Form Methods Syst Des (2012) 40:122–146
DOI 10.1007/s10703-011-0133-1

Efficient emptiness check for timed Büchi automata

Frédéric Herbreteau · B. Srivathsan · Igor Walukiewicz

Published online: 24 December 2011

Abstract The Büchi non-emptiness problem for timed automata refers to deciding if a given automaton has an infinite non-Zeno run satisfying the Büchi accepting condition. The standard solution to this problem involves adding an auxiliary clock to take care of the non-Zenoness. In this paper, it is shown that this simple transformation may sometimes result in an exponential blowup. A construction avoiding this blowup is proposed. It is also shown that in many cases, non-Zenoness can be ascertained without an extra construction. An on-the-fly algorithm for the non-emptiness problem, using a non-Zenoness construction only when required, is proposed. Experiments carried out with a prototype implementation of the algorithm are reported.

Keywords Timed automata · Büchi accepting conditions · Non-Zenoness · On-the-fly algorithm

1 Introduction

Timed automata [1] are widely used to model real-time systems. They are obtained from finite automata by adding clocks that can be reset and whose values can be compared with constants. The crucial property of timed automata is that their emptiness is decidable. This model has been implemented in verification tools like Uppaal [4] or Kronos [10], and used in industrial case studies [5, 17, 20].

While most tools concentrate on the reachability problem, questions concerning infinite executions of timed automata are also of interest. In the case of infinite executions one has to eliminate the so-called Zeno runs. These are executions that contain infinitely many

F. Herbreteau (✉) · B. Srivathsan · I. Walukiewicz
LaBRI, UMR 5800, Univ. Bordeaux, CNRS, 33400 Talence, France
e-mail: fh@labri.fr

B. Srivathsan
e-mail: sri@labri.fr

I. Walukiewicz
e-mail: igw@labri.fr

steps taken in a finite time interval. For obvious reasons such executions are considered unrealistic. One way to treat Zeno runs would be to say that a timed automaton admitting such a run is faulty and should be disregarded. This gives rise to the problem of detecting the existence of Zeno runs in an automaton [9, 16, 19]. The other approach to handling Zeno behaviors, that we adopt here, is to say that due to imprecisions introduced by the modeling process one may need to work with automata having Zeno runs. This leads to the problem of this paper: given a timed automaton decide if it has a non-Zeno run passing through accepting states infinitely often. We call this the *Büchi non-emptiness* problem.

This basic problem [1] has been studied already in the paper introducing timed automata. It has been shown that using so-called region abstraction the problem can be reduced to the problem of finding a path in a finite region graph satisfying some particular conditions. The main difference between the cases of finite and infinite executions is that in the latter one needs to decide if the path that has been found corresponds to a non-Zeno run of the automaton.

Subsequent research has shown that the region abstraction is very inefficient for reachability problems. Another method using zones instead of regions has been proposed [14]. It is used at present in all timed-verification tools. While simple at the first sight, the zone abstraction was delicate to get right [7]. This is mainly because the basic properties of regions do not transfer to zones. The zone abstraction also works for infinite executions, but unlike for regions, it is impossible to decide if a path in a zone graph corresponds to a non-Zeno run of the automaton.

There exists a simple solution to the problem of Zeno runs that amounts to transforming automata in such way that every run passing through an accepting state infinitely often is non-Zeno. An automaton with such a property is called *strongly non-Zeno*. The transformation is easy to describe and requires the addition of one new clock. This paper is motivated by our experiments with an implementation of this construction. We have observed that this apparently simple transformation can give a big overhead in the size of a zone graph.

In this paper we closely examine the transformation to strongly non-Zeno automata [25], and show that it can inflict a blowup of the zone graph; and this blowup could even be exponential in the number of clocks. To substantiate, we exhibit an example of an automaton having a zone graph of polynomial size, whose transformed version has a zone graph of exponential size. We propose another solution to avoid this phenomenon. Instead of modifying the automaton, we modify the zone graph. We show that this modification allows us to detect if a path in the zone graph can be instantiated to a non-Zeno run. Moreover the size of the modified graph is $|ZG(\mathscr{A})| \cdot \mathscr{O}(|X|)$, where $|ZG(\mathscr{A})|$ is the size of the zone graph and $|X|$ is the number of clocks.

In the second part of the paper we propose an on-the-fly algorithm for testing the existence of accepting non-Zeno runs in timed Büchi automata. The problem we face highly resembles the emptiness testing of finite automata with generalized Büchi conditions. Since the most efficient solutions for the latter problem are based on Tarjan's algorithm to detect strongly-connected-components (SCCs) [15, 22], we take the same route here. We additionally observe that Büchi emptiness can sometimes be decided directly from the zone graph. This permits to restrict the use of the modified zone graph construction only to certain parts of the zone graph. In cases when no clock comparisons of the form $x = 0$ are reachable from the initial state of the automaton, the algorithm runs in time $\mathscr{O}(|ZG(\mathscr{A})| \cdot |X|)$. Further, the optimized algorithm runs in time $\mathscr{O}(|ZG(\mathscr{A})|)$ when no reachable SCC contains a blocking clock: that is, a clock that is bounded (e.g. $x \leq 1$) but never reset in the SCC. We also give additional optimizations that prove to be powerful in practice. We include experiments conducted on examples in the literature.

1.1 Related work

The zone approach has been introduced in the Petri net context [6], and then adapted to the framework of timed automata [14]. The advantage of zones over regions is that they do not require to consider every possible unit time interval separately. The delicate point about zones was to find a right approximation operator. Usual approximation operators are sound and complete: each path in the zone graph can be instantiated as a run in the automaton and vice-versa. While this is enough for correctness of the reachability algorithm, it does not allow however to determine if a path can be instantiated to a non-Zeno run. The solution involving adding one clock has been discussed in [2, 23, 25]. Recently, Tripakis [24] has shown a way to extract an accepting run from a zone graph of the automaton. Combined with the construction of adding one clock this gives a solution to the Büchi emptiness problem. Since, as we show here, adding one clock may be costly, this solution is costly too. A different approach has been considered in [9, 16] where some sufficient conditions are proposed for a timed automaton to be free from Zeno runs. Notice that for obvious complexity reasons, any such condition must be either not complete, or of the same algorithmic complexity as the emptiness test itself.

1.2 Organization of the paper

In the next section we formalize our problem, and discuss region and zone abstractions. As an intermediate step we give a short proof of the above mentioned result from [24]. Section 3 explains the problems with the transformation to strongly non-Zeno automata, and describes our alternative method. The following section is devoted to a description of the algorithm. We conclude with the results of the experiments performed.

2 The emptiness problem for timed Büchi automata

2.1 Timed Büchi automata

Let X be a set of clocks, i.e., variables that range over $\mathbb{R}_{\geq 0}$, the set of non-negative real numbers. *Clock constraints* are conjunctions of comparisons of variables with integer constants: $x\#c$ where $x \in X$ is a clock, $c \in \mathbb{N}$ and $\# \in \{<, \leq, =, \geq, >\}$. For instance $(x \leq 3 \wedge y > 0)$ is a clock constraint. Let $\Phi(X)$ denote the set of clock constraints over clock variables X.

A *clock valuation* over X is a function $\nu : X \to \mathbb{R}_{\geq 0}$. We denote $\mathbb{R}^X_{\geq 0}$ for the set of clock valuations over X, and $\mathbf{0} : X \to \{0\}$ for the valuation that associates 0 to every clock in X. We write $\nu \models \phi$ when ν satisfies ϕ, i.e. when every constraint in ϕ holds after replacing every x by $\nu(x)$.

For a valuation ν and $\delta \in \mathbb{R}_{\geq 0}$, let $(\nu + \delta)$ be the valuation such that $(\nu + \delta)(x) = \nu(x) + \delta$ for all $x \in X$. For a set $R \subseteq X$, let $[R]\nu$ be the valuation such that $([R]\nu)(x) = 0$ if $x \in R$ and $([R]\nu)(x) = \nu(x)$ otherwise.

A *Timed Büchi Automaton (TBA)* is a tuple $\mathscr{A} = (Q, q_0, X, T, Acc)$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, X is a finite set of clocks, $Acc \subseteq Q$ is a set of accepting states, and $T \subseteq Q \times \Phi(X) \times 2^X \times Q$ is a finite set of transitions (q, g, R, q') where g is a *guard*, and R is a *reset* of the transition.

A *configuration* of $\mathscr{A}$ is a pair $(q, \nu) \in Q \times \mathbb{R}^X_{\geq 0}$; with $(q_0, \mathbf{0})$ being the *initial configuration*. A *transition* $(q, \nu) \xrightarrow{\delta,t} (q', \nu')$ for $t = (q, g, R, q') \in T$ and $\delta \in \mathbb{R}_{\geq 0}$ is defined when $\nu + \delta \models g$ and $\nu' = [R](\nu + \delta)$.

A *run* of $\mathscr{A}$ is an infinite sequence of configurations connected by transitions, starting from the initial state q_0 and the initial valuation $\nu_0 = \mathbf{0}$:

$$(q_0, \nu_0) \xrightarrow{\delta_0, t_0} (q_1, \nu_1) \xrightarrow{\delta_1, t_1} \cdots$$

A run σ *satisfies the Büchi condition* if it visits *accepting configurations* infinitely often, that is configurations with a state from Acc. The *duration* of the run is the accumulated delay: $\sum_{i\geq 0} \delta_i$. An infinite run σ is *Zeno* if it has a finite duration.

Definition 1 The *Büchi non-emptiness problem* is to decide if $\mathscr{A}$ has a non-Zeno run satisfying the Büchi condition.

The Büchi non-emptiness problem is known to be PSPACE-complete [1].

The class of TBA we consider is usually known as diagonal-free TBA since clock comparisons like $x - y \leq 1$ are disallowed. Since we are interested in the Büchi non-emptiness problem, we can consider automata without an input alphabet and without invariants since they can be simulated by guards.

2.2 Regions and region graphs

A simple decision procedure for the Büchi non-emptiness problem builds from $\mathscr{A}$ a graph called the *region graph* and tests if there is a path in this graph satisfying certain conditions. We will define two types of regions.

Fix a constant M and a finite set of clocks X. Two valuations $\nu, \nu' \in \mathbb{R}_{\geq 0}^X$ are *region equivalent* w.r.t. M, denoted $\nu \sim_M \nu'$ iff for every $x, y \in X$:

1. $\nu(x) > M$ iff $\nu'(x) > M$;
2. if $\nu(x) \leq M$, then $\lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$;
3. if $\nu(x) \leq M$, then $\{\nu(x)\} = 0$ iff $\{\nu'(x)\} = 0$;
4. if $\nu(x) \leq M$ and $\nu(y) \leq M$ then $\{\nu(x)\} \leq \{\nu(y)\}$ iff $\{\nu'(x)\} \leq \{\nu'(y)\}$.

The first three conditions ensure that the two valuations satisfy the same guards as clock constraints are defined with respect to integer bounds and M is the maximal constant in $\mathscr{A}$. The last one enforces that for every $\delta \in \mathbb{R}_{\geq 0}$ there is $\delta' \in \mathbb{R}_{\geq 0}$, such that valuations $\nu + \delta$ and $\nu' + \delta'$ satisfy the same guards since the difference of x and y is invariant by time elapse.

We will also define *diagonal region equivalence (d-region equivalence* for short) that strengthens the last condition to

4^d. for every integer $c \in (-M, M)$: $\nu(x) - \nu(y) \leq c$ iff $\nu'(x) - \nu'(y) \leq c$.

This region equivalence is denoted by $\sim_M^d$. Observe that it is finer than $\sim_M$.

A *region* is an equivalence class of $\sim_M$. We write $[\nu]_{\sim_M}$ for the region of ν, and $\mathscr{R}_M$ for the set of all regions with respect to M. Similarly, for d-region equivalence we write: $[\nu]^d_{\sim_M}$ and $\mathscr{R}^d_M$. If r is a region or a d-region then we will write $r \vDash g$ to mean that every valuation in r satisfies the guard g. Observe that all valuations in a region, or a d-region, satisfy the same guards.

For an automaton $\mathscr{A}$, we define its *region graph*, $RG(\mathscr{A})$, using the $\sim_M$ relation, where M is the biggest constant appearing in the guards of its transitions. Without loss of generality we assume that $M \geq 0$, in other words there is at least one guard in $\mathscr{A}$. Nodes of $RG(\mathscr{A})$ are of the form (q, r) for q a state of $\mathscr{A}$ and $r \in \mathscr{R}_M$ a region. There is a transition $(q, r) \xrightarrow{t} (q', r')$ if there are $\nu \in r$, $\delta \in \mathbb{R}_{\geq 0}$ and $\nu' \in r'$ with $(q, \nu) \xrightarrow{\delta, t} (q', \nu')$. Observe that a transition in the region graph is not decorated with a delay. The graph $RG^d(\mathscr{A})$ is defined similarly but using the $\sim^d_M$ relation.

It will be important to understand the properties of pre- and post-stability of regions or d-regions [25]. We state them formally. A transition $(q,r) \xrightarrow{t} (q',r')$ in a region graph or a d-region graph is:

- *Pre-stable* if for every $\nu \in r$ there are $\nu' \in r'$, $\delta \in \mathbb{R}_{\geq 0}$ s.t. $(q,\nu) \xrightarrow{\delta,t} (q',\nu')$.
- *Post-stable* if for every $\nu' \in r'$ there are $\nu \in r$, $\delta \in \mathbb{R}_{\geq 0}$ s.t. $(q,\nu) \xrightarrow{\delta,t} (q',\nu')$.

The following lemma explains our interest in $\sim^d_M$ relation. The main fact is that both region graphs are pre-stable and this allows to decide the existence of a non-Zeno run easily by Theorem 1.

Lemma 1 (Pre and post-stability [8]) *Transitions in $RG^d(\mathscr{A})$ are pre-stable and post-stable. Transitions in $RG(\mathscr{A})$ are pre-stable but not necessarily post-stable.*

Consider two sequences

$$(q_0,\nu_0) \xrightarrow{\delta_0,t_0} (q_1,\nu_1) \xrightarrow{\delta_1,t_1} \cdots \tag{1}$$

$$(q_0,r_0) \xrightarrow{t_0} (q_1,r_1) \xrightarrow{t_1} \cdots \tag{2}$$

where the first is a run in $\mathscr{A}$, and the second is a path in $RG(\mathscr{A})$ or $RG^d(\mathscr{A})$. We say that the first is an *instantiation* of the second if $\nu_i \in r_i$ for all $i \geq 0$. Equivalently, we say that the second is an *abstraction* of the first. The following lemma is a direct consequence of the pre-stability property.

Lemma 2 *Every path in $RG(\mathscr{A})$ is an abstraction of a run of $\mathscr{A}$, and conversely, every run of $\mathscr{A}$ is an instantiation of a path in $RG(\mathscr{A})$. Similarly for $RG^d(\mathscr{A})$.*

This lemma allows us to relate the existence of an accepting run of $\mathscr{A}$ to the existence of paths with special properties in $RG(\mathscr{A})$ or $RG^d(\mathscr{A})$. We say that a path as in (2) *satisfies the Büchi condition* if it has infinitely many occurrences of states from *Acc*. The path is called *progressive* [1, 25] if for every clock $x \in X$:

- either x is almost always above M: there is n with $r_i \vDash x > M$ for all $i > n$;
- or x is reset infinitely often and strictly positive infinitely often: for every n there are $i, j > n$ such that $r_i \vDash (x = 0)$ and $r_j \vDash (x > 0)$.

Theorem 1 ([1]) *A TBA $\mathscr{A}$ has a non-Zeno run satisfying the Büchi conditions iff $RG(\mathscr{A})$ has a progressive path satisfying the Büchi condition. Similarly for $RG^d(\mathscr{A})$.*

The progress criterion above can be encoded adding an extra Büchi accepting condition [1, 25]. While theorem 1 gives an algorithm for solving our problem, it turns out that this method is very impractical. The number of regions for clocks X and constant M turns out to be $\mathscr{O}(|X|!.2^{|X|}M^{|X|})$ [1] and constructing all of them, or even searching through them on-the-fly, has proved to be very costly.

2.3 Zones and zone graphs

Timed verification tools use zones instead of regions. A zone is a set of valuations defined by a conjunction of two kinds of constraints: comparison of the difference between two clocks with a constant like $x - y\#c$, or comparison of the value of a single clock with a constant like $x\#c$ for $x \in X$, $c \in \mathbb{N}$ and $\# \in \{<, \leq, =, \geq, >\}$. For example $(x - y \geq 1) \wedge (y < 2)$ is a zone. While at first sight it may seem that there are more zones than regions, this is not the case if we count only those that are reachable from the initial valuation.

Since zones are sets of valuations defined by constraints, one can define transitions directly on zones. For a transition t in $\mathscr{A}$ and a zone Z, we have $(q, Z) \xrightarrow{t} (q', Z')$ if Z' is the set of valuations ν' such that there exists $\nu \in Z$ and $\delta \in \mathbb{R}_{\geq 0}$ and $(q, \nu) \xrightarrow{\delta,t} (q', \nu')$. It is well-known that Z' is a zone. Moreover zones can be represented using Difference Bound Matrices (DBMs), and transitions can be computed efficiently on DBMs [14]. The problem is that the number of reachable zones is not guaranteed to be finite [13].

In order to ensure that the number of reachable zones is finite, one introduces abstraction operators. We mention the three most common ones in the literature. They refer to region graphs, $RG(\mathscr{A})$ or $RG^d(\mathscr{A})$, and use the constant M that is the maximal constant appearing in the guards of $\mathscr{A}$.

- $\mathit{Closure}_M(Z)$: the smallest union of regions containing Z;
- $\mathit{Closure}^d_M(Z)$: similarly but for d-regions;
- $\mathit{Approx}_M(Z)$: the smallest union of d-regions that is convex and that contains Z.

The following lemma establishes the links between the three abstraction operators, and is very useful to transpose reachability results from one abstraction to the other.

Lemma 3 ([8]) *For every zone Z: $Z \subseteq \mathit{Closure}^d_M(Z) \subseteq \mathit{Approx}_M(Z) \subseteq \mathit{Closure}_M(Z)$.*

Similar to region graphs, we define simulation graphs where after every transition a specific approximation operation is used. So we have three graphs corresponding to the three approximation operations above. Notice that $\mathit{Closure}_M(Z)$ and $\mathit{Closure}^d_M(Z)$ may not be convex, hence they may not be zones [8].

Take an automaton $\mathscr{A}$ and let M be the biggest constant that appears in the guards of its transitions. The simulation graph $SG(\mathscr{A})$ has nodes of the form (q, S) where q is a state of $\mathscr{A}$ and S is a set of valuations. The initial node is $(q_0, \{\mathbf{0}\})$. There is a transition $(q, S) \xrightarrow{t} (q', \mathit{Closure}_M(S'))$ in $SG(\mathscr{A})$ if S' is the set of valuations ν' such that $(q, \nu) \xrightarrow{\delta,t} (q', \nu')$ for some $\nu \in S$ and $\delta \in \mathbb{R}_{\geq 0}$. Similarly, we define $SG^d(\mathscr{A})$ and $SG^a(\mathscr{A})$ by replacing $\mathit{Closure}_M$ with $\mathit{Closure}^d_M$ and Approx_M respectively. Observe that for every node (q, S) that is reachable in one of the three graphs above, S is a union of regions or d-regions. The notions of an abstraction of a run of $\mathscr{A}$, and an instantiation of a path in the simulation graph, are defined in the same way as that of region graphs.

Tools like Kronos or Uppaal use the Approx_M abstraction. The two other abstractions are less interesting for implementations since the result may not be convex. Nevertheless, they are useful in proofs. The following Lemma (cf. [13]) says that transitions in $SG(\mathscr{A})$ are post-stable with respect to regions.

Lemma 4 *Let $(q, S) \xrightarrow{t} (q', S')$ be a transition in $SG(\mathscr{A})$ such that both S and S' are unions of regions. For every region $r' \subseteq S'$, there is a region $r \subseteq S$ such that $(q, r) \xrightarrow{t} (q', r')$ is a transition in $RG(\mathscr{A})$.*

Proof Take a transition $(q,S) \xrightarrow{t} (q',S')$ and let us examine what it means. By definition, $S' = \mathit{Closure}_M(S'')$ where S'' is the set of valuations ν'' that satisfy $(q,\nu) \xrightarrow{\delta,t} (q',\nu'')$ for some $\nu \in S$ and $\delta \in \mathbb{R}_{\geq 0}$. Consider $r' \subseteq S'$; the intersection $r' \cap S''$ is not empty. Take $\nu' \in r' \cap S''$, and let $\nu \in S$ be a valuation such that $(q,\nu) \xrightarrow{\delta,t} (q',\nu')$ for some $\delta \in \mathbb{R}_{\geq 0}$. Let r be the region of ν. We have that $r \cap S$ is not empty, hence $r \subseteq S$ as S is a union of regions. By definition $(q,r) \xrightarrow{t} (q',r')$ is a transition in $RG(\mathscr{A})$. $\square$

We get a correspondence between paths in simulation graphs and runs of $\mathscr{A}$.

Theorem 2 ([24]) *Every path in $SG(\mathscr{A})$ is an abstraction of a run of $\mathscr{A}$, and conversely, every run of $\mathscr{A}$ is an instantiation of a path in $SG(\mathscr{A})$. Similarly for SG^d and SG^a.*

Proof We first show that a path in $SG(\mathscr{A})$ is an abstraction of a run of $\mathscr{A}$. Take a path $(q_0,S_0) \xrightarrow{t_0} (q_1,S_1) \xrightarrow{t_1} \cdots$ in $SG(\mathscr{A})$. Construct a DAG with nodes (i,q_i,r_i) such that r_i is a region in S_i. We put an edge from (i,q_i,r_i) to $(i+1,q_{i+1},r_{i+1})$ if $(q_i,r_i) \xrightarrow{t_i} (q_{i+1},r_{i+1})$. By Lemma 4, every node in this DAG has at least one predecessor, and the branching of every node is bounded by the number of regions. Hence, this DAG has an infinite path that is a path in $RG(\mathscr{A})$. By Lemma 2 this path can be instantiated to a run of $\mathscr{A}$.

To conclude the proof one can easily verify that a run of $\mathscr{A}$ can be abstracted to a path in $SG^d(\mathscr{A})$. Then using Lemma 3 this path can be converted to a path in $SG^a(\mathscr{A})$, and later to one in $SG(\mathscr{A})$. $\square$

Observe that Theorem 2 does not guarantee that a path we find in a simulation graph has an instantiation that is non-Zeno. This cannot be decided from $SG(\mathscr{A})$ by using the progress criterion defined in page 126 as we show now. Consider for instance the automaton $\mathscr{A}_2$ in Fig. 9 which has only Zeno runs as both x and y must remain equal to 0 on every run. Figure 1 shows a part of $RG(\mathscr{A}_2)$. The infinite path starting from node $(0, 0=x=y)$ is not progressive as none of the clocks can have a positive value. Moreover, it can be seen that every node where a clock has a positive value is a deadlock node. Figure 2 depicts the corresponding part of $SG(\mathscr{A}_2)$. This path satisfies the progress criterion as both x and y are reset and may have positive values infinitely often, despite all its instantiations being Zeno. The progress criterion fails due to the loss of pre-stability in $SG(\mathscr{A}_2)$: none of the valuations with either $x>0$ or $y>0$ have a successor. In Sect. 3, we show how to avoid this problem.

In the subsequent sections, we are interested only in the simulation graph $SG^a(\mathscr{A})$. Observe that the symbolic zone obtained by the approximation of a zone using Approx_M is in fact a zone. Hence, we prefer to call it a zone graph and denote it as $ZG^a(\mathscr{A})$. Every node of $ZG^a(\mathscr{A})$ is of the form (q,Z) where Z is a zone.

$$
\begin{tikzcd}
 & (1,0=x<y) & & (2,0=y<x) & & \\
(0,0=x=y) \arrow[r] \arrow[ur] & (1,0=x=y) \arrow[r] & (0,0=x=y) \arrow[r] \arrow[ur] & (2,0=y=x) \arrow[r] & (0,0=x=y) & \cdots
\end{tikzcd}
$$

Fig. 1 A part of the region graph for the automaton $\mathscr{A}_2$ in Fig. 9

$$(0,0=x=y) \longrightarrow (1,0=x\leq y) \longrightarrow (0,0=x=y) \longrightarrow (2,0=y\leq x) \longrightarrow (0,0=x=y) \cdots$$

Fig. 2 A part of the symbolic graph for the automaton $\mathscr{A}_2$ in Fig. 9

3 Finding non-Zeno paths

As we have remarked above, in order to use Theorem 2 we need to be sure that a path we get can be instantiated to a non-Zeno run. We discuss the solutions proposed in the literature, and then offer a better one. Thanks to pre-stability of the region graph, the progress criterion on regions has been defined in [1] for selecting runs from $RG(\mathscr{A})$ that have a non-Zeno instantiation (see Sect. 2.2). Notice that the semantics of TBA in [1] constrains all delays δ_i to be strictly positive, but the progress criterion can be extended to the stronger semantics that is used nowadays (see [25] for instance). However, since zone graphs are not pre-stable, this method cannot be directly extended to zone graphs.

3.1 Adding one clock

A common solution to deal with Zeno runs is to transform an automaton into a *strongly non-Zeno automaton*, i.e. such that all runs satisfying the Büchi condition are guaranteed to be non-Zeno. We present this solution here and discuss why, although simple, it may add an exponential factor in the decision procedure.

The main idea behind the transformation of $\mathscr{A}$ into a strongly non-Zeno automaton $SNZ(\mathscr{A})$ is to ensure that on every accepting run, time elapses for 1 time unit infinitely often. Hence, it is sufficient to check for the existence of an accepting run as it is non-Zeno for granted. Consider the automaton $\mathscr{V}_k$ and its transformation into $\mathscr{W}_k = SNZ(\mathscr{V}_k)$ in Fig. 3. The transformation adds one clock z and duplicates accepting states (e.g. a^k in $\mathscr{V}_k$). One copy is no longer accepting whereas the other is accepting, but it can be reached only when $z \geq 1$ (these are respectively a_2^k and a_1^k in $\mathscr{W}_k$). Moreover, when an accepting state is reached z is reset to 0. As a result, every accepting run in $\mathscr{V}_k$ has a corresponding run in $\mathscr{W}_k$ where every occurrence of a^k is replaced by an occurrence of either a_1^k or a_2^k. Since two occurrences of the accepting state a_1^k have to be separated by at least one time unit, an accepting run in $\mathscr{W}_k$ is necessarily non-Zeno.

A slightly different construction is mentioned in [2]. Of course one can also have other modifications, and it is impossible to treat all the imaginable constructions at once. Our objective here is to show that the constructions proposed in the literature produce a phenomenon causing proliferation of zones that can sometimes be exponential in the number of clocks. The discussion below will focus on the construction from [25], but the one from [2] suffers from the same problem.

The problem comes from the fact that the constraint $z \geq 1$ may be a source of rapid multiplication of the number of zones in the zone graph of $SNZ(\mathscr{A})$. Consider $\mathscr{V}_k$ and $\mathscr{W}_k$ from Fig. 3 and let us say that $k = 2$. Starting at the state b^2 of $\mathscr{V}_2$ in the zone $0 \leq y \leq x_1 \leq$

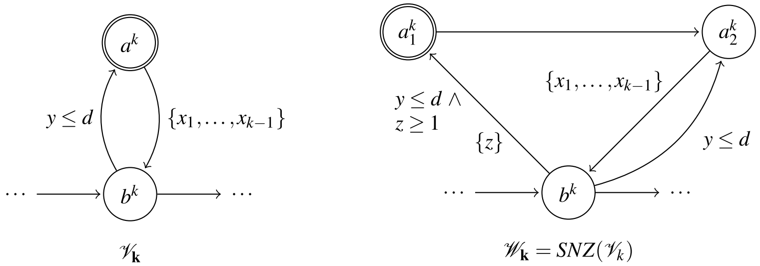


Fig. 3 The gadgets $\mathscr{V}_k$ (*left*) and $\mathscr{W}_k = SNZ(\mathscr{V}_k)$ (*right*)

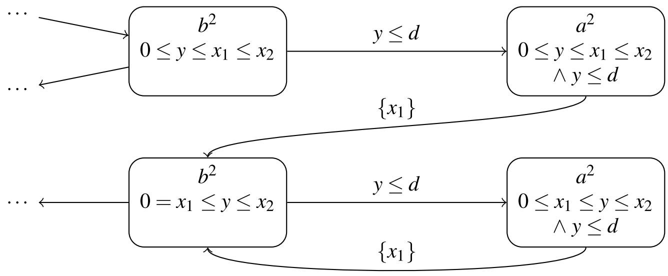


Fig. 4 Part of $ZG(\mathscr{V}_2)$

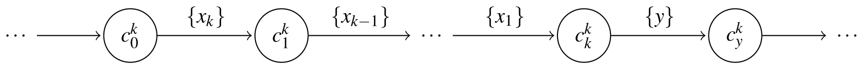


Fig. 5 The gadget $\mathscr{R}_k$

x_2, there are two reachable zones with state b^2. This is depicted in Fig. 4 where after two traversals of the cycle formed by b^2 and a^2, we reach a zone that is invariant for the cycle. Moreover, from the two zones with state b^2 in Fig. 4, reseting x_1 followed by y as $\mathscr{R}_1$ (in Fig. 5) does, we reach the same zone $0 \leq y \leq x_1 \leq x_2$.

In contrast starting in b^2 of $\mathscr{W}_2 = SNZ(\mathscr{V}_2)$ from $0 \leq y \leq x_1 \leq x_2 \leq z$ gives at least d zones. The part of $ZG(\mathscr{W}_2)$ in Fig. 6 gives the sequence of transitions in the zone graph of $\mathscr{W}_2$ starting from the zone $(b^2, 0 \leq y \leq x_1 \leq x_2 \leq z)$ by successive iterations of the cycle that goes through b^2, a_1^2 and a_2^2. After a certain point, every traversal induces an extra distance between the clocks y and z. Clearly, there are at least d zones in this case. Resetting x_1 followed by y as $\mathscr{R}_1$ (in Fig. 5) does still yield d zones.

We now exploit this fact to give an example of a TBA $\mathscr{A}_n$ whose zone graph has a number of zones linear in the number of clocks, but $\mathscr{B}_n = SNZ(\mathscr{A}_n)$ has a zone graph of size exponential in the number of clocks.

$\mathscr{A}_n$, in Fig. 7, is constructed from the automata gadgets $\mathscr{V}_k$ and $\mathscr{R}_k$ as shown in Figs. 3 and 5. Observe that the role of $\mathscr{R}_k$ is to enforce an order $0 \leq y \leq x_1 \leq \cdots \leq x_k$ between clock values. By induction on k one can compute that there are only two zones at locations b^k since $\mathscr{R}_{k+1}$ made the two zones in b^{k+1} collapse into the same zone in b^k. Hence the number of nodes in the zone graph of $\mathscr{A}_n$ is $\mathscr{O}(n)$.

Let us now consider $\mathscr{B}_n$, the strongly non-Zeno automaton obtained from $\mathscr{A}_n$ following [25]. Every gadget $\mathscr{V}_k$ gets transformed to $\mathscr{W}_k$ as shown in Fig. 7. While exploring $\mathscr{W}_k$, one introduces a distance between the clocks x_{k-1} and x_k. So when leaving it one gets zones with $x_k - x_{k-1} \geq c$, where $c \in \{0, 1, 2, \ldots, d\}$. The distance between x_k and x_{k-1} is preserved by $\mathscr{R}_k$. In consequence, $\mathscr{W}_n$ produces at least $d+1$ zones. For each of these zones $\mathscr{W}_{n-1}$ produces $d+1$ more zones. In the end, the zone graph of $\mathscr{B}_n$ has at least $(d+1)^{n-1}$ zones at the state b^2. The zones obtained with the state b^k are of the form

$$0 = x_1 = \cdots = x_{k-1} \leq z \leq y \leq x_k \leq \cdots \leq x_n$$
$$\wedge \bigwedge_{i \in \{k,\ldots,n-1\}} x_{i+1} - x_i \geq c_i \quad \text{where each } c_i \in \{0, 1, \ldots, d\}$$

So the zone graph has at least $(d+1)^{n-k+1}$ zones at state b^k. Hence, the zone graph of $\mathscr{B}_n$ contains at least $(d+1)^{n-1}$ zones.

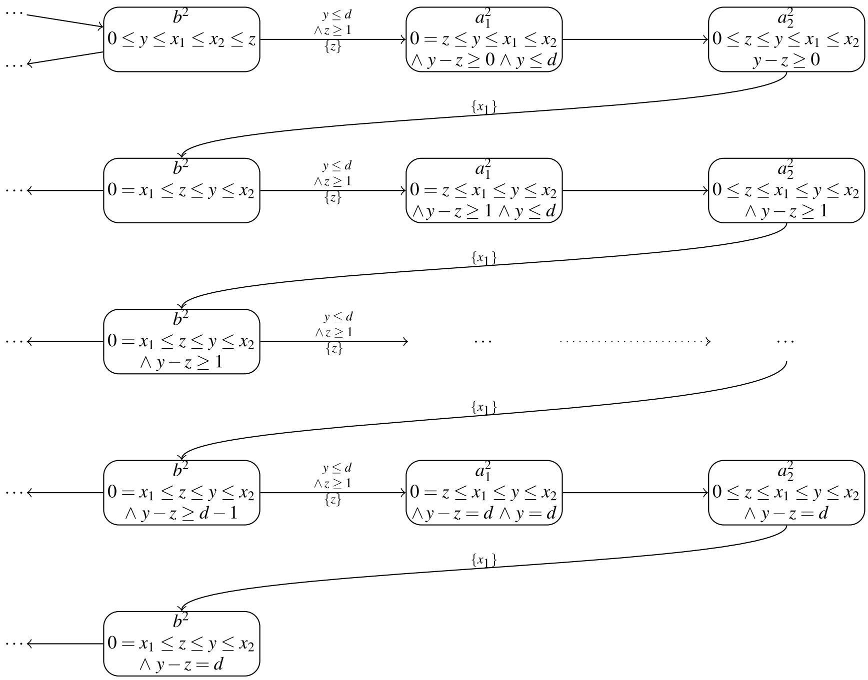


Fig. 6 Part of $ZG(\mathscr{W}_2)$

Fig. 7 Automata $\mathscr{A}_n$ (*left*) and $\mathscr{B}_n = SNZ(\mathscr{A}_n)$ (*right*)

We have thus shown that $\mathscr{A}_n$ has $\mathscr{O}(n)$ zones while $\mathscr{B}_n = SNZ(\mathscr{A}_n)$ has an exponential number of zones even when the constant d is 1. One could argue that the transformation in [25] can be transformed in such a way to prevent the combinatorial explosion. In particular, it is often suggested to replace $z \geq 1$ by a guard that matches the biggest constant in the automaton, that is $z \geq d$ in our case. However, this would still yield an exponential blowup as every zone with state b^k yields two different zones with state b^{k-1} that do not collapse going through $\mathscr{R}_{k-1}$. Observe also that the construction shows that even with two clocks the number of zones blows exponentially in the binary representation of d. Note that the automaton $\mathscr{A}_n$ does not have a non-Zeno accepting run. Hence, every search algorithm is compelled to explore all the zones of $\mathscr{B}_n$.

3.2 A more efficient solution

We aim to decide if a given path in a zone graph has a non-Zeno instantiation. This is equivalent to deciding if all instantiations of a path are Zeno. There are essentially two reasons for this:

– there may be a clock x that is reset finitely many times but bound infinitely many times by guards $x \leq c$:

$$\bullet \xrightarrow{x\leq 1} \bullet \xrightarrow{\{x\}} \bullet \rightarrow \cdots \underbrace{\rightarrow \bullet \xrightarrow{x\leq 2} \bullet \rightarrow \bullet \xrightarrow{x\leq 1} \bullet \rightarrow \cdots}_{\text{suffix with no reset of } x}$$

– or time may not be able to elapse at all due to infinitely many transitions that check $x = 0$, forcing x to stay at 0:

$$\bullet \xrightarrow{\{y\}} \bullet \xrightarrow{x=0} \bullet \xrightarrow{\{x\}} \bullet \xrightarrow{y=0} \bullet \xrightarrow{\{y\}} \bullet \xrightarrow{x=0} \cdots$$

Our solution stems from a realization that we only need one non-Zeno run satisfying the Büchi condition and so in a way transforming an automaton to strongly non-Zeno is excessive. We propose not to modify the automaton, but to introduce additional information to the zone graph $ZG^a(\mathscr{A})$. The nodes will now be triples (q, Z, Y) where $Y \subseteq X$ is the set of clocks that can potentially be equal to 0. It means in particular that other clock variables, i.e. those from $X - Y$ are assumed to be bigger than 0. We write $(X - Y) > 0$ for the constraint saying that all the variables in $X - Y$ are not 0.

Definition 2 Let $\mathscr{A}$ be a TBA over a set of clocks X. The *guessing zone graph* $GZG^a(\mathscr{A})$ has nodes of the form (q, Z, Y) where (q, Z) is a node in $ZG^a(\mathscr{A})$ and $Y \subseteq X$. The initial node is (q_0, Z_0, X), with (q_0, Z_0) the initial node of $ZG^a(\mathscr{A})$. In $GZG^a(\mathscr{A})$ there are transitions:

– $(q, Z, Y) \xrightarrow{t} (q', Z', Y \cup R)$ if there is a transition $(q, Z) \xrightarrow{t} (q', Z')$ in $ZG^a(\mathscr{A})$ with $t = (q, g, R, q')$, and there are valuations $\nu \in Z$, $\nu' \in Z'$, and $\delta \in \mathbb{R}_{\geq 0}$ such that $\nu + \delta \vDash (X - Y) > 0$ and $(q, \nu) \xrightarrow{\delta, t} (q, \nu')$;
– $(q, Z, Y) \xrightarrow{\tau} (q, Z, Y')$, on a new auxiliary letter τ, for $Y' = \emptyset$ or $Y' = Y$.

The additional component Y expresses some information about possible valuations with which we can take a transition. The first case is about transitions that are realizable when clocks outside Y are positive. While it is formulated in a more general way, one can think of this transition as being instantaneous: $\delta = 0$. Then we have the second kind of transitions, namely the transitions on τ, that allow us to nondeterministically guess when time can pass.

It will be useful to distinguish some types of transitions and nodes of $GZG^a(\mathscr{A})$.

Definition 3 We call a transition of $GZG^a(\mathscr{A})$ a *zero-check* when some clock is forced to be equal to 0 by the guard g of the transition; formally, for some clock x, for all $\nu \in Z$, and all $\delta \in \mathbb{R}_{\geq 0}$ such that $\nu + \delta \vDash g$ we have $(\nu + \delta)(x) = 0$.

The role of Y sets will become obvious in the construction below. In short, from a node $(q, Z, \emptyset)$, that is with $Y = \emptyset$, every reachable zero-check will be preceded by the reset of the variable that is checked, and hence nothing prevents a time elapse in this node. We will be particularly interested in the following types of nodes to find non-Zeno accepting runs.

Definition 4 A node (q, Z, Y) of $GZG^a(\mathscr{A})$ is *clear* if the third component is empty: $Y = \emptyset$. A node is A node is *accepting* if q is an accepting state.

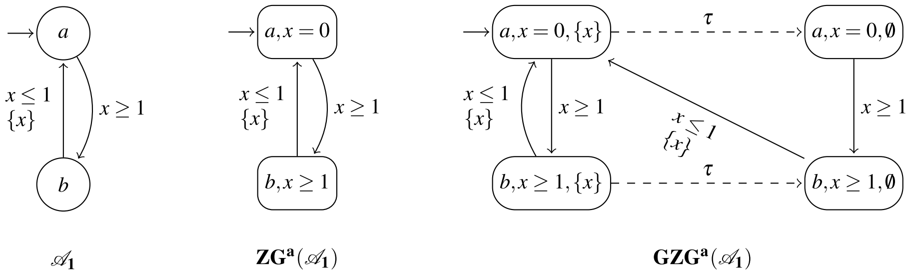


Fig. 8 A TBA $\mathscr{A}_1$ and the guessing zone graph $GZG^a(\mathscr{A}_1)$ (with τ self-loops omitted for clarity)

Example 1 Figure 8 depicts a TBA $\mathscr{A}_1$ along with its zone graph $ZG^a(\mathscr{A}_1)$ and its guessing zone graph $GZG^a(\mathscr{A}_1)$ where τ-loops have been omitted.

The guessing zone graph construction can be optimized by restricting the guessed sets to clocks that are indeed equal to zero in some valuation in the zone. For instance, from the node $(b, x \geq 1, \{x\})$ in Fig. 8, x cannot be checked for zero unless it is first reset. Hence, this node can safely be removed from $GZG^a(\mathscr{A}_1)$, yielding a smaller graph. In the resulting graph, the only loop goes through a τ transition. This emphasizes that time must elapse from node $(a, x = 0, \{x\})$ in order to take a transition with guard $x \geq 1$. An optimized guessing zone graph construction is given in [19].

Notice that directly from the definition it follows that a path in $GZG^a(\mathscr{A})$ determines a path in $ZG^a(\mathscr{A})$ obtained by removing τ transitions and the third component from nodes.

In order to state the main theorem succinctly we need some notions.

Definition 5 A variable x is *bounded* by a transition of $GZG^a(\mathscr{A})$ if the guard of the transition implies $x \leq c$ for some constant c. More precisely: x is bounded by the transition $(q, Z, Y) \xrightarrow{(q,g,R,q')} (q', Z', Y')$, if for all $\nu \in Z$ and $\delta \in \mathbb{R}_{\geq 0}$ such that $\nu + \delta \vDash g$, we have $(\nu + \delta)(x) \leq c$ for some $c \in \mathbb{N}$. A variable is *reset* by the transition if it belongs to the reset set R of the transition.

Definition 6 We say that a path is *blocked* if there is a variable that is bounded infinitely often and reset only finitely often by the transitions on the path. Otherwise the path is called *unblocked*.

Obviously, paths corresponding to non-Zeno runs are unblocked.

Theorem 3 *A TBA $\mathscr{A}$ has a non-Zeno run satisfying the Büchi condition iff there exists an unblocked path in $GZG^a(\mathscr{A})$ visiting both an accepting node and a clear node infinitely often.*

The proof of Theorem 3 follows from Lemmas 5 and 6 below. It is in Lemma 6 that the third component of states is used.

At the beginning of the section we had recalled that the progress criterion [1] stated in page 126 characterizes the paths in region graphs that have non-Zeno instantiations. We had mentioned that it cannot be directly extended to zone graphs since their transitions are not pre-stable. Lemma 6 below shows that by slightly complicating the zone graph we can recover a result very similar to Lemma 4.13 in [1].

Lemma 5 *If $\mathscr{A}$ has a non-Zeno run satisfying the Büchi condition, then in $GZG^a(\mathscr{A})$ there is an unblocked path visiting both an accepting node and a clear node infinitely often.*

Proof Let ρ be a non-Zeno run of $\mathscr{A}$:

$$(q_0, \nu_0) \xrightarrow{\delta_0, t_0} (q_1, \nu_1) \xrightarrow{\delta_1, t_1} \cdots$$

By Theorem 2, it is a concretization of a path σ in $ZG^a(\mathscr{A})$:

$$(q_0, Z_0) \xrightarrow{t_0} (q_1, Z_1) \xrightarrow{t_1} \cdots$$

Let σ' be the following sequence:

$$(q_0, Z_0, Y_0) \xrightarrow{\tau} (q_0, Z_0, Y_0') \xrightarrow{t_0} (q_1, Z_1, Y_1) \xrightarrow{\tau} (q_1, Z_1, Y_1') \xrightarrow{t_1} \cdots$$

where $Y_0 = X$, Y_i is determined by the transition, and $Y_i' = Y_i$ unless $\delta_i > 0$ when we put $Y_i' = \emptyset$. We need to see that this is indeed a path in $GZG^a(\mathscr{A})$. For this we need to see that every transition $(q_i, Z_i, Y_i') \xrightarrow{t_i} (q_{i+1}, Z_{i+1}, Y_{i+1})$ is realizable from a valuation ν such that $\nu \vDash (X - Y_i') > 0$. But an easy induction on i shows that actually $\nu_i \vDash (X - Y_i') > 0$.

Since ρ is non-Zeno there are infinitely many i with $Y_i' = \emptyset$. Since the initial run is non-Zeno, σ' is unblocked. □

Lemma 6 *Suppose $GZG^a(\mathscr{A})$ has an unblocked path visiting infinitely often both a clear node and an accepting node then $\mathscr{A}$ has a non-Zeno run satisfying the Büchi condition.*

Proof Let σ be a path in $GZG^a(\mathscr{A})$ as required by the assumptions of the lemma (without loss of generality we assume every alternate transition is a τ transition):

$$(q_0, Z_0, Y_0) \xrightarrow{\tau} (q_0, Z_0, Y_0') \xrightarrow{t_0} \cdots (q_i, Z_i, Y_i) \xrightarrow{\tau} (q_i, Z_i, Y_i') \xrightarrow{t_i} \cdots$$

Take a corresponding path in $ZG^a(\mathscr{A})$ and one instantiation $\rho = (q_0, \nu_0), (q_1, \nu_1) \ldots$ that exists by Theorem 2. If it is non-Zeno then we are done.

Suppose ρ is Zeno. We now show how to build a non-Zeno instantiation of σ from ρ. Let X^r be the set of variables reset infinitely often on σ. As σ is unblocked, every variable not in X^r is bounded only finitely often. Since ρ is Zeno, there is an index m such that the duration of the suffix of the run starting from (q_m, ν_m) is bounded by $1/2$, and no transition in this suffix bounds a variable outside X^r. Let $n > m$ be such that every variable from X^r is reset between m and n. Observe that $\nu_n(x) < 1/2$ for every $x \in X^r$.

Take positions i, j such that $i, j > n$, $Y_i = Y_j = \emptyset$ and all the variables from X^r are reset between i and j. We look at the part of the run ρ:

$$(q_i, \nu_i) \xrightarrow{\delta_i, t_i} (q_{i+1}, \nu_{i+1}) \xrightarrow{\delta_{i+1}, t_{i+1}} \ldots (q_j, \nu_j)$$

and claim that for every $\zeta \in \mathbb{R}_{\geq 0}$ the sequence of the form

$$(q_i, \nu_i') \xrightarrow{\delta_i, t_i} (q_{i+1}, \nu_{i+1}') \xrightarrow{\delta_{i+1}, t_{i+1}} \ldots (q_j, \nu_j')$$

is a part of a run of $\mathscr{A}$ where ν_k' for $k = i, \ldots, j$ satisfy:

1. $\nu'_k(x) = \nu_k(x) + \zeta + 1/2$ for all $x \notin X^r$.
2. $\nu'_k(x) = \nu_k(x) + 1/2$ if $x \in X^r$ and x has not been reset between i and k.
3. $\nu'_k(x) = \nu_k(x)$ otherwise, i.e., when $x \in X^r$ and x has been reset between i and k.

Before proving this claim, let us explain how to use it to conclude the proof. The claim shows that in (q_i, ν_i) we can pass $1/2$ units of time and then construct a part of the run of $\mathscr{A}$ arriving at (q_j, ν'_j) where $\nu'_j(x) = \nu_j(x)$ for all variables in X^r, and $\nu'_j(x) = \nu_j(x) + 1/2$ for other variables. Now, we can find $l > j$, so that the pair (j, l) has the same properties as (i, j). We can pass $1/2$ units of time in j and repeat the above construction getting a longer run that has passed $1/2$ units of time twice. This way we construct a run that passes $1/2$ units of time infinitely often, hence it is non-Zeno. By the construction it passes also infinitely often through accepting nodes.

It remains to prove the claim. Take a transition $(q_k, \nu_k) \xrightarrow{\delta_k, t_k} (q_{k+1}, \nu_{k+1})$ and show that $(q_k, \nu'_k) \xrightarrow{\delta_k, t_k} (q_{k+1}, \nu'_{k+1})$ is also a transition allowed by the automaton. Let g and R be the guard of t_k and the reset of t_k, respectively.

First we need to show that $\nu'_k + \delta_k$ satisfies the guard of t_k. For this, we need to check if for every variable $x \in X$ the constraints in g concerning x are satisfied. We have three cases:

- If $x \notin X^r$ then x is not bounded by the transition t_k, that means that in g the constraints on x are of the form $(x > c)$ or $(x \geq c)$. Since $(\nu_k + \delta_k)(x)$ satisfies these constraints so does $(\nu'_k + \delta_k)(x) \geq (\nu_k + \delta_k)(x)$.
- If $x \in X^r$ and it is reset between i and k then $\nu'_k(x) = \nu_k(x)$ so we are done.
- Otherwise, we observe that $x \notin Y_k$. This is because $Y_i = \emptyset$, and then only variables that are reset are added to Y. Since x is not reset between i and k, it cannot be in Y_k. By definition of transitions in $GZG^a(\mathscr{A})$ this means that $g \wedge (x > 0)$ is consistent. We have that $0 \leq (\nu_k + \delta_k)(x) < 1/2$ and $1/2 \leq (\nu'_k + \delta_k)(x) < 1$. So $\nu'_k + \delta_k$ satisfies all the constraints in g concerning x as $\nu_k + \delta_k$ does.

This shows that there is a transition $(q_k, \nu'_k) \xrightarrow{\delta_k, t_k} (q_{k+1}, \nu')$ for the uniquely determined $\nu' = [R](\nu'_k + \delta_k)$. It is enough to show that $\nu' = \nu'_{k+1}$. For variables not in X^r it is clear as they are not reset. For variables that have been reset between i and k this is also clear as they have the same values in ν'_{k+1} and ν'. For the remaining variables, if a variable is not reset by the transition t_k then its value is the same in ν' and ν'_k. If it is reset then its value in ν' becomes 0; but so it is in ν'_{k+1}, and so the third condition holds. This proves the claim. □

Finally, we provide an explanation as to why the proposed solution does not produce an exponential blowup. At first it may seem that we have gained nothing because when adding arbitrary sets Y we have automatically caused exponential blowup to the zone graph. We claim that this is not the case for the part of $GZG^a(\mathscr{A})$ reachable from the initial node, namely a node with the initial state of $\mathscr{A}$, the zone putting every clock to 0, and $Y = X$.

We say that a zone *orders clocks* if for every two clocks x, y, the zone implies that at least one of $x \leq y$, or $y \leq x$ holds.

Lemma 7 *If a node with a zone Z is reachable from the initial node of the zone graph $ZG^a(\mathscr{A})$ then Z orders clocks. The same holds for $GZG^a(\mathscr{A})$.*

Proof First notice that in the initial zone, all the clocks are equal to each other. Now, consider a zone Z that orders clocks. Let $(q, Z) \xrightarrow{t} (q', Z')$ be a transition of $ZG^a(\mathscr{A})$. This means that there exists a transition $(q, Z) \xrightarrow{t} (q', Z_1')$ in the (unabstracted) zone graph $ZG(\mathscr{A})$ such that $Z' = Approx_M(Z_1')$. Directly from the definition of transitions we have that Z_1' orders clocks. It remains to check that, the clock ordering in Z_1' is preserved in $Z' = Approx_M(Z_1')$. Suppose not, then let $x_1 \leq \cdots \leq x_n$ be the ordering in Z_1'. We get that $Z' \wedge (x_1 \leq \cdots \leq x_n)$ is a smaller convex union of d-regions than Z' that contains Z_1' (recall that $M \geq 0$)—a contradiction. For the second statement observe that for every node (q, Z, Y) in $GZG^a(\mathscr{A})$, (q, Z) is reachable in $ZG^a(\mathscr{A})$. $\square$

Suppose that Z orders clocks. We say that a set of clocks Y *respects the order given by* Z if whenever $y \in Y$ and Z implies $x \leq y$ then $x \in Y$. In other words, Y is downward closed with respect to the ordering constraint in Z.

Lemma 8 *If a node (q, Z, Y) is reachable from the initial node of the guessing zone graph $GZG^a(\mathscr{A})$ then Y respects the order given by Z.*

Proof The proof is by induction on the length of a path. In the initial node (q_0, Z_0, X), the set X obviously respects the order as it is the set of all clocks. Now take a transition $(q, Z, Y) \xrightarrow{t} (q', Z', Y')$ with Y respecting the order in Z. We need to show that Y' respects the order in Z'. By the definition of transitions in $GZG^a(\mathscr{A})$ there are $\nu \in Z$, $\nu' \in Z'$ and $\delta \in \mathbb{R}_{\geq 0}$ such that $(q, \nu) \xrightarrow{\delta, t} (q', \nu')$ and $\nu + \delta \vDash (X - Y) > 0$. Take $y \in Y'$ and suppose that Z' implies $x \leq y$ for some clock x. There are three cases depending on which of the variables y, x are being reset by the transition.

- If x is reset by the transition then, by definition $x \in Y'$.
- If y is reset then Z' implies $y = 0$. Hence Z' implies that $x = 0$. When x is not reset, x is checked for 0 on t. Hence, $x \in Y$ and $x \in Y'$.
- The remaining case is when none of the two variables is reset by the transition. As $\nu' \in Z'$, we have that $\nu' \vDash x \leq y$; and in consequence $\nu \vDash x \leq y$. Since Z orders clocks and $\nu \in Z$, we must have that Z implies $x \leq y$. As y has not been reset, $y \in Y$. By assumption that Y orders clocks, $x \in Y$.

$\square$

The above two lemmas give us the desired bound.

Theorem 4 *Let $|ZG^a(\mathscr{A})|$ be the size of the zone graph, and $|X|$ be the number of clocks in $\mathscr{A}$. The number of reachable nodes of $GZG^a(\mathscr{A})$ is bounded by $|ZG^a(\mathscr{A})|.(|X| + 1)$.*

The theorem follows directly from the above two lemmas. Of course, imposing that zones have ordered clocks in the definition of $GZG^a(\mathscr{A})$ we would get the same bound for the entire $GZG^a(\mathscr{A})$.

3.3 Examples of guessing zone graphs

Figure 8 in Sect. 3.2 depicts a TBA $\mathscr{A}_1$ along with $ZG^a(\mathscr{A}_1)$ and $GZG^a(\mathscr{A}_1)$ (where the τ-loops have been omitted). In order to fire transition $b \xrightarrow{x \leq 1, \{x\}} a$ time must not elapse in b. The third component Y does not help to detect that time cannot elapse in b as in $GZG^a(\mathscr{A}_1)$ the transition is allowed for both $Y = \{x\}$ and $Y = \emptyset$. However, as soon as a strongly-connected component (SCC) contains a transition $x \geq 1$ and a transition that resets x, it has a non-Zeno run, and the third component does not play any role.

The third component is only useful for the case where an SCC contains no transition with a guard implying $x > 0$ for some clock x that is also reset on some transition in the SCC. In such a case, zero-checks may prevent time to elapse. We illustrate this case on the next two examples that emphasize how the third component added to the states of the zone graph allows to distinguish between Zeno runs and non-Zeno runs.

The TBA $\mathscr{A}_2$ shown in Fig. 9 has only runs where the time cannot elapse at all. This is detected in $GZG^a(\mathscr{A}_2)$ as all states in the only non-trivial SCC have $Y = \{x, y\}$ as the third component. This means that from every state there exists a reachable zero-check that is not preceded by the corresponding reset, hence preventing time to elapse. Notice that the correctness of this argument relies on the fact that for every (q, Z, Y) in $GZG^a(\mathscr{A}_2)$, and for every transition $t = (q, g, R, q')$, even if t is fireable in $ZG^a(\mathscr{A}_2)$ from (q, Z), it must also be fireable *under the supplementary hypothesis* $(X - Y) > 0$ given by Y in $GZG^a(\mathscr{A}_2)$.

The TBA $\mathscr{A}_3$ in Fig. 9 admits a non-Zeno run. This can be read from $GZG^a(\mathscr{A}_3)$ since the SCC composed of the four zones with $Y = \{x, y\}$ together with $(z_2, \emptyset)$ and $(z_3, \{y\})$ contains a clear node. This is precisely the state where time can elapse as every reachable zero-check is preceded by the corresponding reset.

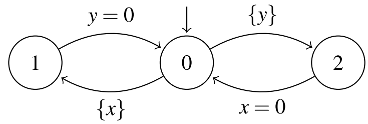
4 Algorithm

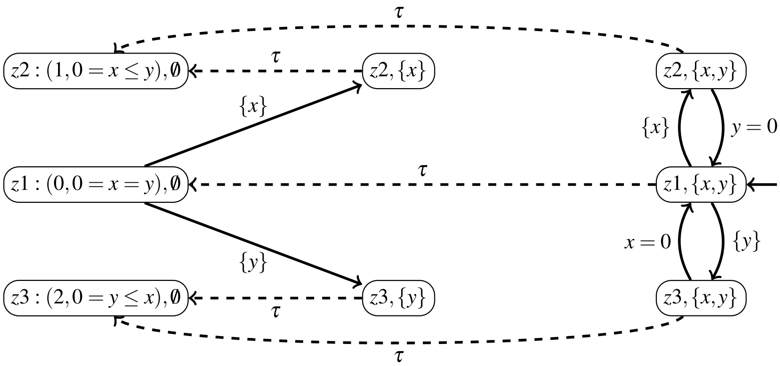
In this section, we provide an on-the-fly algorithm for the Büchi non-emptiness problem using the guessing zone graph construction developed in Sect. 3.2. In the later part of the section, we observe that in most cases, non-Zenoness could be detected directly from the standard zone graph, without extra construction. We provide an optimized on-the-fly algorithm taking into account these observations.

We will use Theorem 3 to algorithmically check if an automaton $\mathscr{A}$ has a non-Zeno run satisfying the Büchi condition. The theorem requires to find an unblocked path in $GZG^a(\mathscr{A})$ visiting both an accepting node and a clear node infinitely often. This problem is similar to that of testing for emptiness of automata with generalized Büchi conditions as we need to satisfy two infinitary conditions at the same time. The requirement of a path being unblocked adds additional complexity to the problem. The best algorithms for testing emptiness of automata with generalized Büchi conditions are based on Tarjan's algorithm for strongly connected components (SCC) [15, 22]. So this is the way we take here. In particular, we adopt the variant given by Couvreur [11, 12].

In general, the verification problem for timed systems involves checking if a network of timed automata $\mathscr{A}_1, \dots, \mathscr{A}_n$ satisfies a given property ϕ. Assuming that ϕ can be translated into a (timed) Büchi automaton $\mathscr{A}_{\neg\phi}$, we reduce the verification problem to the emptiness of a timed Büchi automaton $\mathscr{A}$ defined as a product $\mathscr{A}_1 \times \mathscr{A}_2 \times \cdots \times \mathscr{A}_n \times \mathscr{A}_{\neg\phi}$ for some synchronization policy. Couvreur's algorithm is an extension of Tarjan's algorithm for computing maximal SCCs in a graph. One of its main features is that it stops as soon as a (non necessarily maximal) SCC with an accepting state has been found. In addition, it handles multiple accepting conditions efficiently. To this regard, the algorithm computes the set of accepting conditions in each maximal SCC of $\mathscr{A}$. Initially, each state s in $\mathscr{A}$ is considered as a trivial SCC labelled with the accepting conditions of s. The algorithm computes the states of $\mathscr{A}$ on-the-fly in a depth-first search (DFS) manner starting from the initial state. During the search, when a cycle is found, all the SCCs in the cycle are merged into a bigger SCC Γ that inherits their accepting conditions. If Γ contains all the required accepting conditions, the algorithm stops declaring $\mathscr{A}$ to be not empty. Notice that Γ need not be maximal. Otherwise it resumes the DFS on $\mathscr{A}$. We direct the reader to [11, 12, 18] for further details on the Couvreur's algorithm.

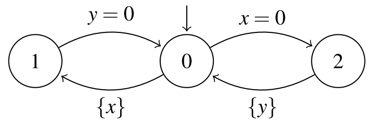
$\mathscr{A}_2$

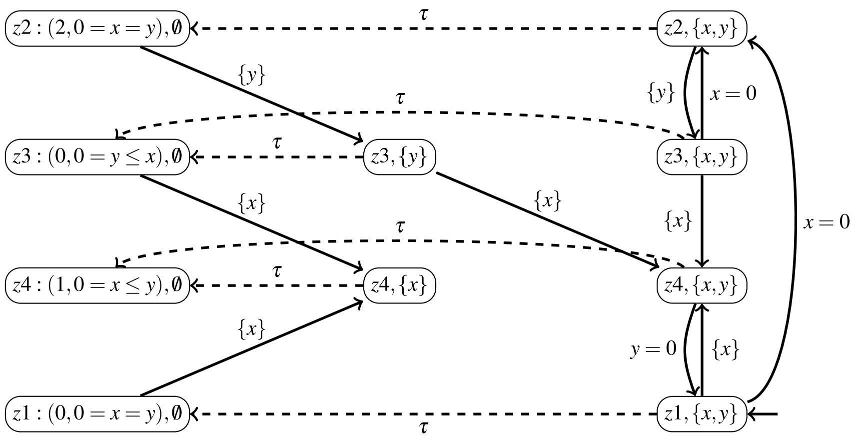




Reachable part of GZG^a$(\mathscr{A}_2)$

$\mathscr{A}_3$





Reachable part of GZG^a$(\mathscr{A}_3)$

Fig. 9 Examples of guessing zone graphs (τ self-loops have been omitted for clarity)

In the next section, we show how to enhance Couvreur's algorithm to detect runs that are not only accepting but also non-Zeno. It is achieved by associating extra information to the SCCs in $\mathscr{A}$. This information is updated when SCCs are merged like for accepting conditions.

4.1 Emptiness check on $GZG^a(\mathscr{A})$

We apply Couvreur's algorithm for detecting maximal SCCs in $GZG^a(\mathscr{A})$. During the computation of the maximal SCCs, we keep track of whether an accepting node and a clear node have been seen. For the unblocked condition we use two sets of clocks U_Γ and R_Γ that respectively contain the clocks that are bounded and the clocks that are reset in the SCC Γ. A clock from $U_\Gamma - R_\Gamma$ is called *blocking* since being bounded and not reset it puts a limit on the time that can pass. At the end of the exploration of Γ we check if:

1. we have passed through an accepting node and a clear node,
2. there are no blocking clocks: $U_\Gamma \subseteq R_\Gamma$.

If the two conditions are satisfied then we can conclude saying that $\mathscr{A}$ has an accepting non-Zeno run. Indeed, a path passing infinitely often through all the nodes of Γ would satisfy the conditions of Theorem 3, giving a required run of $\mathscr{A}$. If the first condition does not hold then the same theorem says that Γ does not have a witness for a non-Zeno run of $\mathscr{A}$ satisfying the Büchi condition.

The interesting case is when the first condition holds but not the second. The following lemma yields an algorithm in that case.

Lemma 9 *Let Γ be an SCC in $GZG^a(\mathscr{A})$ with an accepting node and a clear node, and such that $U_\Gamma \nsubseteq R_\Gamma$. There exists an unblocked path in Γ that visits both an accepting node and a clear node infinitely often iff there exists a sub-SCC $\Gamma' \subseteq \Gamma$ with an accepting node and a clear node and such that $U_{\Gamma'} \subseteq R_{\Gamma'}$.*

Proof Assume that Γ has an unblocked path that visits both an accepting node and a clear node infinitely often. Then, define Γ' as the set of nodes and edges that are visited infinitely often on that path.

Conversely, if such a sub-SCC Γ' exists, then consider an infinite path in Γ' that goes infinitely often through each node and each transition in Γ'. This path is unblocked and visits both an accepting node and a clear node. This path is also a path in Γ. □

We call *blocking edges* all the edges in Γ that bound a clock from $U_\Gamma \setminus R_\Gamma$. We proceed as follows. We discard all the blocking edges from Γ as every unblocked path in Γ goes only finitely many times through these edges. In general, this yields several candidates for Γ'. Each of them is a proper sub-SCC of Γ. Then, we restart our algorithm on each such Γ'. Since we have discarded some edges from Γ (hence some resets), a clock may be now blocking in Γ'. If this is the case, the blocking edges in Γ' will be discarded, and the resulting sub-SCCs of Γ' will be explored, and so on. Observe that each transition in $GZG^a(\mathscr{A})$ will be visited at most $|X| + 1$ times, as we eliminate at least one clock at each restart. If after exploring the entire graph, the algorithm has not found a subgraph satisfying the two conditions then it declares that there is no run of $\mathscr{A}$ with the desired properties. The correctness of the procedure is based on Theorem 3. All the procedure: exploring Γ, discarding blocking edges, exploring all Γ' candidates, etc, can be done on-the-fly without storing Γ as described in [18].

Recall that by Theorem 4 the size of $GZG^a(\mathscr{A})$ is $\mathscr{O}(|ZG^a(\mathscr{A})| \cdot |X|)$. The complexity of the algorithm follows from the linear complexity of Couvreur's algorithm and the remark about the bound on the number of times each transition is visited. We hence obtain the following.

Theorem 5 *The above algorithm is correct and runs in time* $\mathscr{O}(|ZG^a(\mathscr{A})| \cdot |X|^2)$.

Although the guessing zone graph provides a way to detect non-Zeno paths, it is useful only when the automaton indeed contains zero-checks. The next challenge therefore lies in optimizing the use of the guessing zone graph construction, that is, applying Couvreur's algorithm directly on the standard zone graph and using the guessing zone graph construction only when required.

4.2 Optimized use of guessing zone graph construction

The idea is to apply Couvreur's algorithm directly on $ZG^a(\mathscr{A})$ and find an SCC with an accepting node. An SCC is said to be *unblocked* if it contains no blocking clock; recall that it is a clock x that is checked for a guard which implies $x \leq c$ for a constant c and that is reset in no transition of the SCC.

Non-Zenoness can be ensured if the SCC satisfies one of the following conditions:

– It is unblocked and free from zero-checks. A zero-check is detected for a transition $(q, Z) \xrightarrow{g,R} (q', Z')$ and some clock x when for each $\nu \in Z$ and $\delta \in \mathbb{R}_{\geq 0}$ such that $\nu + \delta \vDash g$, we have $(\nu + \delta)(x) = 0$.
– There is a clock x that is reset in the SCC and one of the transitions in the SCC implies $x \geq 1$.

For the second condition, note that such a reachable SCC instantiates into a path ρ of $\mathscr{A}$ whose suffix corresponds to repeated traversal of this SCC. Every traversal resets x and checks for a guard that implies $x \geq 1$. Therefore, at least 1 time unit elapses in each traversal, implying that ρ is a non-Zeno run. Notice that this relies on the same principle as the one used in the Strongly Non-Zeno construction [25] (see Sect. 3.1). However, in our case we exploit the information from $\mathscr{A}$: we do not add any new clock. Our algorithm will compute on the fly the set L_Γ of clocks x such that $x \geq 1$ is implied by some guard in Γ. This is done in the same way as for U_Γ in the previous subsection. Then, Γ satisfies the second condition above if $L_\Gamma \cap R_\Gamma$ is not empty.

The first condition is justified by the following lemma.

Lemma 10 *If* $ZG^a(\mathscr{A})$ *has an unblocked path that visits an accepting node infinitely often, and has only finitely many transitions with zero-checks, then* $\mathscr{A}$ *has a non-Zeno run satisfying the Büchi condition.*

Proof Let σ be the path in $ZG^a(\mathscr{A})$ as required by the assumptions of the lemma:

$$(q_0, Z_0) \xrightarrow{t_0} \cdots (q_i, Z_i) \xrightarrow{t_i} \cdots$$

Since zero-checks occur only finitely often in σ, we can find j such that the suffix $(q_j, Z_j) \xrightarrow{t_j} \dots$ of σ contains no zero-checks in its transitions. Let σ' be the following sequence:

$$(q_0, Z_0, Y_0) \xrightarrow{\tau} (q_0, Z_0, Y_0') \xrightarrow{t_0} (q_1, Z_1, Y_1) \xrightarrow{\tau} (q_1, Z_1, Y_1') \xrightarrow{t_1} \cdots$$

where $Y_0 = X$, Y_i is determined by the transition, and $Y'_i = Y_i$ for all $i \leq j$ and for $i > j$, $Y'_i = \emptyset$. Note that σ' is a path in $GZG^a(\mathscr{A})$. For this to be true, each transition $(q_i, Z_i, Y'_i) \xrightarrow{t_i} (q_{i+1}, Z_{i+1}, Y_{i+1})$ should be realizable from a valuation ν_i such that $\nu_i \vDash (X - Y'_i) > 0$. This is vacuously true if $i \leq j$ since $Y'_i = X$ for all $i \leq j$. For $i > j$, $Y'_i = \emptyset$ and since t_i does not contain a zero-check, the transition is realizable from a valuation ν_i in which all clocks are strictly greater than 0.

Since σ is unblocked, σ' is unblocked too. By definition all but finitely many nodes for σ' are clear. Finally, σ' visits an accepting node infinitely often. By Theorem 3, $\mathscr{A}$ has a non-Zeno run satisfying the Büchi condition. □

The above two observations give a sufficient condition for terminating with a success when an SCC Γ with an accepting node is found in $ZG^a(\mathscr{A})$. If the above two conditions do not hold, then Γ has no clock bounded from below (i.e. $x \geq 1$) and Γ either has blocking clocks or zero-checks. If it has only blocking clocks, we apply the procedure that restarts the exploration with blocking edges removed, as described in Sect. 4.1. If Γ has zero-checks, we indeed use the guessing zone graph construction, however *restricted only to the nodes of* Γ. The problem is to know the initial set of clocks that need to be zero. We first define a few notations.

Let (q^Γ, Z^Γ) be the root of Γ as determined by Couvreur's algorithm. Let $GZG^a_{|\Gamma}(\mathscr{A})$ be the part of $GZG^a(\mathscr{A})$ rooted at (q^Γ, Z^Γ, X) and restricted only to the nodes and transitions that occur in Γ. We say that a run ρ of $\mathscr{A}$ is *trapped* in an SCC Γ of $ZG^a(\mathscr{A})$ if a suffix of ρ is an instantiation of a path in Γ. The following lemma justifies the use of the restricted guessing zone graph construction starting from (q^Γ, Z^Γ, X).

Lemma 11 *The automaton $\mathscr{A}$ has an accepting non-Zeno run trapped in an SCC Γ of $ZG^a(\mathscr{A})$ iff $GZG^a_{|\Gamma}$ has an SCC that is accepting, unblocked and contains a clear node.*

Proof For the left-right direction, consider the following run ρ of $\mathscr{A}$ trapped in Γ:

$$(q_0, \nu_0) \xrightarrow{\delta_0, t_0} \cdots (q_m, \nu_m) \xrightarrow{\delta_m, t_m} \cdots$$

where $q_m = q^\Gamma$, $\nu_m \in Z^\Gamma$ and (q^Γ, Z^Γ) is the root of Γ. Consider the sequence σ':

$$(q_0, Z_0, Y_0) \xrightarrow{\tau} (q_0, Z_0, Y'_0) \xrightarrow{t_0} (q_1, Z_1, Y_1) \xrightarrow{\tau} (q_1, Z_1, Y'_1) \xrightarrow{t_1} \cdots$$

where

- (q_0, Z_0) is the initial node of $ZG^a(\mathscr{A})$, the zone Z_i is determined by the transition t_{i-1},
- $Y_0 = X$, Y_i is determined by the transition,
- $Y'_i = Y_i$ for all $i \leq m$; for $i > m$, $Y'_i = \emptyset$ if $\delta_i > 0$ and $Y'_i = Y_i$ otherwise.

Observe that $Y_m = X$ and the suffix of σ' starting from (q_m, Z_m, Y_m) is a path of $GZG^a_{|\Gamma}(\mathscr{A})$. Since there are infinitely many i with $\delta_i > 0$, this suffix corresponds to an SCC that has a clear node. It is accepting and unblocked since the run ρ that we started with is accepting and non-Zeno.

For the right-left direction, note that an accepting, unblocked SCC with a clear node in $GZG^a_{|\Gamma}(\mathscr{A})$ corresponds to an accepting, unblocked path of $GZG^a(\mathscr{A})$ starting from (q^Γ, Z^Γ, X) that visits a clear node infinitely often. It is straightforward to see that (q^Γ, Z^Γ, X) is reachable from the initial node (q_0, Z_0, X) of $GZG^a(\mathscr{A})$ through a path in which for all transitions $(q, Z, Y) \xrightarrow{\tau} (q', Z', Y')$, $Y' = Y$. Indeed, the restriction of $GZG^a(\mathscr{A})$ to its nodes with $Y = X$ is isomorphic to the zone graph $ZG^a(\mathscr{A})$. From this path of $GZG^a(\mathscr{A})$ and using Lemma 6, we can construct a accepting, non-Zeno run of $\mathscr{A}$ that is trapped in Γ. □

Based on the above observations, we give the schema of the overall optimized algorithm in Fig. 10. In the worst case, the algorithm runs in time $\mathscr{O}(|ZG^a(\mathscr{A})| \cdot |X|^2)$. When the automaton does not have zero-checks it runs in time $\mathscr{O}(|ZG^a(\mathscr{A})| \cdot |X|)$. When the automaton further has no blocking clocks, it runs in time $\mathscr{O}(|ZG^a(\mathscr{A})|)$.

5 Experiments

We have implemented our algorithms in a prototype verification tool. Given a *network* $\mathscr{A}_1, \ldots, \mathscr{A}_n$ of timed Büchi automata, we want to check if this network satisfies a property ϕ specified in some logic. We consider a property ϕ that can be translated into a timed automaton $\mathscr{A}_{\neg\phi}$ such that the network satisfies ϕ iff the product timed automaton $\mathscr{A}_1 \times \cdots \mathscr{A}_n \times \mathscr{A}_{\neg\phi}$ has an empty language. Table 1 presents the results that we obtained on several classical examples. The "Models" column represents the product of the network Timed Büchi Automata and the property to verify. We give the number of processes in the network for each model. A tick in the "Sat." columns tells that the property is satisfied by the model. The "Zone Graph" column gives the number of nodes in the zone graph. Next, for the "Strongly non-Zeno" construction, we give the size of the resulting zone graph followed by the number of nodes that are visited during verification using the Couvreur's algorithm. Similarly for the "Guessing Zone Graph" but using the algorithm in Sect. 4.1. Finally, the last column corresponds to our fully optimized algorithm as described in Sect. 4.2.

We have considered three types of properties: reachability properties (mutual exclusion, collision detection for CSMA/CD), liveness properties (access to the resource infinitely often), and bounded response properties (which are reachability properties with real-time requirements). Reachability properties require to find a path to a target state starting from the initial state. Although this path is a finite sequence, it is realistic only if this finite sequence can be extended to a non-Zeno path of the automaton. Therefore, while verifying reachability properties, we check if the automaton has a non-Zeno path that contains the target state.

The strongly non-Zeno construction outperforms the guessing zone graph construction for reachability properties. This is particularly the case for mutual exclusion on the Fischer's protocol and collision detection for the CSMA/CD protocol. For liveness properties, the results are more balanced. On the one hand, the strongly non-Zeno construction is once again more efficient for the CSMA/CD protocol. On the other hand the differences are tight in the case of Fischer protocol. The guessing zone graph construction distinguishes itself for bounded response properties. Indeed, the Train-Gate model is an example of exponential blowup for the strongly non-Zeno construction.

We notice that on-the-fly algorithms perform well. Even when the graphs are big, particularly in case when automata are not empty, the algorithms are able to conclude after having explored only a small part of the graph. Our optimized algorithm outperforms the two others on most examples. Particularly, for the CSMA/CD protocol with 5 stations our algorithm needs to visit only 4841 nodes while the two other methods visited 8437 and 21038 nodes. This confirms our initial hypothesis: most of the time, the zone graph contains enough information to ensure time progress. As a consequence, checking non-Zenoness and emptiness

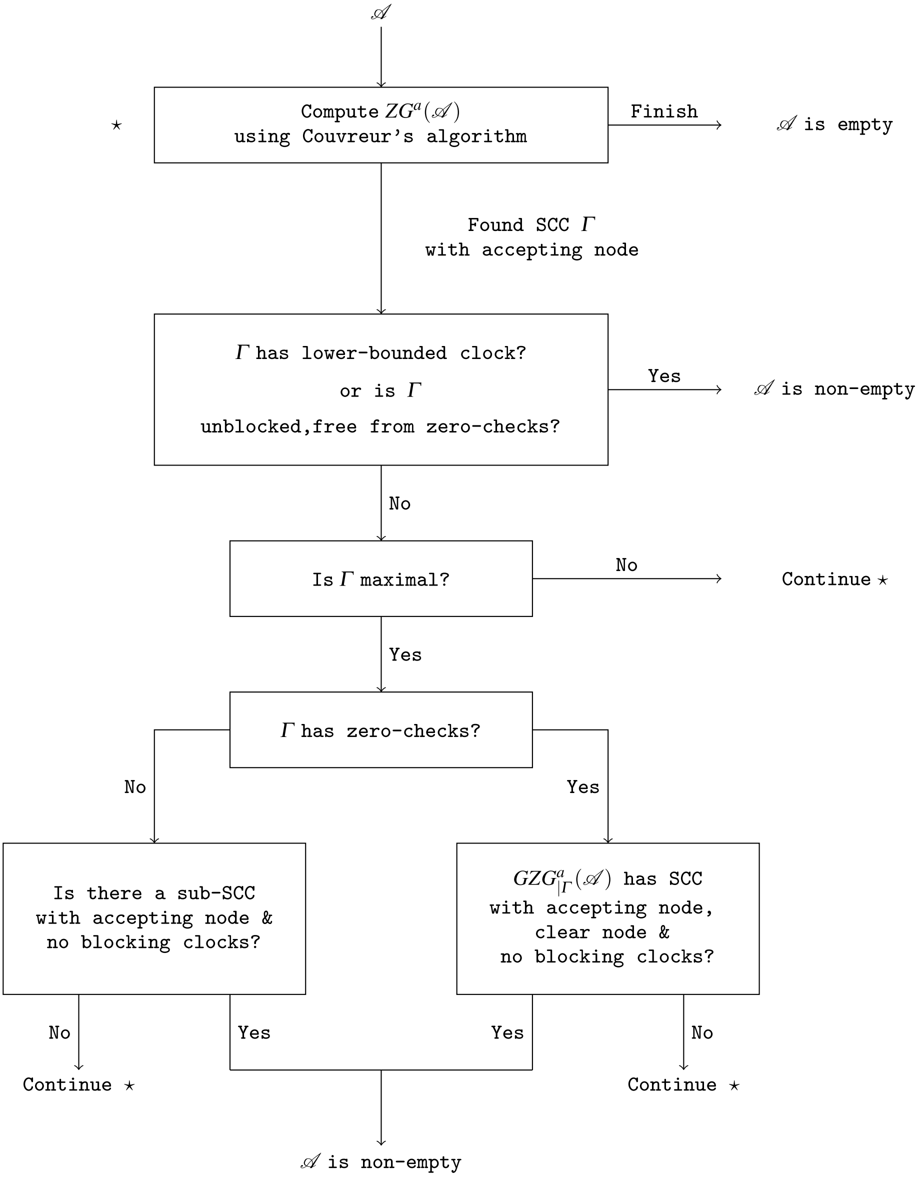


Fig. 10 Algorithm to check for Büchi emptiness of $\mathscr{A}$. "Continue" loops back to computing $ZG^a(\mathscr{A})$ using Couvreur's Algorithm

is done at the same cost as checking emptiness only. This is in turn achieved at a cost that is similar to reachability checking.

Our optimization using lower bounds on clocks also proves useful for the FDDI protocol example. One of its processes has zero-checks, but since some other clock is bounded from below and reset, it was not necessary to explore the guessing zone graph to conclude non-emptiness.

Table 1 Experimental results. The "Sat." column tells which properties are satisfied by the model. The "size" columns give the number of nodes in the corresponding graphs. The "visited" columns give the number of nodes that are visited by the corresponding algorithm. The results correspond to the Couvreur's algorithm for $ZG^a(SNZ(\mathscr{A}))$, the algorithm in Sect. 4.1 for $GZG^a(\mathscr{A})$ and the algorithm in Sect. 4.2 for the "Optimized" column

Models ($\mathscr{A}$)	Sat.	$ZG^a(\mathscr{A})$	$ZG^a(SNZ(\mathscr{A}))$		$GZG^a(\mathscr{A})$		Optimized
		size	size	visited	size	visited	visited
Train-Gate2 (mutex)	√	134	194	194	400	400	134
Train-Gate2 (bound. resp.)		988	227482	352	3840	1137	292
Train-Gate2 (liveness)		100	217	35	298	53	33
Fischer3 (mutex)	√	1837	3859	3859	7292	7292	1837
Fischer4 (mutex)	√	46129	96913	96913	229058	229058	46129
Fischer3 (liveness)		1315	4962	52	5222	64	40
Fischer4 (liveness)		33577	147167	223	166778	331	207
FDDI3 (liveness)		508	1305	44	3654	79	42
FDDI5 (liveness)		6006	15030	90	67819	169	88
FDDI3 (bound. resp.)		6252	41746	59	52242	114	60
CSMA/CD4 (collision)	√	4253	7588	7588	20146	20146	4253
CSMA/CD5 (collision)	√	45527	80776	80776	260026	260026	45527
CSMA/CD4 (liveness)		3038	9576	1480	14388	3075	832
CSMA/CD5 (liveness)		32751	120166	8437	186744	21038	4841

6 Conclusions

The Büchi non-emptiness problem is one of the standard problems for timed automata. Since the paper introducing the model, it has been widely accepted that the addition of one auxiliary clock is an adequate method to deal with the problem of Zeno paths. This technique is also used in the recently proposed zone based algorithm for the problem [24].

In this paper, we have shown that in some cases the auxiliary clock may cause exponential blowup in the size of the zone graph. We have proposed another method that is based on a modification of the zone graph. The resulting graph grows only by a factor that is linear in the number of clocks. In our opinion, the efficiency gains of our method outweigh the fact that it requires some small modifications in the code dealing with zone graph exploration. Moreover, liveness can be checked at the same cost as reachability as demonstrated by our experiments. This also shows that in most cases the zone graph already contains enough information to handle non-Zenoness.

As future work we plan to extend our algorithm to commonly used syntactic extensions of timed automata. For example, UPPAAL and Kronos allow reset of clocks to arbitrary values, which is convenient for modeling real life systems. This would require to extend the guessing zone graph construction and consequently our algorithm. In this paper, we considered the *Approx* abstraction that has been largely improved by later works [3]. It has been shown that these new abstractions preserve Büchi conditions [21]. We plan to study the extension of our technique to these abstractions. Finally, we also plan to extend our construction to extract non-Zeno strategies in timed games.

References

1. Alur R, Dill DL (1994) A theory of timed automata. Theor Comput Sci 126(2):183–235
2. Alur R, Madhusudan P (2004) Decision problems for timed automata: A survey. In: Bernardo M, Corradini F (eds) Formal methods for the design of real-time systems, international school on formal methods for the design of computer, communication and software systems, SFM-RT 2004, Bertinoro, Italy, September 13–18, 2004, Revised Lectures. Lecture notes in computer science, vol 3185. Springer, Berlin, pp 1–24
3. Behrmann G, Bouyer P, Larsen KG, Pelanek R (2006) Lower and upper bounds in zone-based abstractions of timed automata. Int J Softw Tools Technol Transf 8(3):204–215
4. Behrmann G, David A, Larsen KG, Haakansson J, Pettersson P, Yi W, Hendriks M (2006) Uppaal 4.0. In: Third international conference on the quantitative evaluation of systems (QEST 2006), 11–14 September 2006, Riverside, California, USA. IEEE Computer Society, Los Alamitos, pp 125–126
5. Bérard B, Bouyer B, Petit A (2004) Analysing the pgm protocol with UPPAAL. Int J Prod Res 42(14):2773–2791
6. Berthomieu B, Menasche M (1983) An enumerative approach for analyzing time petri nets. In: IFIP Congress, pp 41–46
7. Bouyer P (2003) Untameable timed automata! In: Alt H, Habib M (eds) STACS 2003, 20th annual symposium on theoretical aspects of computer science, Proceedings, Berlin, Germany, February 27–March 1, 2003. Lecture notes in computer science, vol 2607. Springer, Berlin, pp 620–631
8. Bouyer P (2004) Forward analysis of updatable timed automata. Form Methods Syst Des 24(3):281–320
9. Bowman H, Gómez R (2006) How to stop time stopping. Form Asp Comput 18(4):459–493
10. Bozga M, Daws C, Maler O, Olivero A, Tripakis S, Yovine S (1998) Kronos: a model-checking tool for real-time systems. In: Hu AJ, Vardi MY (eds) Computer aided verification, 10th international conference, CAV '98 Proceedings, Vancouver, BC, Canada, June 28–July 2, 1998. Lecture notes in computer science, vol 1427. Springer, Berlin, pp 546–550
11. Couvreur J-M (1999) On-the-fly verification of linear temporal logic. In: FM'99—formal methods, world congress on formal methods in the development of computing systems, Proceedings, vol I. Toulouse, France, September 20–24, 1999, p 1708
12. Couvreur J-M, Duret-Lutz A, Poitrenaud D (2005) On-the-fly emptiness checks for generalized Büchi automata. In: Godefroid P (ed) Model checking software, 12th international SPIN workshop, Proceedings, San Francisco, CA, USA, August 22–24, 2005. Lecture notes in computer science, vol 3639. Springer, Berlin, pp 169–184
13. Daws C, Tripakis S (1998) Model checking of real-time reachability properties using abstractions. In: Steffen B (ed) Tools and algorithms for construction and analysis of systems, 4th international conference, TACAS '98, held as part of the European joint conferences on the theory and practice of software, ETAPS'98, Proceedings, Lisbon, Portugal, March 28–April 4, 1998. Lecture notes in computer science, vol 1384, pp 313–329
14. Dill DL (1990) Timing assumptions and verification of finite-state concurrent systems. In: Sifakis J (ed) Automatic verification methods for finite state systems, international workshop, Proceedings, Grenoble, France, June 12–14, 1989. Lecture notes in computer science, vol 407. Springer, Berlin, pp 197–212
15. Gaiser A, Schwoon S (2009) Comparison of algorithms for checking emptiness on Büchi automata. In: Hilený P, Matyás V, Vojnar T (eds) Annual doctoral workshop on mathematical and engineering methods in computer science, MEMICS 2009, November 13–15, Prestige Hotel, Znojmo, Czech Republic, OASICS, vol 13. Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, Germany, 2009, pp 69–77
16. Gómez R, Bowman H (2007) Efficient detection of Zeno runs in timed automata. In: Raskin J-F, Thiagarajan PS (eds) Formal modeling and analysis of timed systems, 5th international conference, FORMATS 2007, Proceedings, Salzburg, Austria, October 3–5, 2007. Lecture notes in computer science, vol 4763. Springer, Berlin, pp 195–210
17. Havelund K, Skou A, Larsen KG, Lund K (1997) Formal modeling and analysis of an audio/video protocol: An industrial case study using UPPAAL. In: Proceedings of the 18th IEEE real-time systems symposium (RTSS '97), December 3–5, 1997, San Francisco, CA, USA. IEEE Computer Society, Los Alamitos, pp 2–13
18. Herbreteau F, Srivathsan B (2010) Efficient on-the-fly emptiness check for timed Büchi automata. In: Bouajjani A, Chin W-N (eds) Automated technology for verification and analysis: 8th international symposium, ATVA 2010, Proceedings, Singapore, September 21–24, 2010. Lecture notes in computer science, vol 6252. Springer, Berlin, pp 218–232
19. Herbreteau F, Srivathsan B (2011) Coarse abstractions make Zeno behaviors difficult to detect. In: Katoen J-P, König B (eds) Concurrency theory, 22nd international conference, CONCUR 2011, Proceedings, Aachen, Germany, September 6–9, 2011. Lecture notes in computer science, vol 6901. Springer, Berlin, pp 92–107

20. Jessen JJ, Rasmussen JI, Larsen KG, David A (2007) Guided controller synthesis for climate controller using UPPAAL TiGA. In: Raskin J-F, Thiagarajan PS (eds) Formal modeling and analysis of timed systems, 5th international conference, FORMATS 2007, Proceedings, Salzburg, Austria, October 3–5, 2007. Lecture notes in computer science, vol 4763. Springer, Berlin, pp 227–240
21. Li G (2009) Checking timed Büchi automata emptiness using LU-abstractions. In: Ouaknine J, Vaandrager F (eds) Formal modeling and analysis of timed systems, 7th international conference, FORMATS 2009, Proceedings, Budapest, Hungary, September 14–16, 2009. Lecture notes in computer science, vol 5813. Springer, Berlin, pp 228–242
22. Schwoon S, Esparza J (2005) A note on on-the-fly verification algorithms. In: Halbwachs N, Zuck LD (eds) Tools and algorithms for the construction and analysis of systems, 11th international conference, TACAS 2005, held as part of the joint European conferences on theory and practice of software, ETAPS 2005, Proceedings, Edinburgh, UK, April 4–8, 2005. Lecture notes in computer science, vol 3440, pp 174–190
23. Tripakis S (1999) Verifying progress in timed systems. In: Katoen J-P (ed) Formal methods for real-time and probabilistic systems, 5th international AMAST workshop, ARTS'99, Proceedings, Bamberg, Germany, May 26–28, 1999. Lecture notes in computer science, vol 1601. Springer, Berlin, pp 299–314
24. Tripakis S (2009) Checking timed Büchi emptiness on simulation graphs. ACM Trans Comput Logic, 10(3)
25. Tripakis S, Yovine S, Bouajjani A (2005) Checking timed Büchi automata emptiness efficiently. Form Methods Syst Des 26(3):267–292