S_1) &= \mathsf{Closure}_\alpha(\mathsf{Post}(S_0, e_0)) \\ &\stackrel{\text{by Lemma 6}}{=} \mathsf{Closure}_\alpha(\mathsf{Post}(\mathsf{Closure}_\alpha(S_0), e_0)) \\ &= \mathsf{Closure}_\alpha(\mathsf{Post}(S'_0, e_0)). \end{aligned}$$

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. $\square$

4.3 Checking Emptiness on the Zone-Closed Simulation Graph

THEOREM 3. *Let A be a strongly nonzeno and diagonal-free timed Büchi automaton. $Lang(A) \neq \emptyset$ iff $\mathsf{SG}_{\mathcal{Z}_\alpha}(A)$ contains an accepting lasso.*

PROOF. The direction "$Lang(A) \neq \emptyset$ implies ... " is proven using Lemma 2 as mentioned earlier. For the converse, suppose $\mathsf{SG}_{\mathcal{Z}_\alpha}(A)$ has an accepting lasso.

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

Define $S'_i = \mathsf{Closure}^{df}_\alpha(S_i)$, for all $i = 0, \ldots, n+l$. 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 $\mathsf{SG}_{\mathcal{R}^{df}_\alpha}(A)$. The result follows from Theorem 1.

We now prove the claim. First, we have

$$\begin{aligned} S'_0 = \mathsf{Closure}^{df}_\alpha(S_0) &= \mathsf{Closure}^{df}_\alpha(\mathsf{Approx}_\alpha(\{(q_0, \vec{0})\})) \\ &\stackrel{\text{by Lemma 5}}{=} \mathsf{Closure}^{df}_\alpha(\{(q_0, \vec{0})\}). \end{aligned}$$

Thus, S'_0 is indeed the initial node of $\mathsf{SG}_{\mathcal{R}^{df}_\alpha}(A)$.

Second, we have

$$\begin{aligned} S'_1 = \mathsf{Closure}^{df}_\alpha(S_1) &= \mathsf{Closure}^{df}_\alpha(\mathsf{Approx}_\alpha(\mathsf{Post}(S_0, e_0))) \\ &\stackrel{\text{by Lemma 7}}{=} \mathsf{Closure}^{df}_\alpha(\mathsf{Post}(\mathsf{Closure}^{df}_\alpha(S_0), e_0)) \\ &= \mathsf{Closure}^{df}_\alpha(\mathsf{Post}(S'_0, e_0)). \end{aligned}$$

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

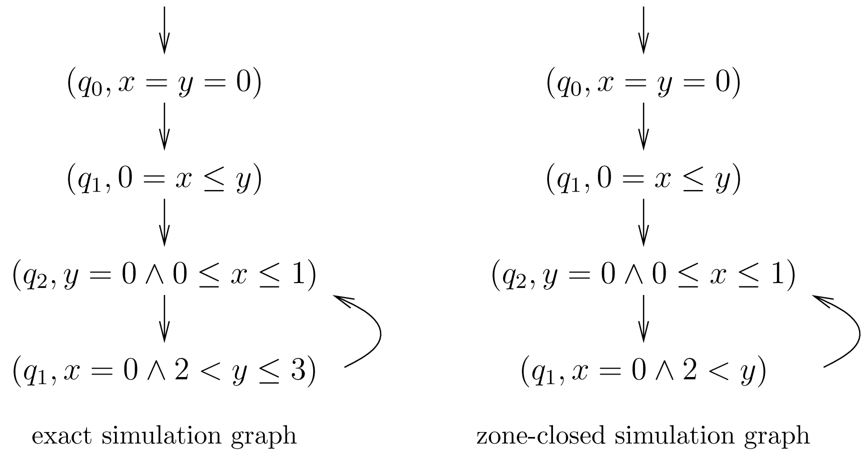

Fig. 7. Exact and zone-closed simulation graphs for the timed Büchi automaton shown in Figure 1.

is, we have a lasso in $\mathsf{SG}_{\mathcal{R}_\alpha^{df}}(A)$. Moreover, this is an accepting lasso since it visits the same locations as the original lasso of $\mathsf{SG}_{\mathcal{Z}_\alpha}(A)$. $\square$

4.4 An Example

We end this section with a simple example of how simulation graphs can be used to check emptiness of timed Büchi automata. We consider the TBA shown in Figure 1. Its exact simulation graph and its zone-closed simulation graph are shown in Figure 7. Notice that the exact simulation graph is finite in this case. Also note that the two graphs differ only in one node: node $(q_1, x = 0 \wedge 2 < y \le 3)$ in the exact graph versus node $(q_1, x = 0 \wedge 2 < y)$ in the zone-closed graph. Indeed, zone $x = 0 \wedge 2 < y$ is the zone closure of zone $x = 0 \wedge 2 < y \le 3$ with respect to the set of regions shown in Figure 2.

Both graphs have accepting cycles, which implies that the language of the automaton is nonempty. Indeed this is the case as stated in Section 2.

5. CONCLUSIONS AND PERSPECTIVES

This article completes the work of Bouajjani et al. [1997] and Tripakis et al. [2005] on checking language emptiness of timed Büchi automata efficiently. In Bouajjani et al. [1997] and Tripakis et al. [2005] we showed how to check emptiness on the region-closed simulation graph. However, the latter is not used in practice, since its nodes are nonconvex, thus, not efficiently representable. Using recent results of Bouyer [2004] on simulation-graph overapproximations that preserve convexity, we show that the main result of Bouajjani et al. [1997] and Tripakis et al. [2005] carries over to the zone-closed simulation graph. Popular timed automata model checkers such as Kronos and Uppaal generate this graph while checking for reachability. Our result implies that these tools can be used not only for reachability, but also to check emptiness of (strongly nonzeno) timed Büchi automata. This can be done with small modifications to the tools, namely, implementing an algorithm to find accepting cycles Courcoubetis et al. [1992] or strongly connected components Tarjan [1972] in a graph. Our result also proves the correctness of (strongly nonzeno) timed Büchi automata emptiness algorithms implemented in the tool Open-Kronos.[3]

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 the *inclusion abstraction* proposed in Daws and Tripakis [1998] or the abstractions proposed in Behrmann et al. [2004], can be used to check timed Büchi automata emptiness.

REFERENCES

ALUR, R., COURCOUBETIS, C., AND DILL, D. 1993. Model checking in dense real time. *Inf. Comput. 104,* 1, 2–34.

ALUR, R. AND DILL, D. 1994. A theory of timed automata. *Theor. Comput. Sci. 126*, 183–235.

BEHRMANN, G., BOUYER, P., LARSEN, K., AND PELÁNEK, R. 2004. Lower and upper bounds in zone based abstractions of timed automata. In *Proceedings of the Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*. Lecture Notes in Computer Science, vol. 2988. Springer.

BENGTSSON, J. AND YI, W. 2003. On clock difference constraints and termination in reachability analysis of timed automata. In *Proceedings of the IEEE International Conference of Formal Engineering Methods (ICFEM'03)*. Lecture Notes in Computer Science, vol. 2885. Springer.

BERTHOMIEU, B. AND MENASCHE, M. 1983. An enumerative approach for analyzing time Petri nets. *IFIP Congress Series 9*, 41–46.

BOUAJJANI, A., TRIPAKIS, S., AND YOVINE, S. 1997. On-the-fly symbolic model checking for real-time systems. In *Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS'97)*. IEEE Computer Society, 25–34.

BOUYER, P. 2004. Forward analysis of updatable timed automata. *Formal Methods Syst. Des. 24,* 3, 281–320.

CLARKE, E., GRUMBERG, O., AND PELED, D. 2000. *Model Checking*. MIT Press.

CLARKE, E. M., EMERSON, E. A., AND SISTLA, A. P. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst. 8,* 2, 244–263.

COURCOUBETIS, C., VARDI, M., WOLPER, P., AND YANNAKAKIS, M. 1992. Memory efficient algorithms for the verification of temporal properties. *Formal Methods Syst. Desi. 1*, 275–288.

DAWS, C., OLIVERO, A., TRIPAKIS, S., AND YOVINE, S. 1996. The Tool KRONOS. In *Hybrid Systems III: Verification and Control*, R. Alur, et.al., Eds. Lecture Notes in Computer Science, vol. 1066. Springer, 208–219.

DAWS, C. AND TRIPAKIS, S. 1998. Model checking of real-time reachability properties using abstractions. In *Proceedings of the 4th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'98)*, B. Steffen, Ed. Lecture Notes in Computer Science, vol. 1384. Springer, 313–329.

DILL, D. 1989. Timing assumptions and verification of finite-state concurrent systems. In *Automatic Verification Methods for Finite State Systems*, J. Sifakis, Ed. Lecture Notes in Computer Science, vol. 407. Springer, 197–212.

HENZINGER, T., NICOLLIN, X., SIFAKIS, J., AND YOVINE, S. 1994. Symbolic model checking for real-time systems. *Inf. Comput. 111,* 2, 193–244.

LAMPORT, L. 1977. Proving the correctness of multiprocess programs. *IEEE Trans. Softw. Eng.*, 125–143.

LARSEN, K., PETTERSON, P., AND YI, W. 1997. Uppaal in a nutshell. *Softw. Tools Technol. Transfer 1,* 1-2.

QUEILLE, J. AND SIFAKIS, J. 1981. Specification and verification of concurrent systems in CESAR. In *Proceedings of the 5th International Symposium on Programming*. Lecture Notes in Computer Science, vol. 137.

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

TARJAN, R. 1972. Depth first search and linear graph algorithms. *SIAM J. Comput. 1,* 2, 146–170.

TRIPAKIS, S. AND COURCOUBETIS, C. 1996. Extending Promela and spin for real time. In *Proceedings of the 2nd International Workshop on Tools and Algorithms for Construction and Analysis of Systems (TACAS'96)*, T. Margaria and B. Steffen, Eds. Lecture Notes in Computer Science, vol. 1055. Springer, 329–348.

TRIPAKIS, S. AND YOVINE, S. 2001. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods Syst. Desi. 18,* 1 , 25–68.

TRIPAKIS, S., YOVINE, S., AND BOUAJJANI, A. 2005. Checking timed Büchi automata emptiness efficiently. *Formal Methods Syst. Desi. 26,* 3, 267–292.

Received June 2006; revised October 2007; accepted February 2008