Shrinking Timed Automata

Ocan Sankur, Patricia Bouyer, and Nicolas Markey

LSV, CNRS & ENS Cachan, France
`{sankur,bouyer,markey}@lsv.ens-cachan.fr`

Abstract

We define and study a new approach to the implementability of timed automata, where the semantics is perturbed by imprecisions and finite frequency of the hardware. In order to circumvent these effects, we introduce *parametric shrinking* of clock constraints, which corresponds to tightening these. We propose symbolic procedures to decide the existence of (and then compute) parameters under which the shrunk version of a given timed automaton is non-blocking and can time-abstract simulate the exact semantics. We then define an implementation semantics for timed automata with a digital clock and positive reaction times, and show that for shrinkable timed automata, non-blockingness and time-abstract simulation are preserved in implementation.

1998 ACM Subject Classification D.2.4 Software/Program Verification; F.1.1 Specifying and Verifying and Reasoning about Programs

Keywords and phrases timed automata, implementability, robustness

1 Introduction

Timed automata [3] are a well-established model in real-time system design. They offer an automata-theoretic framework to design, verify and synthesize systems with timing constraints. The theory behind timed automata has been extensively studied and mature model-checking tools are available. However, this model makes unrealistic assumptions on the system, such as the perfect continuity of clocks and instantaneous reaction times, which are not preserved in implementation even in digital hardware with arbitrary finite precisions. Often, the *synchrony hypothesis* allows one to ignore this issue and greatly simplifies the design phase [7]. However, the synchrony hypothesis still needs to be formally validated once the design phase is over. In fact, perturbations on clocks, either imprecisions or clock drifts, however small they may be, may yield extra qualitative behaviours in some timed systems [22, 14]; positive reaction times can also disable desired behaviours [11, 1].

This raises the question of *implementability*, *i.e.*, whether the model can be implemented on physical machines, preserving (the properties of) its exact semantics. In order to model the behaviour of *implementations* of timed automata, and validate the synchrony hypothesis, De Wulf *et al.* introduced the *program semantics* for timed automata, which defines the behaviour of a timed automaton on a simple micro-processor with a digital clock [15]. This semantics is a bit jagged, and the *enlarged semantics* has been proposed as a convenient over-approximation: it models imprecisions by relaxing all guards, turning clock constraints of the form $x \in [a, b]$ into $x \in [a - \Delta, b + \Delta]$ for some positive parameter Δ. *Robust model checking*, which consists in deciding the existence of a value for Δ under which a property is satisfied in the enlarged semantics, has been proven decidable for safety properties [14], and for richer linear-time properties [9, 10]. In this framework, the implementation (the program semantics) always has more behaviours than the abstract model (the exact semantics), due to

Partly supported by project ImpRo (ANR-2010-BLAN-0317) of the French National Agency for Research.

the relaxation of the timing constraints, and robust model-checking only ensures correctness for properties preserved by timed simulation.

We adopt the following approach: instead of checking properties on the enlarged semantics of a given timed automaton $\mathcal{A}$, we look for a new timed automaton $\mathcal{B}$ whose semantics would *correspond* to the (exact) semantics of $\mathcal{A}$. To circumvent the effect of the imprecisions, our idea is to construct $\mathcal{B}$ by *shrinking* the guards of $\mathcal{A}$, which is the opposite to enlargement, so that all behaviours of $\mathcal{B}$ under enlargement are included in those of $\mathcal{A}$. The timed automaton $\mathcal{B}$ constructed in this manner preserves in particular all universal properties (like linear-time properties) proven (say, by model-checking) for $\mathcal{A}$. This also means that all timing requirements satisfied by $\mathcal{A}$, such as critical deadlines, are strictly respected by $\mathcal{B}$. However, such a transformation may remove too many behaviours and even introduce deadlocks in $\mathcal{B}$. The preservation of the desired behaviours in $\mathcal{B}$ is the problem we are interested in.

A timed automaton $\mathcal{A}$ is said *shrinkable* if it can be shrunk into a timed automaton that is non-blocking, and/or can time-abstract-simulate $\mathcal{A}$. We do not restrict to one single shrinking parameter, but to one parameter per atomic clock constraint in the automaton. We give algorithms to decide the existence of these shrinking parameters, and compute the least parameters when they exist. We show that shrinkability w.r.t. non-blockingness can be checked in PSPACE, shrinkability w.r.t. simulation in EXPTIME, and shrinkability (w.r.t. both requirements) in EXPTIME, by symbolic procedures manipulating difference bound matrices.

As a second result, we define an implementation semantics of timed automata executed by a digital system with a digital clock. Our semantics is similar to the program semantics of [15] but it is valid under slightly different assumptions. We study the relations between the exact semantics and the implementation semantics, and prove additional properties besides the one given in [15]. We show that when a timed automaton $\mathcal{A}$ is shrinkable, say to a timed automaton $\mathcal{B}$, then the implementation semantics of $\mathcal{B}$ is non-blocking and time-abstract-simulates the exact semantics of $\mathcal{A}$. Thus, our framework allows not only to obtain an implementation that contains no more behaviour than the abstract model but also to ensure non-blockingness and time-abstract similarity. This provides a precise motivation for the shrinkability problem: shrinkability is a sufficient condition for the correctness of the implementation semantics.

Finally, notice that shrinkability is also an interesting property by itself, since it asks whether the given automaton is vulnerable to infinitesimal shrinkings (which can be due to imprecisions). Zeno behaviours and other different convergence phenomena [11] are also naturally excluded in shrunk systems (see Section 3.2).

2 Preliminaries

A *timed transition system (TTS)* is a tuple $(S, s_0, \Sigma, \rightarrow)$, where S is the set of *states*, $s_0 \in S$ the initial state, Σ a finite alphabet, and $\rightarrow \subseteq S \times (\Sigma \times \mathbb{R}_{\geq 0}) \times S$ the *transitions*. Transitions are labelled by $\sigma(T)$, with $T \in \mathbb{R}_{\geq 0}$ the *timestamp* of *action* $\sigma \in \Sigma$. In all TTSs we consider, the timestamps of consecutive actions are assumed to be nondecreasing. A TTS $(S, s_0, \Sigma, \rightarrow)$ is *non-blocking* if for any transition $s_1 \xrightarrow{\sigma(T)} s_2$, there exist $\sigma' \in \Sigma$, $T' \geq T$ and $s_3 \in S$ such that $s_2 \xrightarrow{\sigma'(T')} s_3$. Notice that, in this definition, we do not require s_1 to be reachable from s_0.

▶ **Definition 1.** Let $\mathcal{S} = (S, s_0, \Sigma, \rightarrow)$ be a TTS. A relation $R \subseteq S \times S$ is a *timed* (resp. *time-abstract*) *simulation* if for all $(s_1, s_2) \in R$, if $s_1 \xrightarrow{\sigma(T)} s_1'$ for some $(\sigma, T) \in \Sigma \times \mathbb{R}_{\geq 0}$, then $s_2 \xrightarrow{\sigma(T)} s'_2$ (resp. $s_2 \xrightarrow{\sigma(T')} s'_2$ for some $T' \in \mathbb{R}_{\geq 0}$) for some s'_2 with $(s'_1, s'_2) \in R$. A state s_2 *timed-simulates* (resp. *time-abstract-simulates*) a state s_1 if there exists a timed (resp. time-abstract) simulation R such that $(s_1, s_2) \in R$. In that case, we write $s_1 \sqsubseteq s_2$ (resp. $s_1 \sqsubseteq_{\text{t.a.}} s_2$).

Given two TTSs $\mathcal{S}$ and $\mathcal{T}$, we write $\mathcal{S} \sqsubseteq \mathcal{T}$ if the initial state of $\mathcal{T}$ timed-simulates that of $\mathcal{S}$ in their disjoint union. We write $\mathcal{S} \sqsubseteq_{\text{t.a.}} \mathcal{T}$ in case of an time-abstract simulation. For any state s of $\mathcal{S}$, we write ta-sim$_{\mathcal{T}}(s)$ for the set of states of $\mathcal{T}$ that time-abstract simulate s. This set is called *the (time-abstract) simulator set of s in $\mathcal{T}$*.

Given a finite set of clocks $\mathcal{C}$, we call *valuations* the elements of $\mathbb{R}^{\mathcal{C}}_{\geq 0}$. For a subset $R \subseteq \mathcal{C}$, a real number $\alpha \in \mathbb{R}_{\geq 0}$ and a valuation v, we write $v[R \leftarrow \alpha]$ for the valuation defined by $v[R \leftarrow \alpha](x) = v(x)$ for $x \in \mathcal{C} \setminus R$ and $v[R \leftarrow \alpha](x) = \alpha$ for $x \in R$. Given $d \in \mathbb{R}_{\geq 0}$, the valuation $v + d$ is defined by $(v + d)(x) = v(x) + d$ for all $x \in \mathcal{C}$. We extend these operations to sets of valuations in the obvious way.

Let $\mathbb{Q}_\infty = \mathbb{Q} \cup \{-\infty, \infty\}$. An *atomic clock constraint* is a formula of the form $k \leq x \leq l$ or $k \leq x - y \leq l$ where $x, y \in \mathcal{C}$ and $k, l \in \mathbb{Q}_\infty$. A *guard* is a conjunction of atomic clock constraints. We denote by $\Phi_{\mathcal{C}}$ the set of guards on the clock set $\mathcal{C}$. We define the *enlargement* of atomic clock constraints by $\delta \in \mathbb{Q}$ as follows: for $x, y \in \mathcal{C}$ and $k, l \in \mathbb{Q}_{>0}$, we let

$$\langle k \leq x - y \rangle_\delta = k - \delta \leq x - y, \qquad \langle x - y \leq l \rangle_\delta = x - y \leq l + \delta.$$

(and similarly for $\langle k \leq x \rangle_\delta$ and $\langle x \leq l \rangle_\delta$). The enlargement of a guard g, denoted by $\langle g \rangle_\delta$, is obtained by enlarging all its atomic clock constraints. Notice that δ can be negative here; this operation is then called *shrinking*. A valuation v *satisfies* a guard g, denoted $v \models g$, if all constraints are satisfied when each $x \in \mathcal{C}$ is replaced by $v(x)$. We denote by $[\![g]\!]$ the set of valuations that satisfy g. We will write $v \models_\delta g$ to mean $v \models \langle g \rangle_\delta$.

▶ **Definition 2.** A timed automaton $\mathcal{A}$ is a tuple $(\mathcal{L}, l_0, \mathcal{C}, \Sigma, E)$, with finite sets $\mathcal{L}$ of *locations*, $\mathcal{C}$ of *clocks*, Σ of *labels*, $E \subseteq \mathcal{L} \times \Phi_{\mathcal{C}} \times \Sigma \times 2^{\mathcal{C}} \times \mathcal{L}$ of edges, with $l_0 \in \mathcal{L}$ the *initial location*. An edge $e = (l, g, \sigma, R, l')$ is also written as $l \xrightarrow{g,\sigma,R} l'$. Guard g is called the *guard* of e.

For any timed automaton $\mathcal{A}$, let $(g_i)_{i \in I}$ denote the vector of all atomic clock constraints used in its guards. Given a vector of rational numbers $\overline{\delta} = (\delta_i)_{i \in I}$, we define $\mathcal{A}_{\overline{\delta}}$ as the timed automaton obtained from $\mathcal{A}$ by replacing g_i with $\langle g_i \rangle_{\delta_i}$. For any $\Delta \in \mathbb{Q}$, $\mathcal{A}_\Delta$ will denote the timed automaton where all guards are enlarged by Δ.

▶ **Definition 3.** The semantics of a timed automaton $\mathcal{A} = (\mathcal{L}, l_0, \mathcal{C}, \Sigma, E)$ is a TTS over alphabet Σ, denoted $[\![\mathcal{A}]\!]$, whose state space is $\mathcal{L} \times \mathbb{R}^{\mathcal{C}}_{\geq 0} \times \mathbb{R}_{\geq 0}$. The initial state is $(l_0, \overline{0}, 0)$, where $\overline{0}$ denotes the valuation where all clocks have value 0. There is a transition $(l, v, T) \xrightarrow{\sigma(T+\tau)} (l', v', T + \tau)$, for any edge $l \xrightarrow{g,\sigma,R} l'$ and $\tau \geq 0$, such that $v + \tau \models g$ and $v' = (v + \tau)[R \leftarrow 0]$.

We assume familiarity with the usual notions of *region equivalence* and *region automaton*, and refer to [3] for definitions. The important property used here is that time-abstract-simulating a timed automaton is equivalent to (time-abstract-)simulating its region automaton.

3 Shrinkability

3.1 Robustness and Shrinking

Robust model-checking, that is, the analysis of timed automata under clock imprecisions have been studied in [22, 14, 9, 10, 21, 23]. It is shown in [15] how this framework allows one to *validate the synchrony hypothesis*, that is, prove that the semantics is preserved in a physical implementation with imprecisions. See [22, 14] for examples of timed automata that are *not robust*: their behaviours change in presence of the slightest positive guard enlargement. In a recent work [8], we defined transformations that provide, for any given timed automaton $\mathcal{A}$, a timed automaton $\mathcal{A}'$ such that $\mathcal{A}$ and $\mathcal{A}'_\Delta$ are ϵ-bisimilar, that is, there is a timed bisimulation in which the differences in delays are bounded by ϵ at each step. The advantage of that approach is that it works for all timed automata, and that we obtain an ϵ-bisimilar enlarged timed automaton, for any desired $\epsilon > 0$. However, the timed behaviour of the resulting automaton may not be included in the abstract model; it is only preserved approximately. Also, the size of $\mathcal{A}'$ is exponential, and we do not make the link with an implementation semantics.

We now define *shrinking* of timed automata and show how it provides an alternative way to construct robust systems. Our method provides a construction of the same size as the initial automaton, and whose timed behaviour is always included in the abstract model. Our algorithms then allow to decide whether further properties, such as non-blockingness and time-abstract similarity, can be satisfied. In order to circumvent the effect of the imprecisions, we propose to *shrink* any guard of the form "$x \in [a, b]$" into "$x \in [a + \delta, b - \delta]$" for some $\delta > 0$, so that under a small enlargement parameter $\Delta > 0$, we have $[a + \delta - \Delta, b - \delta + \Delta] \subseteq [a, b]$; in other terms, the satisfaction of the enlarged guard implies the satisfaction of the original guard. Formally, we will consider the *shrunk timed automaton* $\mathcal{A}_{-\mathbf{k}\delta}$ where $\mathbf{k} = (k_i)_{i \in I} \in \mathbb{N}^I_{>0}$ and $\delta > 0$. Clearly, if $\Delta < \max_{i \in I}(k_i) \cdot \delta$, $\mathcal{A}_{-\mathbf{k}\delta+\Delta}$ does not contain more behaviours than $\mathcal{A}$; in fact $[\![\mathcal{A}_{-\mathbf{k}\delta+\Delta}]\!] \sqsubseteq [\![\mathcal{A}]\!]$.

Shrinking is a natural idea when one is interested in the preservation of strict timing constraints, such as critical deadlines. However, shrinking may remove too many behaviours and the resulting automaton may even become blocking. We are interested in deciding the existence of shrinking parameters $\mathbf{k}$ and δ, and in their computation, for which the shrunk timed automaton is non-blocking and/or is able to time-abstract simulate the original automaton. We will see in Section 6 that when the shrunk automaton satisfies these properties, these are preserved in a concrete implementation semantics.

▶ **Definition 4.** A timed automaton $\mathcal{A}$ is *shrinkable* if there exists $\mathbf{k} \in \mathbb{N}^I_{>0}$ and $\delta_0 \in \mathbb{Q}_{>0}$ such that for all $0 \le \delta \le \delta_0$,

- $[\![\mathcal{A}_{-\mathbf{k}\delta}]\!]$ is non-blocking,
- $[\![\mathcal{A}]\!] \sqsubseteq_{\text{t.a.}} [\![\mathcal{A}_{-\mathbf{k}\delta}]\!]$, with the following additional technical requirement: for each region (l, r) of $\mathcal{R}(\mathcal{A})$, there is a guard g and a vector $\mathbf{h}$ of non-negative integers such that for all $0 \le \delta \le \delta_0$, the simulator set of (l, r) in $[\![\mathcal{A}_{-\mathbf{k}\delta}]\!]$ equals $[\![\langle g \rangle_{-\mathbf{h}\delta}]\!]$.

We say that $\mathcal{A}$ is *shrinkable w.r.t. non-blockingness* (resp. *w.r.t. simulation*) if only the first (resp. second) condition holds. The above $\mathbf{k}$ and δ_0 are shrinking parameters for $\mathcal{A}$.

We define shrinkability "for all $0 \le \delta \le \delta_0$", so if an automaton is shrinkable, we require it to remain correct when imprecisions are reduced, that is when δ is chosen smaller. In fact, the shrunk automaton can be seen as an approximation of the initial automaton, and we would like to be able to obtain arbitrarily close correct approximations by only adjusting δ. This requirement is also related to the property called "faster-is-better" [2, 15]. Notice also that when a timed automaton is shrinkable w.r.t. simulation, then we require that for all small enough δ, each simulator set can be expressed as shrinkings $\langle g \rangle_{-\mathbf{h}\delta}$ where $\mathbf{h}$ is the same for all δ (that is, parameters $\mathbf{h}$ are *uniform*). Then, when we adjust the parameter δ, the expressions of the simulator sets do not change. This is desirable for instance when one needs to use these constraints in the system.

3.2 Shrinking as a Remedy to Unrealistic Behaviour

Shrinkability also excludes *unrealistic* timing constraints, such as Zeno behaviours. In fact, for any timed automaton $\mathcal{A}$, consider the automaton $\mathcal{A}'$ obtained from $\mathcal{A}$ by adding a new clock u, the constraint $u \geq 0$ and the reset $u := 0$ at every edge. Clearly, $\mathcal{A}$ and $\mathcal{A}'$ are isomorphic. If automaton $\mathcal{A}'$ is shrinkable, then $\mathcal{A}$ does not need Zeno strategies to satisfy the properties proven for the exact semantics and preserved by time-abstract similarity (in fact, each $u \geq 0$ is shrunk to some $u \geq \delta_i$ with $\delta_i > 0$).

But unrealistic timing constraints are not limited to Zeno behaviours. The automaton in Fig. 1 provides an example of a timed automaton which is non-blocking for $\delta_1 = \delta_2 = \delta_3 = 0$, and lets the time diverge but it becomes blocking whenever $\delta_2 > 0$ or $\delta_3 > 0$, so it is not shrinkable. A similar example was provided in [11] but with equality constraints, so it is trivially not shrinkable. In section 5, we give an example of a shrinkable timed automaton (Fig. 3).

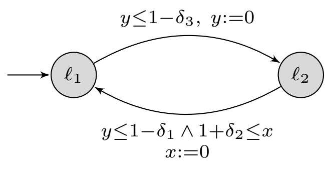
$y{\leq}1{-}\delta_3,\ y{:=}0$
ℓ_1
ℓ_2
$y{\leq}1{-}\delta_1 \wedge 1{+}\delta_2{\leq}x$
$x{:=}0$

Figure 1 A shrunk timed automaton that is blocking whenever $\delta_2 > 0$ or $\delta_3 > 0$.

3.3 Decidability of Shrinkability

Our main result is the decidability of shrinkability:

▶ **Theorem 5.** *Shrinkability w.r.t. non-blockingness can be decided in* PSPACE, *and in* NP *if the number of outgoing transitions from each location is bounded. Shrinkability w.r.t. simulation is decidable in* EXPTIME. *Finally, shrinkability is decidable in* EXPTIME.

Moreover, we will show that when a given timed automaton is shrinkable, the least shrinking parameters can be computed (see Section 5 for details). In the rest, we present the proof of this result. We begin by defining parametric *difference-bound matrices* (DBMs) and give tools for solving fixpoint equations on DBMs through max-plus equations. We then explain how this can be used to decide shrinkability. In Section 6, we present a concrete implementation semantics and prove that non-blockingness and simulation are preserved in this semantics for all shrinkable timed automata.

4 Some algebraic tools

4.1 Parameterized Difference Bound Matrices

Difference bound matrices are data structures used to represent sets of clock valuations in timed automata analysis [17]. Write $\mathcal{C} = \{1, \ldots, C\}$, and add an artificial clock of index 0, that has constant value 0. We let $\mathcal{C}_0 = \mathcal{C} \cup \{0\}$. A *difference bound matrix* (DBM) over $\mathcal{C}_0$ is an element of $\mathcal{M}_{C+1}(\mathbb{Q}_\infty)$[1]. Each $M \in \mathcal{M}_{C+1}(\mathbb{Q}_\infty)$ defines a *zone*, that is, a convex subset of $\mathbb{R}^{\mathcal{C}}_{\geq 0}$ defined by $[\![M]\!] = \{v \in \mathbb{R}^{\mathcal{C}}_{\geq 0} \mid \forall x, y \in \mathcal{C}_0, -M_{y,x} \leq v(x) - v(y) \leq M_{x,y}\}$. Clearly, each DBM can be equivalently described by a guard, and conversely. A DBM M is *normalized* when for all $x, y, z \in \mathcal{C}_0$, it holds $M_{x,y} \leq M_{x,z} + M_{z,y}$. Any non-empty DBM can be made normalized in polynomial time, by interpreting it as an adjacency matrix of a weighted graph and computing all shortest paths between any two clocks.

[1] $\mathcal{M}_n(X)$ is the set of $n \times n$ matrices with coefficients in X, where all diagonal coefficients are 0.

We define several *elementary operations* on DBMs. Given a DBM M, we let $\mathrm{Pre}_{\mathsf{time}}(M)$ be the normalized DBM that describes the time predecessors of $[\![M]\!]$, *i.e.*, $[\![\mathrm{Pre}_{\mathsf{time}}(M)]\!] = \{v \in \mathbb{R}^{\mathcal{C}}_{\geq 0} \mid \exists t \in \mathbb{R}_{\geq 0} \text{ s.t. } v + t \in [\![M]\!]\}$. Given $R \subseteq \mathcal{C}$, we let $\mathrm{Unreset}_R(M)$ be the normalized DBM that defines $\{v \in \mathbb{R}^{\mathcal{C}}_{\geq 0} \mid v[R \leftarrow 0] \in [\![M]\!]\}$. For two DBMs M and N, we write $M \cap N$ for the normalized DBM describing $[\![M]\!] \cap [\![N]\!]$. A function $f\colon \mathcal{M}_{C+1}(\mathbb{Q}_\infty)^n \to \mathcal{M}_{C+1}(\mathbb{Q}_\infty)$ (for some $n > 0$), is said *elementary* if it combines its arguments using elementary operations. Efficient algorithms exist for computing these operations on DBMs [6, 12].

We extend standard DBMs in order to manipulate sets of states in shrunk timed automata. We fix a tuple of parameters $\mathbf{k} = (k_i)_{i\in I}$, which will take nonnegative integer values. The *max-plus polynomials* over $\mathbf{k}$, denoted by $\mathcal{G}(\mathbf{k})$, are generated by the grammar $\phi ::= l \in \mathbb{N} \mid k_i,\ i \in I \mid \phi + \phi \mid \max(\phi, \phi)$. For any max-plus polynomial ϕ and valuation $\nu\colon \mathbf{k} \to \mathbb{N}$, we denote by $\phi[\nu]$ the value of formula ϕ replacing each parameter k by $\nu(k)$. A (resp. *positive*, *parameterized*) *shrinking matrix* is an element of $\mathcal{M}_{C+1}(\mathbb{N})$ (resp. $\mathcal{M}_{C+1}(\mathbb{N}_{>0})$, $\mathcal{M}_{C+1}(\mathcal{G}(\mathbf{k}))$). If P is a parameterized shrinking matrix (PSM) and ν is a valuation, the shrinking matrix $P[\nu]$ is defined in a natural way. Note that our definition of PSM is different from parametric DBMs considered for instance in [20], since we use max-plus polynomials instead of linear expressions and only consider natural number valuations. In what follows, we manipulate parameterized DBMs, also called *shrunk DBMs*, of the form $M - \delta \cdot P$, where δ is a fresh parameter, and P is a PSM. If M is a DBM for guard g, the shrunk guard $\langle g\rangle_{-\delta}$ will be represented by the shrunk DBM $M - \mathbf{1} \cdot \delta$, where matrix $\mathbf{1}$ has 0's on the diagonal and 1's everywhere else.

Shrunk DBMs will be used as a data structure for manipulating the state space of shrunk timed automata. The following lemma explains how elementary operations can be computed on shrunk DBMs. In particular, it shows how shrinking parameters (*i.e.*, the PSM) can be propagated in a backward analysis while staying in the max-plus theory.

▶ **Lemma 6.** *Let $M, M_1, \ldots, M_n$ be non-empty normalized DBMs and $f\colon \mathcal{M}_{C+1}(\mathbb{Q}_\infty)^n \to \mathcal{M}_{C+1}(\mathbb{Q}_\infty)$ be an elementary function with $M = f(M_1, \ldots, M_n)$. Let $P_1, \ldots, P_n$ be matrices in $\mathcal{M}_{C+1}(\mathcal{G}(\mathbf{k}))$. Then, we can compute $P' \in \mathcal{M}_{C+1}(\mathcal{G}(\mathbf{k}))$ s.t. for all $\nu\colon \mathbf{k} \to \mathbb{N}$, there exists a (computable) $\delta_0 > 0$ s.t. $M - \delta P'[\nu] = f(M_1 - \delta P_1[\nu], \ldots, M_n - \delta P_n[\nu])$ for all $0 \leq \delta < \delta_0$. All these computations can be achieved in polynomial time, and in particular P' has size polynomial in the size of $P_1, \ldots, P_n$ and f. If the above property holds, we write $M - \delta P' = f(M_1 - \delta P_1, \ldots, M_n - \delta P_n)$.*

For solving the shrinkability problems, we will use fixpoint equations on shrunk DBMs (see Section 5). As a prerequisite, we therefore first investigate max-plus equations, and then give a general theorem for solving those equations.

4.2 Max-plus equations

In PSMs, formal expressions using maximization and sum are manipulated. The set $\mathbb{R}_{\geq 0}$ endowed with these operations is called the *max-plus algebra*. There is a well-established theory on solving equations in this algebra, with applications to discrete-event systems [5].

Let $k_1, \ldots, k_n, k_{n+1}, \ldots, k_{n+n'}$ be parameters, and $\phi_1, \ldots, \phi_n$ be max-plus polynomials. We will be interested in computing solutions of fixpoint equations of the following form:

$$k_i = \phi_i(k_1, \ldots, k_n, k_{n+1}, \ldots, k_{n+n'}), \quad \forall 1 \leq i \leq n. \tag{E}$$

Notice that variables $k_{n+1}, \ldots, k_{n+n'}$ only appear at the right hand side of the equation. Equation (E) defines a *non-linear* equation (polynomials ϕ_i have arbitrary degrees). Although Tarski's Theorem [24] guarantees the existence of fixpoint solutions in $\mathbb{N} \cup \{\infty\}$, we are interested in *finite* solutions, *i.e.*, solutions in $\mathbb{N}$ which is not a complete lattice.

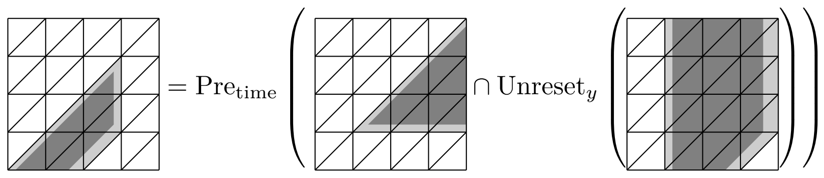

Figure 2 Consider an edge $\ell \xrightarrow{g,\sigma,R} \ell'$ in a timed automaton where $g = 1 \leq y \wedge 0 \leq x - y$ and $R = \{y\}$. For any pair of zones X, Y, the equation $Y = \mathrm{Pre}_{\mathsf{time}}(\llbracket g \rrbracket \cap \mathrm{Unreset}_y(X))$ expresses the fact that X can be reached in one step starting from Y. Consider $Y = \llbracket 0 \leq x, y \leq 3 \wedge 0 \leq x - y \leq 2 \rrbracket$, and $X = \llbracket 1 \leq x \leq 4 \wedge x - y \leq 3 \rrbracket$. In the figure, the union of dark gray and light gray areas illustrate this equation while the dark gray areas illustrate the equation between *shrunk* zones. Let us assume that we shrink g to $g' = 1 + k_1\delta \leq y \wedge k_2\delta \leq x - y$ and X to $X' = \llbracket 1 + k_3\delta \leq x \leq 4 - k_4\delta \wedge x - y \leq 3 - k_5\delta \rrbracket$, for positive integers k_i and $\delta > 0$ small enough so that these sets are non-empty. Then, by Lemma 6, we can compute the shrinking parameters for Y, so that the shrunk zone Y' satisfies the new equation $Y' = \mathrm{Pre}_{\mathsf{time}}(\llbracket g' \rrbracket \cap \mathrm{Unreset}_y(X'))$. We get:

$$\begin{array}{ll}
\mathrm{Unreset}_y(X') & = \llbracket 1 + k_3\delta \leq x \leq 3 - k_5\delta \rrbracket, \\
\llbracket g' \rrbracket \cap \mathrm{Unreset}_y(X') & = \llbracket 1 + \max(k_1 + k_2, k_3)\delta \leq x \leq 3 - k_5\delta \\
 & \quad \wedge 1 + k_1\delta \leq y \leq 3 - (k_2 + k_5)\delta \wedge k_2\delta \leq x - y \leq 2 - (k_1 + k_5)\delta \rrbracket, \\
Y' = \mathrm{Pre}_{\mathsf{time}}(\llbracket g' \rrbracket \cap \mathrm{Unreset}_y(X')) & = \llbracket k_2\delta \leq x \leq 3 - k_5\delta \wedge 0 \leq y \leq 3 - (k_2 + k_4)\delta \\
 & \quad \wedge k_2\delta \leq x - y \leq 2 - (k_1 + k_5)\delta \rrbracket.
\end{array}$$

This equality holds for all $0 \leq \delta < \min(\frac{1}{k_4 - k_5}, \frac{2}{\max(k_1+k_2,k_3)+k_5}, \frac{3}{k_2+k_5}, \frac{2}{k_1+k_2+k_5}, \frac{2}{k_1-k_2+k_5})$, where a term is $+\infty$ if the denominator is zero.

▶ **Theorem 7.** *For any equation of the form* (E)*, the existence of a solution in* $\mathbb{N}$ *is decidable in polynomial time in the size of the equation. Moreover, assume there is a solution in* $\mathbb{N}$ *in which* $k_{n+1}, \ldots, k_{n+n'}$ *take positive values; then given any fixed positive values* $v_{n+1}, \ldots, v_{n+n'}$*, Equation* (E) *with the additional constraints* $k_{n+i} = v_{n+i}$ *for all* $1 \leq i \leq n'$ *has a least solution, computable in polynomial time.*

The second point of Theorem 7 states that the existence of solutions with positive values for the unconstrained variables does not depend on their exact values. These results rely on an analysis of max-plus graphs, that we associate to max-plus equations.

4.3 Equations on shrunk DBMs

We now apply the previous results to solving equations on shrunk DBMs. We consider fixpoint equations on DBMs of the form:

$$M_i = f_i(M_1, \ldots, M_n, M_{n+1}, \ldots, M_{n+n'}), \quad \forall 1 \leq i \leq n, \tag{1}$$

where $M_1, \ldots, M_{n+n'}$ are unknown normalized DBMs ($M_{n+1}, \ldots, M_{n+n'}$ are unconstrained) and f_i's are elementary functions. We are interested in *shrunk solutions* defined as follows.

▶ **Definition 8.** Fix a solution $(M_i)_i$ of (1). A *shrunk solution* of (1) w.r.t. $(M_i)_i$ is a triple $\big((M_i)_i, (Q_i)_i, \delta_0\big)$, where $\delta_0 > 0$ and Q_i's are shrinking matrices such that for all $0 \leq \delta \leq \delta_0$, $(M_i - \delta Q_i)_i$ is a solution of (1). A shrunk solution is called the *greatest shrunk solution* if $(Q_i)_i$ are the least shrinking matrices which define a shrunk solution w.r.t. $(M_i)_i$.

Assume that (1) has a solution and fix one, say $(M_i)_i$. From Lemma 6, there exist matrices $(\phi_i)_{1 \leq i \leq n}$ of max-plus polynomials s.t. for all shrinking matrices $(Q_i)_i$, there exists $\delta_0 > 0$ such that $M_j - \phi_j((Q_i)_i) \cdot \delta = f_j((M_i - Q_i \cdot \delta)_i)$ for all $0 \leq \delta \leq \delta_0$ and all $1 \leq j \leq n$. This suggests that we study the following fixpoint equation on PSMs P_i's, where each coefficient is a fresh parameter and polynomials ϕ_i's are those from Lemma 6 for these P_i's:

$$P_i = \phi_i(P_1, \ldots, P_n, P_{n+1}, \ldots, P_{n+n'}), \quad \forall 1 \leq i \leq n. \tag{2}$$

This is a max-plus equation (like (E)), whose size is polynomial in the size of (1). The following lemma links the shrunk solutions of (1) with the solutions of (2).

▶ **Lemma 9.** *Fix any solution $(M_i)_i$ of (1) and consider max-plus polynomial matrices $(\phi_i)_{1 \leq i \leq n}$ as defined above. Then,*

- *For all shrinking matrices $(Q_i)_i$, there exists $\delta_0 > 0$ s.t. $\big((M_i)_i, (Q_i)_i, \delta_0\big)$ is a shrunk solution of (1) if, and only if, $(Q_i)_i$ is a solution of (2) in $\mathbb{N}$.*
- *$\big((M_i)_i, (Q_i)_i, \delta_0\big)$ is the greatest shrunk solution of (1) iff $(Q_i)_i$ is the least solution of (2).*
- *If (2) has a solution $(Q_i)_i$ where $Q_{n+1}, \ldots, Q_{n+n'}$ have positive coefficients (except for 0 on the diagonal), then for any matrices $R_{n+1}, \ldots, R_{n+n'}$ in $\mathcal{M}_{C+1}(\mathbb{N}_{>0})$, (2) with the additional constraints $P_{n+i} = R_{n+i}$ for all $1 \leq i \leq n'$ has a least shrunk solution, computable in polynomial time.*

Notice that by Lemma 9, one can also decide the existence of a solution of (2), where all $Q_{n+1}, \ldots, Q_{n+n'}$ are positive shrinking matrices: it suffices to add to (2) the equalities $Q_{n+i} = \mathbf{1}$, where $\mathbf{1}$ is the matrix with 1's everywhere except for 0's on the diagonal.

5 Deciding shrinkability

We now apply the results we developed in previous sections to shrinkability. We fix a non-blocking timed automaton $\mathcal{A} = (\mathcal{L}, l_0, \mathcal{C}, \Sigma, E)$. We assume that all edges of $\mathcal{A}$ have distinct labels, and identify edges with their labels. This is harmless for our purpose since we compare an automaton to its shrinking that has the same structure. For any edge with label $\sigma \in \Sigma$ and guard g_σ, let G_σ be the DBM that represents $[\![g_\sigma]\!]$, and R_σ the reset set.

5.1 Shrinkability w.r.t. simulation.

Thanks to the hypothesis on distinct edge labels, the simulator sets of regions (l, r) in $[\![\mathcal{A}]\!]$ can be expressed by the following fixpoint equation:

$$[\![M_{l,r}]\!] = \bigcap_{\sigma \in \Sigma} \; \bigcap_{(l,r) \xrightarrow{\sigma} (l',r')} \mathrm{Pre}_{\mathsf{time}}(\mathrm{Unreset}_{R_\sigma}([\![M_{l',r'}]\!]) \cap [\![G_\sigma]\!]), \tag{3}$$

for all $(l, r) \in \mathcal{R}(\mathcal{A})$, where $(M_{l,r})_{l,r}$ are the unknown DBMs. In the greatest solution, each $M_{l,r}$ represents the simulator set of region (l, r) in $[\![\mathcal{A}]\!]$. Consider the greatest solution $(M_{l,r})_{l,r} \cup (G_\sigma)_\sigma$, where we see G_σ's as a part of the solution. Let $(l_0, \vec{0})$ denote the initial state of $\mathcal{R}(\mathcal{A})$. Solving shrinkability means deciding whether (3) has a shrunk solution with respect to $(M_{l,r})_{l,r} \cup (G_\sigma)_\sigma$, such that $M_{l_0,\vec{0}}$ is shrunk to a zone that contains $\vec{0}$ (since $(l_0, \vec{0})$ is the initial state of any shrinking of $\mathcal{A}$). For any shrinking matrix $P_{l_0,\vec{0}}$, it can be checked in polynomial time whether $\vec{0}$ belongs to $[\![M_{l_0,\vec{0}} - \delta P_{l_0,\vec{0}}]\!]$ for sufficiently small δ. Now, if there is a shrunk solution to (3), then for any positive shrinking matrices $(K_\sigma)_\sigma$ Lemma 9 provides a (greatest) shrunk solution where the shrinking matrices for $(G_\sigma)_\sigma$ are fixed to $(K_\sigma)_\sigma$. In particular, shrinkability w.r.t. simulation does not depend on how much guards are shrunk: either all positive integer vectors $\mathbf{k}$ witness the shrinkability of $\mathcal{A}$ (into $\mathcal{A}_{-\mathbf{k}\delta}$), or $\mathcal{A}$ is not shrinkable w.r.t. simulation for any value of $\mathbf{k}$.

The simulator sets $M_{l,r}$ can be computed in exponential time ([18]), and Equation (3) has size polynomial in the size of these sets. By Lemma 9, the overall complexity is in EXPTIME.

5.2 Shrinkability w.r.t. non-blockingness.

Since automaton $\mathcal{A}$ is non-blocking, $(G_\sigma)_\sigma$ satisfies the following equation.

$$\forall \sigma \in \Sigma, \quad [\![G_\sigma]\!] \subseteq \bigcup_{\sigma' : (\sigma,\sigma') \in \Sigma_{E \circ E}} \mathrm{Unreset}_{R_\sigma}(\mathrm{Pre}_{\mathsf{time}}([\![G_{\sigma'}]\!])), \tag{4}$$

where we let $\Sigma_{E \circ E} = \{(\sigma, \sigma') \mid \exists l, l', l'' \in \mathcal{L}, l \xrightarrow{\sigma} l' \xrightarrow{\sigma'} l'' \in E\}$, that is the set of pairs of labels of consecutive transitions in $\mathcal{A}$. We rewrite this equivalently as follows.

$$\forall \sigma \in \Sigma, \quad [\![G_\sigma]\!] = \bigcup_{\sigma' : (\sigma,\sigma') \in \Sigma_{E \circ E}} \mathrm{Unreset}_{R_\sigma}(\mathrm{Pre}_{\mathsf{time}}([\![G_{\sigma'}]\!])) \cap [\![G_\sigma]\!], \tag{5}$$

Now, $\mathcal{A}$ is shrinkable w.r.t. non-blockingness if, and only if, this equation has a shrunk solution w.r.t. $(G_\sigma)_\sigma$. We can unfortunately not directly use our general results on shrunk solutions since our equation contains a union. We instead apply transformations to this equation in order to remove the union. We start by rewriting the above equation as follows:

$$\forall \sigma \in \Sigma, \quad [\![G_\sigma]\!] = \bigcup_{\sigma' : (\sigma,\sigma') \in \Sigma_{E \circ E}} [\![M_{\sigma,\sigma'}]\!], \qquad [\![M_{\sigma,\sigma'}]\!] = \mathrm{Unreset}_{R_\sigma}(\mathrm{Pre}_{\mathsf{time}}([\![G_{\sigma'}]\!])) \cap [\![G_\sigma]\!]. \tag{6}$$

Fix a solution $(G_\sigma)_\sigma \cup (M_{\sigma,\sigma'})_{\sigma,\sigma'}$, which exists again by the non-blockingness assumption. We solve the max-plus equation corresponding to the right part of (6) by Lemma 9, but we will add to this equation some inequalities which "encode" the left part of (6). We use the following technical lemma to choose these inequalities.

▶ **Lemma 10.** *Let $C_1, \ldots, C_b$ and D be normalized DBMs s.t. $[\![D]\!] = \bigcup_{1 \le i \le b} [\![C_i]\!]$ and $P_1, \ldots, P_b$ and Q shrinking matrices s.t. for some $\delta_0 > 0$, $D - \delta Q$ and $C_i - \delta P_i$ are normalized for all $\delta \in [0, \delta_0]$. Then, one can decide the existence of (and then compute) some $\delta_1 > 0$ s.t. $[\![D - \delta Q]\!] = \cup_{1 \le i \le b} [\![C_i - \delta P_i]\!]$ for all $0 < \delta < \min(\delta_0, \delta_1)$, in polynomial space and in time $O(|\mathcal{C}_0|^{2b} p(|\mathcal{A}|))$, where $p(\cdot)$ is a polynomial.*

Moreover, in this case, for all shrinking matrices $Q', P'_1, \ldots, P'_b$ s.t. $Q_{x,y} \bowtie (P_i)_{x,y} \Leftrightarrow Q'_{x,y} \bowtie (P'_i)_{x,y}$ and $(P_i)_{x,y} \bowtie (P_j)_{x,y} \Leftrightarrow (P'_i)_{x,y} \bowtie (P'_j)_{x,y}$ for all $i, j \in \{1, \ldots, b\}$, $x, y \in \mathcal{C}_0$ and $\bowtie \in \{<, =\}$, it holds $[\![D - \delta Q']\!] = \cup_{1 \le i \le b} [\![C_i - \delta P'_i]\!]$ for all small enough $\delta > 0$.

Note that checking equality between a zone and a union of zones is a difficult problem; some heuristics were suggested in [13]. The second point of the lemma says that the satisfaction of the left part of (6) by a shrunk solution only depends on the relative ordering of the coefficients of the shrinking matrices. Therefore, we only need to guess the ordering between all parameters (there is at least one if there exists a shrunk solution), and solve the right part of (6) augmented with these guessed (in)equalities.

Formally, let Φ be the max-plus equation corresponding to the right part of (6), as given by Lemma 9. Let $\mathbf{k}'$ denote the set of all parameters that appear in Φ (there is one parameter per element of each matrix G_σ and $M_{\sigma,\sigma'}$). Notice that $\mathbf{k}'$ has size $O\big((|\mathcal{C}_0| \cdot |\mathcal{L}| \cdot b)^2\big)$, where b is the maximal number of outgoing edges in $\mathcal{A}$, and that Φ has size polynomial in the size of $\mathcal{A}$. Φ is a conjunction of equations $k = \phi_k(\mathbf{k}')$ for all $k \in \mathbf{k}'$. For all pairs $k, l \in \mathbf{k}'$, we guess a relation among $\{<, =, >\}$, and define equation Φ' by adding these relations to Φ. This can be done, for the case $k = l$, by replacing the constraints on k and l respectively by $k = \max(\phi_k(\mathbf{k}'), l)$ and $l = \max(\phi_k(\mathbf{k}'), k)$, and in the case $k > l$, by replacing the constraint on k by $k = \max(\phi_k(\mathbf{k}'), l + 1)$. Notice that Φ' is obtained from Φ in polynomial time and with a polynomial number of guesses. We then solve Φ' using Theorem 7. If we find a solution, say $(P_\sigma)_\sigma \cup (P_{\sigma,\sigma'})_{\sigma,\sigma'}$, we verify that $[\![G_\sigma - \delta P_\sigma]\!] = \cup_{\sigma'} [\![M_{\sigma,\sigma'} - \delta P_{\sigma,\sigma'}]\!]$ for small δ, for all pairs $(\sigma, \sigma') \in \Sigma_{E \circ E}$, in time $O(|\mathcal{C}_0|^{2b} p(|\mathcal{A}|))$ and in polynomial space by Lemma 10.

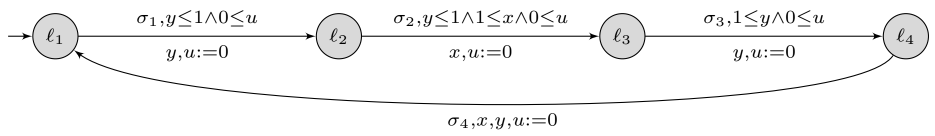
ℓ_1
ℓ_2
ℓ_3
ℓ_4
$\sigma_1, y \le 1 \wedge 0 \le u$
$y, u := 0$
$\sigma_2, y \le 1 \wedge 1 \le x \wedge 0 \le u$
$x, u := 0$
$\sigma_3, 1 \le y \wedge 0 \le u$
$y, u := 0$
$\sigma_4, x, y, u := 0$

Figure 3 A shrinkable timed automaton. One can in fact shrink guards g_{σ_i} into $g'_{\sigma_1} = 3\delta \le x \wedge y \le 1 - \delta \wedge y - x \le 1 - 4\delta \wedge f(\delta)$, $g'_{\sigma_2} = 1 + \delta \le x \wedge y \le 1 - 2\delta \wedge 3\delta \le x - y \wedge f(\delta)$ and $g'_{\sigma_3} = y \le 1 - \delta \wedge f(\delta)$, where $f(\delta) = \delta \le u \wedge y - u \le 1 - 2\delta$, The resulting shrunk automaton can be seen to be non-blocking and time-abstract similar to $\mathcal{A}$ for any $\delta \in [0, \frac{1}{4}]$. Notice how additional constraints appear in the guards.

We accept if all verifications succeed and reject otherwise. If accepted, any solution provides a shrunk solution of (6), by Lemma 9. Conversely, if there is a shrunk solution of (6), then, Φ' can be constructed for the guesses corresponding to this solution, and by Lemma 10, Φ' has a solution. If b is fixed, this procedure is in NP. Otherwise, instead of making guesses, we can deterministically try all possible guesses (the number of possible guesses is $O(2^{(|\mathcal{C}|\cdot|\mathcal{L}|\cdot b)^2})$ and verify in polynomial space, so the procedure is then in PSPACE.

Finally, to decide shrinkability, one can first compute parameters $\mathbf{k}$ and δ_0 for non-blockingness, then check shrinkability w.r.t. simulation since the latter does not depend on $\mathbf{k}$ and δ_0. Figure 3 shows an example of a shrinkable timed automaton.

6 Implementation Semantics

In this section, we present an *implementation semantics*, which takes into account reaction times and clock imprecisions. Our semantics corresponds to the execution of timed automata by a digital system that has a single digital clock and nonzero reaction time. Our semantics is closely related to the one studied in [15] with minor differences, but we prove additional properties besides the one given there. We first define our semantics and state its properties, then compare it with [15], and with other related work.

We describe a system which interacts, via sending and receiving signals, with a physical environment (*e.g.* via sensors). We distinguish input and output actions, and define the transitions of the system taking into account the imprecisions of the clock, the transmission delay of signals and the reaction time of the system. When an event is generated at time T by the environment, it is treated by the system at time $T + \epsilon$, for some $\epsilon > 0$ which will be bounded but unpredictable. Similarly, when the environment receives a signal at time T, it must have been sent at some time $T - \epsilon$. We assume that the system ignores any signal that is received during the treatment of the previous signal; this *reaction time* will be also bounded but unpredictable. We define the timestamps of both input and output actions as the reaction times of the environment, since we are interested in the behaviour of the environment controlled by a digital timed system.

The implementation semantics has three parameters: a) Δ_c is the clock period, b) Δ_r is the maximum *reaction time*, following each action, c) Δ_t is the maximum *transmission delay* of signals between the system and the environment (ϵ above). We suppose the system has a Δ_c-periodic clock, whose value, at any real time T, is $\lfloor T \rfloor_{\Delta_c} = \max_{k \ge 0}\{k\Delta_c \mid k\Delta_c \le T\}$.

▶ **Definition 11.** Let $\mathcal{A} = (\mathcal{L}, \ell_0, \mathcal{C}, \Sigma, E)$ be a TA with $\Sigma = \Sigma_{\mathsf{in}} \cup \Sigma_{\mathsf{out}}$, and $\Delta_r, \Delta_c, \Delta_t > 0$. The *implementation semantics* $[\![\mathcal{A}]\!]^{\mathsf{Impl}}$ is the TTS $(S_{\mathcal{A}}, s_0, \Sigma, E)$ in which states are tuples (ℓ, T, v, u_0): ℓ is a location, $T \in \mathbb{R}_{\ge 0}$ the current real time, $v \in \mathbb{R}^{\mathcal{C}}_{\ge 0}$ the timestamp of the latest reset for each clock, and $u_0 \in [0, \Delta_r]$ the reaction time following the latest location change. From any state (ℓ, T, v, u_0), for any edge $\ell \xrightarrow{\sigma,g,R} \ell'$ and $T' \geq T$, we let,

- if $\sigma \in \Sigma_{\mathsf{in}}$, $(\ell, T, v, u_0) \xrightarrow{\sigma(T')} (l', T'+\epsilon, v[R \leftarrow T'+\epsilon], u_0')$, whenever $\lfloor T'+\epsilon \rfloor_{\Delta_c} - \lfloor v \rfloor_{\Delta_c} \models g$ and $T' + \epsilon \geq T + u_0$, where $(\epsilon, u_0') \in [0, \Delta_t] \times [0, \Delta_r]$ is chosen non-deterministically,
- if $\sigma \in \Sigma_{\mathsf{out}}$, $(\ell, T, v, u_0) \xrightarrow{\sigma(T')} (\ell', T', v[R \leftarrow T' - \epsilon], u_0')$, whenever $\lfloor T' - \epsilon \rfloor_{\Delta_c} - \lfloor v \rfloor_{\Delta_c} \models g$, $\epsilon < (T' - T)$, and $T' - \epsilon \geq T + u_0$, where $(\epsilon, u_0') \in [0, \Delta_t] \times [0, \Delta_r]$ is chosen non-deterministically.

Notice that ϵ and u_0 are bounded by known values but are unpredictable, so they cannot be chosen by the system. We will consider *scheduler* functions ρ, which, depending on the history of a given run, chooses (ϵ, u_0) at each transition. For any scheduler ρ, we denote by $[\![\mathcal{A}]\!]_\rho^{\mathsf{Impl}}$ the implementation semantics, where (ϵ, u_0) is given by ρ at each transition. We will not formally define ρ here, but it can be done without difficulty.

The following proposition states the relation between the exact semantics and the implementation semantics of timed automata. All properties hold under *any* scheduler ρ. For any TTS $\mathcal{T}$, let us write $\mathcal{T}^{\geq \alpha}$, the TTS obtained from $\mathcal{T}$ where consecutive transitions are separated by at least α time units.

▶ **Proposition 12.** *Let $\mathcal{A}$ be a timed automaton s.t. $[\![\mathcal{A}]\!]$ is non-blocking, and $\Delta_r, \Delta_c, \Delta_t > 0$. Then, for any $\Delta \geq 2\Delta_c + 4\Delta_t + \Delta_r$ and scheduler ρ, $[\![\mathcal{A}_\Delta]\!]_\rho^{\mathsf{Impl}}$ is non-blocking and,*

$$[\![\mathcal{A}]\!]^{\geq 2\Delta_r + \Delta_t} \sqsubseteq [\![\mathcal{A}_\Delta]\!]_\rho^{\mathsf{Impl}} \sqsubseteq [\![\mathcal{A}_{\Delta+2\Delta_c+4\Delta_t}]\!].$$

Now, for any timed automaton $\mathcal{A}$, consider $\mathcal{A}'$ as defined in Section 3.2. We have

$$[\![\mathcal{A}'_{-\mathbf{k}\delta}]\!] \quad \sqsubseteq \quad [\![\mathcal{A}'_{-\mathbf{k}\delta+\Delta}]\!]_\rho^{\mathsf{Impl}} \quad \sqsubseteq \quad [\![\mathcal{A}'_{-\mathbf{k}\delta+\Delta'}]\!] \quad \sqsubseteq \quad [\![\mathcal{A}']\!] = [\![\mathcal{A}]\!],$$

whenever $\Delta \geq 2\Delta_c + 4\Delta_t + \Delta_r$, $\Delta' = \Delta + 2\Delta_c + 4\Delta_t$ and $\delta \geq \max(2\Delta_r + \Delta_t, \Delta')$. In fact, $[\![\mathcal{A}'_{-\mathbf{k}\delta}]\!]^{\geq 2\Delta_r+\Delta_t}$ is equal to $[\![\mathcal{A}'_{-\mathbf{k}\delta}]\!]$ whenever $\delta \geq 2\Delta_r + \Delta_t$, and the rightmost simulation is due to the fact that $-\mathbf{k}\delta + \Delta' < 0$. Thus, given appropriate parameters, the implementation semantics of a shrunk automaton is always a refinement of the exact semantics of the original automaton. Moreover, when $\mathcal{A}'$ is shrinkable (say, with parameters $\mathbf{k}\delta$), then $[\![\mathcal{A}'_{-\mathbf{k}\delta+\Delta}]\!]_\rho^{\mathsf{Impl}}$ is also non-blocking and $[\![\mathcal{A}']\!] \sqsubseteq_{\text{t.a.}} [\![\mathcal{A}'_{-\mathbf{k}\delta+\Delta}]\!]_\rho^{\mathsf{Impl}}$. Thus, the properties of shrinkable timed automata are preserved in implementation.

6.1 Related Work

A similar semantics, called the *program semantics*, was defined in [15] and was proven to be simulated by the enlarged semantics (as in the rightmost simulation in Proposition 12). Our definition follows their ideas, but the main difference is that our semantics is not *input-enabled*, that is, it can ignore signals during the treatment of another signal, and has no buffer. Both assumptions are applicable to different platforms (see [4, 16] for examples of systems that ignore any signal unless it is maintained long enough). Moreover, instead of detailing the reception and the treatment of the signals in several steps, we rather define action transitions taking a positive unpredictable amount of time, during which computations take place. This allows us to model the unpredictability using schedulers and state our properties *for any scheduler*. Note that two results in Proposition 12 are new compared to [15]: the leftmost simulation and the preservation of non-blockingness. Another recent work considers the behaviour of timed automata when actions have long execution times [1] but it assumes perfect clocks. A different line of work considers the implementability of timed automata extended with tasks but imprecisions and reaction times are not considered [19].

References

1 T. Abdellatif, J. Combaz, and J. Sifakis. Model-based implementation of real-time applications. In EMSOFT'10, p. 229–238, New York, NY, USA, 2010. ACM.

2 K. Altisen and S. Tripakis. Implementation of timed automata: An issue of semantics or modeling? In FORMATS'05, LNCS 3829, p. 273–288. Springer, 2005.

3 R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

4 E. Asarin, O. Maler, and A. Pnueli. On discretization of delays in timed automata and digital circuits. In CONCUR'98, LNCS 1466, p. 470–484. Springer, 1998.

5 F. Baccelli et al. *Synchronization and Linearity – An Algebra For Discrete Event Systems*. John Wiley & Sons, 1992.

6 J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, LNCS 2098, p. 87–124. Springer, 2004.

7 G. Berry. The foundations of Esterel. In *Proof, Language, and Interaction – Essays in Honour of Robin Milner*, p. 425–454. MIT Press, 2000.

8 P. Bouyer et al. Timed automata can always be made implementable. In CONCUR'11, LNCS 6901, p. 76–91, Aachen, Germany, 2011. Springer.

9 P. Bouyer, N. Markey, and P.-A. Reynier. Robust model-checking of linear-time properties in timed automata. In LATIN'06, LNCS 3887, p. 238–249. Springer, 2006.

10 P. Bouyer, N. Markey, and P.-A. Reynier. Robust analysis of timed automata via channel machines. In FoSSaCS'08, LNCS 4962, p. 157–171. Springer, 2008.

11 F. Cassez, T. A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In HSCC'02, LNCS 2289, p. 134–148. Springer, 2002.

12 P. Chamuczyński. *Algorithms and data structures for parametric analysis of real time systems*. PhD thesis, University of Göttingen, Germany, 2009.

13 A. David et al. Model checking timed automata with priorities using DBM subtraction. In FORMATS'06, LNCS 4202, p. 128–142. Springer, 2006.

14 M. De Wulf et al. Robust safety of timed automata. *Formal Methods in System Design*, 33(1-3):45–84, 2008.

15 M. De Wulf, L. Doyen, and J.-F. Raskin. Almost ASAP semantics: From timed models to timed implementations. *Formal Aspects of Computing*, 17(3):319–341, 2005.

16 H. Dierks. PLC-automata: a new class of implementable real-time automata. *Theoretical Computer Science*, 253:61–93, 2001.

17 D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In AVMFSS'89, LNCS 407, p. 197–212. Springer, 1990.

18 M. R. Henzinger, T. A. Henzinger, and P. W. Kopke. Computing simulations on finite and infinite graphs. In FOCS'95, p. 453–462, 1995.

19 T. A. Henzinger, B. Horowitz, and C. M. Kirsch. Giotto: A time-triggered language for embedded programming. In EMSOFT'01, LNCS 2211, p. 166–184. Springer, 2001.

20 T. Hune et al. Linear parametric model checking of timed automata. In TACAS'01, LNCS 2031, p. 189–203. Springer, 2001.

21 R. Jaubert and P.-A. Reynier. Quantitative robustness analysis of flat timed automata. In FOSSACS'11, LNCS 6604, p. 229–244. Springer, 2011.

22 A. Puri. Dynamical properties of timed automata. *Discrete Event Dynamic Systems*, 10(1-2):87–113, 2000.

23 O. Sankur. Untimed language preservation in timed systems. In MFCS'11, LNCS 6907, p. 556–567, Warsaw, Poland, 2011. Springer.

24 A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.