Shrinking Timed Automata [☆]

Ocan Sankur[a], Patricia Bouyer[b], Nicolas Markey[b]

[a] *Université Libre de Bruxelles, Brussels, Belgique*
[b] *LSV, CNRS & ENS Cachan, Cachan, France.*

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.

1. Introduction

Timed automata [1] 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, the formalism of timed automata is mainly intended to be used to describe abstract designs, so it is based on several idealistic assumptions. For instance, actions are assumed to be instantaneous, and the delays can be infinitely precise; in fact, one assumes perfectly continuous clocks and perfect measure of time. These properties are clearly not preserved in a hardware implementation of a system design.

The issue of the adequacy of the semantics of timed automata has been addressed in several works. It is known that timed automata, in general, are not *robust* to the relaxation of the idealistic assumptions mentioned above: guard errors or clock drifts, however small they may be, may yield extra qualitative behaviors (such as newly reachable locations) in some timed automata [2, 3]; whereas requiring positive reaction times can also disable some desired behaviors [4, 5]. We provide examples of such timed automata in Section 3. Therefore, given a target implementation platform, one needs to make sure that the verification results obtained using timed automata carry over to the implementation.

[☆]Partly supported by ANR project ImpRo (ANR-10-BLAN-0317), by ERC Starting grants EQualIS (FP7-308087) and inVEST (FP7-279499), and by European project Cassting (FP7-601148)

Email addresses: `ocan.sankur@normalesup.org` (Ocan Sankur), `patricia.bouyer@ens-cachan.fr` (Patricia Bouyer), `nicolas.markey@ens-cachan.fr` (Nicolas Markey)

Robust model-checking. In [2, 3], the semantics of timed automata was considered under guard enlargement, and algorithms were given to check the robustness of timed automata against this kind of perturbations. Guard enlargement models imprecisions in the semantics by syntactically relaxing all guards, thus 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 positive value for Δ under which a property is satisfied, has been proven decidable for safety properties [2, 3], and for richer linear-time properties [6, 7, 8, 9]. Properties verified in this framework are said to *robustly* hold, since they hold even if all the timing measures are subject to some bounded error. Such a robustness check allows to relax the idealistic assumptions of the original semantics of timed automata.

The guard enlargement formalism can also offer correct implementations in a concrete sense, as follows. In [10], a concrete semantics was given, the so-called *program semantics*, for a simple model of microprocessor executing a given timed automaton, taking into account small reaction times, and imprecise measure of time. It was proven that enlarging the guards of a timed automaton yields an overapproximation (in the sense of timed simulation) of the program semantics. Thus, studying the enlargement, say, by robust model-checking, is one way to ensure the correctness of an implementation modelled by the program semantics. Clearly, this method is valid for properties preserved by timed simulation; these include safety and linear-time properties mentioned above.

Contributions. In this work, we consider an alternative way to obtain robust timed automata models: To circumvent the effect of the clock imprecisions, given a timed automaton $\mathcal{A}$, we construct timed automaton $\mathcal{B}$ by *shrinking* the guards of $\mathcal{A}$, which is the opposite to enlargement, so that all behaviors of $\mathcal{B}$ under enlargement are included in those of $\mathcal{A}$. For instance, a given guard $x \in [a, b]$ is rewritten in $\mathcal{B}$ as $x \in [a + \delta, b - \delta]$, so that when $\mathcal{B}$ is subject to guard enlargement by Δ, this guard becomes

$$x \in [a + \delta - \Delta, b - \delta + \Delta] \subseteq [a, b],$$

provided that $\delta > \Delta$. Thus, the timed automaton $\mathcal{B}$ constructed in this manner is, even under guard enlargement, a *refinement*[1] of the abstract model $\mathcal{A}$ (under the assumption that $\Delta \leq \delta$). This also means that all timing requirements satisfied by $\mathcal{A}$, such as critical deadlines, or bounded response properties are strictly respected by $\mathcal{B}$. Hence, this method of obtaining implementations has the advantages of being simple and of introducing no behaviors.

Shrinking the guards may however remove interesting behaviors from the model and even introduce deadlocks. We will in fact see in Example 3.4 that this phenomenon may occur even in very simple real-time systems. In this paper, we are interested in the preservation of the desired behaviors in $\mathcal{B}$, in the following sense. A timed automaton $\mathcal{A}$ is said *non-blocking-shrinkable* if it can be shrunk into a timed automaton $\mathcal{B}$ that is non-blocking. It is *simulation-shrinkable* with respect to some automaton $\mathcal{F}$ if some shrinking $\mathcal{B}$ can time-abstract simulate $\mathcal{F}$. Thus, if both conditions hold, not only $\mathcal{B}$ is a refinement of $\mathcal{A}$, but it also still contains some behavior $\mathcal{F}$, and is non-blocking. Notice that if we choose $\mathcal{F}$ as the region graph of $\mathcal{A}$, then the shrinkability requires that $\mathcal{B}$ should 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 non-blocking-shrinkability can

[1]We employ this term to mean that there is a timed-simulation between $\mathcal{B}$ and $\mathcal{A}$.

be checked and computed in polynomial space, simulation-shrinkbility in pseudo-polynomial time. Thus, shrinkability w.r.t. the region graph can be checked in exponential time. *Strong shrinkability*, which requires both conditions to hold for a same shrinking can also be checked in polynomial space and in time polynomial in $\mathcal{F}$. Our algorithms are symbolic and manipulate a parameterized extension of difference bound matrices.

As a second result, we show that shrinkable timed automata are implementable in the following sense. We define an implementation semantics of timed automata corresponding to the execution by a digital system (with a digital clock). Our semantics is closely related to the program semantics of [10] 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 [10]. 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 is a refinement of 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 implementability in this model.

Finally, we note that shrinkability should also be seen as a robustness property by itself, since it asks whether the given automaton is vulnerable to the removal of limit behaviors by disallowing the borders of the guards. In fact, shrinkability can detect, for instance, whether the liveness of a timed automaton depends on the idealistic semantics to take transitions at infinitely precise time instants. Such an example of a non-robust system is given in Section 6.3. Thus, while [2, 3] and following work were interested in verifying the additional behaviors in timed automata due to enlargement or clock drifts, we consider here the dual problem of behavior preservation by checking the semantics under shrinking. Zeno behaviors and other convergence phenomena [4] are excluded naturally in shrunk timed automata (see Section 3.3).

Comparison with robust model-checking. We believe that the method of shrinking timed automata in order to construct robust timed automata complements that of robust model-checking. In fact, as mentioned above, shrinking has the advantage of yielding timed automata that are refinements of the original ones, thus the initial model can be used itself as a specification; whereas robust model-checking only guarantees that some chosen (and verified) properties will hold in the model under enlargement. On the other hand, shrinking requires all guards to be written as non-punctual intervals, so the initial timed automaton design should foresee an error interval for all measurements[2]. In contrast, robust model-checking can always be applied to timed automata with punctual guards. Such guards are often used to keep the design more abstract and simple, with the advantage of yielding smaller state spaces during classical verification. Note that all guards become non-punctual under enlargement, and robustness issues can arise even in timed automata with only punctual guards [3].

The choice between robust model-checking and shrinking thus depends on the design and specification at hand; for instance, whether it is sufficient to model-check against a set of properties under enlargement, or a stronger notion of refinement is preferred.

[2]Note that although it is possible to apply shrinkability on timed automata with punctual guards (see Section 6.3), the relation with the implementation semantics cannot be guaranteed (Section 7).

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 2.1. *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. $s_1 \sqsubseteq_{t.a.} s_2$).*

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 a time-abstract simulation. For any state s of $\mathcal{S}$, we write $\text{ta-sim}_{\mathcal{T}}(s)$ for the set of states of $\mathcal{T}$ that time-abstract simulate s in the disjoint union of $\mathcal{S}$ and $\mathcal{T}$. This set is called the *time-abstract simulator set* (or simply the *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. 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 consider *difference-bound matrices*, which are data structures used to represent sets of clock valuations in timed automata analysis [11]. Write $\mathcal{C} = \{1, \ldots, C\}$, and add an artificial clock of index 0, that has constant value 0. We let $\mathcal{C}_0 = \mathcal{C} \cup \{0\}$. We denote by $\mathcal{M}_n(X)$ the set of matrices of size $n \times n$ with coefficients in X. A *difference bound matrix* (DBM) over $\mathcal{C}_0$ is an element of $\mathcal{M}_{C+1}(\mathbb{Q}_\infty)$. 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 $[\![M]\!] = \{v \in \mathbb{R}^{\mathcal{C}}_{\geq 0} \mid \forall x, y \in \mathcal{C}_0, -M_{y,x} \leq v(x) - v(y) \leq M_{x,y}\}$. We will also denote the components of matrices M by $[M]_{x,y}$ to ease reading, for instance when M contains indices. Clearly, each DBM can be equivalently described by a guard, and conversely. Notice that we only consider closed zones, since we only have closed guards, but this is sufficient for our purpose. A DBM M is non-empty if $[\![M]\!]$ is non-empty. 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. The normalization of M is written $\mathsf{norm}(M)$. By extension, we say that a guard is normalized if it is the conjunction of all entries of a normalized DBM. Notice that a normalized guard contains $|\mathcal{C}_0|^2$ atomic clock constraints. We denote by $\Phi_{\mathcal{C}}$ the set of normalized guards on the clock set $\mathcal{C}$.

Definition 2.2. *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.*

Notice that following other works on robustness such as [2, 3], we only consider timed automata with *closed* guards, that is, we do not allow strict inequalities. In this paper, we assume that $\Sigma = E$ in all timed automata we consider, that is, each edge has a unique label. This may be restrictive for simulation relations. However, we are going to compare timed automata that have the same structure, so this assumption will allow us to ensure stronger relations. See Section 6.

Definition 2.3. *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 $\mathcal{L} \times \mathbb{R}_{\geq 0}^{\mathcal{C}} \times \mathbb{R}_{\geq 0}$. 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 $v' = (v+\tau)[R \leftarrow 0]$.*

Note that our definition of the timed automata semantics include the global time elapsed since the beginning; this is not the case in the original definition of [1]. Both definitions are equivalent, this additional component will ease the comparison with non-standard semantics in Section 7.

A timed automaton is non-blocking if the TTS it defines is.

An example of a timed automaton is given in Fig. 1.

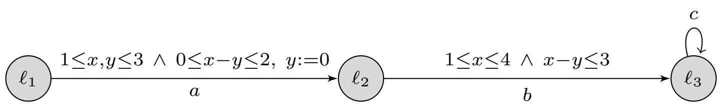

c
$1 \leq x, y \leq 3 \wedge 0 \leq x - y \leq 2,\ y := 0$
a
$1 \leq x \leq 4 \wedge x - y \leq 3$
b
ℓ_1
ℓ_2
ℓ_3

Figure 1: Timed automaton $\mathcal{A}$. Atomic clock constraints of the form $y < \infty$, $0 \leq y$ or $-\infty < x - y$ are omitted. One possible run of this timed automaton is the following: $(\ell_1, (0,0), 0) \xrightarrow{1.5} (\ell_1, (1.5, 1.5), 1.5) \xrightarrow{a} (\ell_2, (1.5, 0), 1.5)$ $\xrightarrow{2} (\ell_2, (3.5, 2), 3.5) \xrightarrow{b} (\ell_3, (3.5, 2), 3.5) \xrightarrow{c} (\ell_3, (3.5, 2), 3.5) \ldots$.

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

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

The enlargement of a guard g, denoted by $\langle g \rangle_\delta$, is obtained by enlarging all its atomic clock constraints. Note that δ can be negative here; this operation is then called *shrinking*. We will write $v \models_\delta g$ to mean $v \models \langle g \rangle_\delta$.

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 $\delta = (\delta_i)_{i \in I}$, we define $\mathcal{A}_\delta$ 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. If no timed automaton is implicit in the context, for any guard $g \in \Phi_{\mathcal{C}}$, we also define a shrinking as $\langle g \rangle_{-\mathbf{k}\delta}$ where $\delta > 0$ and $\mathbf{k} \in \mathbb{N}^{|\mathcal{C}_0|^2}$.

3. Robustness and Shrinkability

3.1. Previous Work on Robustness

The effect of small imprecisions on the semantics of timed automata was investigated by Puri [2], who gave an example of a timed automaton that is *not robust*: its behaviour changes in the presence

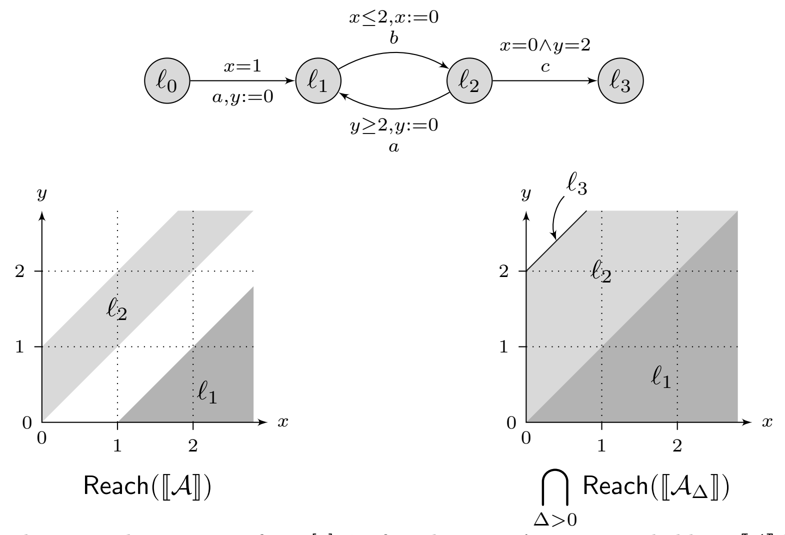


Figure 2: A non-robust timed automaton from [2]. In fact, location ℓ_3 is not reachable in $[\![\mathcal{A}]\!]$ but it is reachable in $[\![\mathcal{A}_\Delta]\!]$ for any $\Delta > 0$ [3].

of the slightest clock drift. Clock drifts were further investigated in [12, 13]. De Wulf *et al.* later established strong similarities between clock drifts and guard enlargement [3] with respect to location reachability, showing that Puri's example (shown in Fig. 2) is also not robust with respect to guard enlargement. Indeed, consider the difference $y - x$ in ℓ_2 (or, equivalently, the value of y when entering ℓ_2): in $\mathcal{A}$, this value can only decrease, since between two consecutive visits to ℓ_2, clock y goes down by at least 2 (it is reset to zero when it is larger than or equal to 2), while clock x goes down by at most 2. In $[\![\mathcal{A}_\Delta]\!]$, the difference $y - x$ can increase by as much as 2Δ between two consecutive visits to ℓ_2 (as long as it is less than or equal to $2 - \Delta$), which eventually makes ℓ_3 reachable, however small Δ may be.

Motivated by the non-robustness of the semantics of timed automata, previous works considered validating the behaviour of timed automata under small guard enlargement. Robust model-checking, that is, determining a bound on the enlargement under which the system satisfies a given property was investigated first for safety, then for richer linear-time properties in [2, 3, 6, 7, 9, 8]. Symbolic algorithms were suggested for subclasses of timed automata in [14, 15]. The idea of syntactic enlargement was re-visited in [16] in a game semantics, where perturbations are controlled by a separate player, so that the controller can react to observed perturbations in order to fulfill a given objective. See also [17, 18] for a variant of the game semantics.

It was shown in [10] how this framework allows one to prove that the semantics is preserved in a model of physical implementation with digital clocks. In fact, the semantics under enlargement was shown to be an overapproximation (in the sense of timed simulation) of a semantics corresponding to the execution of the system by a simple model of a microprocessor.

In a recent work [19], 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 transformation can be applied to any timed automaton, and an ϵ-bisimilar enlarged model can be obtained for any desired $\epsilon > 0$. However, the resulting automaton is not, in general, a refinement of the abstract model, since the behaviour is only preserved approximately. Moreover, $\mathcal{A}'$ is constructed by embedding the region automaton so its size is exponential.

Other approaches to robustness have also been considered. In [20], the authors show how various kinds of clock imprecisions can be incorporated in the timed automaton model, without redefining the semantics. Such an approach was used in several case studies *e.g.* [21, 22]. The limitation is that this can significantly increase the size of the model, and that one has to fix the imprecision parameter. Other notions of robustness include the "tube semantics" of [23]. This consists in discarding some isolated behaviour, rather than introducing new ones, and it is not directly related to our work. See also [24] for a survey of different approaches on defining robustness in timed automata.

3.2. Shrinkability

We define *shrinkings* of timed automata and show how they can provide an alternative way to construct robust systems. Our method always yields a refinement of the abstract model, and moreover, we only modify the guards of a given automaton, so the size is unchanged. Our algorithms then allow to decide whether further properties, such as non-blockingness and time-abstract simulation of some automaton, are satisfied.

In order to circumvent the effect of the imprecisions, we suggest *shrinking* any guard of the form "$x \in [a, b]$" into "$x \in [a + \delta, b - \delta]$" for some $\delta > 0$. The idea is 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 new guard implies the satisfaction of the original guard, even with imprecisions.

Formally, a *shrinking* of a timed automaton $\mathcal{A}$ is $\mathcal{A}_{-\mathbf{k}\delta}$, where $\mathbf{k} \in \mathbb{N}^{I}_{>0}$ and $\delta > 0$, I denoting the set of all atomic clock constraints of $\mathcal{A}$. Following the remark on guards above, the enlarged automaton $\mathcal{A}_{-\mathbf{k}\delta+\Delta}$ is a refinement of $\mathcal{A}$: $[\![\mathcal{A}_{-\mathbf{k}\delta+\Delta}]\!] \sqsubseteq [\![\mathcal{A}]\!]$, provided that $\Delta < \min_{i \in I}(k_i) \cdot \delta$. The application of shrinking to implementability following this idea is developed in detail in Section 7.

Shrinking is a natural idea when one is interested in the strict preservation of timing constraints, such as critical deadlines or bounded response properties. However, shrinking may remove too many behaviours and the resulting automaton may even become blocking (See *e.g.* Fig. 5). We are interested in deciding the existence of shrinking parameters $\mathbf{k}$ and $\delta > 0$, and in their computation, for which the shrunk timed automaton is non-blocking, or able to time-abstract simulate a given automaton, or both.

The formal definition of *non-blocking-shrinkability* is the following.

Definition 3.1. *Let $\mathcal{A}$ be a timed automaton, and I the set of its atomic clock constraints. $\mathcal{A}$ is* non-blocking-shrinkable *if there exists $\mathbf{k} \in \mathbb{N}^{I}_{>0}$ and $\delta_0 \in \mathbb{Q}_{>0}$ such that for all $\delta \in [0, \delta_0]$, $[\![\mathcal{A}_{-\mathbf{k}\delta}]\!]$ is non-blocking.*

We now define *simulation-shrinkability.* Ideally, we would like the shrunk timed automaton to be able to time-abstract simulate the original timed automaton. But we give a more general definition, which allows us to define shrinkings that contain *some* part of the time-abstract behaviour of the original automaton. Given a timed automaton $\mathcal{A}$, and some finite automaton $\mathcal{F}$ such that $[\![\mathcal{F}]\!] \sqsubseteq_{\text{t.a.}} [\![\mathcal{A}]\!]$, $\mathcal{A}$ is said to be *simulation-shrinkable with respect to* $\mathcal{F}$, if some shrinking $\mathcal{A}_{-\mathbf{k}\delta}$ simulates $\mathcal{F}$[3]. Notice that $\mathcal{F}$ can be chosen as the region automaton of $\mathcal{A}$ [1] or a smaller bisimulation quotient [25], in which case a shrinking is required to time-abstract simulate $\mathcal{A}$ entirely. Otherwise, we still guarantee that the shrinking contains some relevant behaviour $\mathcal{F}$ of the abstract model. The formal definition is the following.

[3]To define $[\![\mathcal{F}]\!]$ as a TTS, we see $\mathcal{F}$ as a timed automaton with no clocks.

Definition 3.2. *Let $\mathcal{A}$ be a timed automaton, and I the set of its atomic clock constraints. Consider any finite automaton $\mathcal{F}$ such that $[\![\mathcal{F}]\!] \sqsubseteq_{t.a.} [\![\mathcal{A}]\!]$. $\mathcal{A}$ is* simulation-shrinkable w.r.t. $\mathcal{F}$ *if there exists $\mathbf{k} \in \mathbb{N}^{I}_{>0}$ and $\delta_0 \in \mathbb{Q}_{>0}$ such that for all $\delta \in [0, \delta_0]$,*

$$[\![\mathcal{F}]\!] \sqsubseteq_{t.a.} [\![\mathcal{A}_{-\mathbf{k}\delta}]\!]$$

with the following additional requirement: for each state f of $\mathcal{F}$, there exists a guard g and $\mathbf{h} \in \mathbb{N}^{|\mathcal{C}_0|^2}$ such that for all $\delta \in [0, \delta_0]$, ta-sim$_{[\![\mathcal{A}_{-\mathbf{k}\delta}]\!]}(f)$ equals $[\![\langle g \rangle_{-\mathbf{h}\delta}]\!]$.

Furthermore, we say that a timed automaton is *strongly shrinkable w.r.t.* $\mathcal{F}$ if it has a shrinking witnessing both its non-blocking-shrinkability and its simulation-shrinkability w.r.t. $\mathcal{F}$. The above $\mathbf{k}$ and δ_0 are called the *shrinking parameters* of $\mathcal{A}$.

We now comment on the above definitions. We define shrinkability "for all $\delta \in [0, \delta_0]$", 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 underapproximation 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" [20, 10]. 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*). If this is the case, then when we change slightly the parameter $\delta > 0$, the simulator sets also change slightly. Moreover, simulator sets have a uniform expression, where δ is only a parameter. When one uses the simulator sets to control the original system, this property yields a uniform representation for the constraints to add to the system. Thus, the controlled system can still be represented as a timed automaton, where the guards contain the parameter δ. Note however that we do not know whether there exist timed automata that would be shrinkable but violate the technical requirement of Definition 3.2.

Example 3.3. *We illustrate shrinkability on the timed automaton $\mathcal{A}$ of Fig. 1. This timed automaton is shrinkable both for non-blockingness and for simulation. One can check that $\mathcal{A}_{-\mathbf{k}\delta}$, shown in Fig. 3 is non-blocking, and can time-abstract simulate $\mathcal{A}$ for all $\delta \in [0, \frac{1}{6}]$. One can also see that the simulator sets are uniform. For instance, the set of states of $\mathcal{A}_{-\mathbf{k}\delta}$ that simulate the initial state of $\mathcal{A}$ is $[\![\delta \le x \le 3 - \delta \wedge y \le 3 - 2\delta \wedge \delta \le x - y \le 2 - 2\delta]\!]$ for all $\delta \in [0, \frac{1}{6}]$. The computation of these parameters, and that of the simulator sets of this example are explained in Example 4.10.*

$$\begin{tikzcd}[column sep=huge]
\ell_1 \arrow[r, "{1+2\delta\le x\le 3-\delta \ \wedge\ 1+\delta\le y\le 3-2\delta \quad \delta\le x-y\le 2-2\delta,\ \ y:=0}", "a"'] & \ell_2 \arrow[r, "{1+\delta\le x\le 4-\delta \ \wedge\ \delta\le y \ \wedge\ x-y\le 3-\delta}", "b"'] & \ell_3 \arrow[loop above, "c"]
\end{tikzcd}$$

Figure 3: A shrinking $\mathcal{A}_{-\mathbf{k}\delta}$ of timed automaton $\mathcal{A}$ defined in Fig. 1.

Example 3.4. *We give a simple real-time scheduling example to illustrate shrinkability. We consider a simple producer-consumer system where an event is produced at least every τ time units, while the consumer processes an event at most every τ time units. For the sake of the example, we will assume that productions and consumptions should alternate; this corresponds to assuming no buffer between both subsystems. Let us assume further that at each iteration, the producer can be made run more or less faster as long as one production is made at least every τ time units and both events alternate. Similarly, the frequency of the consumer can be adjusted but only to go slower.*

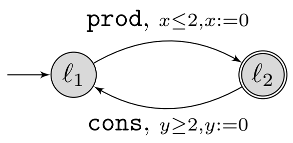


Figure 4: A timed automaton that models an alternating producer-consumer system. The only timing requirements are that one production should happen at least once every 2 time units, and one consumption at most once every 2 time units. The speed can be adjusted as long as these constraints are satisfied and both events alternate.

The resulting system can be summarized by the timed automaton $\mathcal{A}$ of Fig. 4, for $\tau = 2$. The reader will notice that this automaton is precisely the cycle in the timed automaton we already saw in Fig. 2.

As we discussed earlier, this system is not robust to guard enlargement. That is, if productions occur once every $2 + \delta$ time units, or consumptions every $2 - \delta$ time units, then an error state can be reached, where the time delay between corresponding productions and consumptions become larger than or equal to 2 (the error state is omitted in Fig. 4 but could be added as in Fig. 2). Thus, not all runs are satisfying our purpose. In this case, applying shrinking might be a good idea since we know that it will yield, even under enlargement, a refinement of the exact model. In other terms, shrinking will restrict the possible behaviors of the system to desired ones. However, we would like to check for shrinkability, to make sure that the system is able to run forever under shrinking.

To apply simulation-shrinkability, we consider the finite automaton $\mathcal{F}$ which consists of one simple cycle of label `cons` $\cdot$ `prod`. One can check that the timed automaton $\mathcal{A}$ can time-abstract simulate $\mathcal{F}$; in fact, $\mathcal{A}$ has infinite runs. Now, the simulation-shrinkability of $\mathcal{A}$ w.r.t. $\mathcal{F}$ consists in asking whether this timed system has infinite behaviors (simulating $\mathcal{F}$) when we assume that the periods of consumption and production cannot be guaranteed to be adjusted to τ exactly. In fact, shrinking the guards yields $x \leq 2 - k_1\delta$ for the production, and $y \geq 2 + k_2\delta$ for the consumption, where $\mathbf{k}\delta$ are unknown positive parameters.

One can prove however that $\mathcal{A}$ is not simulation-shrinkable w.r.t. $\mathcal{F}$. In fact, if productions occur faster than once every $2 - \delta$ time units, and consumptions at most every $2 + \delta$ time units, the delay between consecutive production-consumption event pairs will keep growing, and the system will eventually enter a deadlock at location ℓ_1 because of the upper bound $x \leq 2 - k_1\delta$. Thus, the infinite behavior one can find by analyzing the exact semantics of this timed automaton actually requires highly precise timings. Thus, the model is not only not robust to guard enlargement, but its infinite behaviors are not realizable neither under finite precision.

The algorithms presented in this paper allows to detect this non-shrinkability; we show in Section 6.3 how to correct this system in order to construct a robust and shrinkable one.

3.3. 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, say, simulation-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 $\delta_i > 0$, so time diverges in any infinite run.

But unrealistic timing constraints are not limited to Zeno behaviours. The automaton in Fig. 5 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.

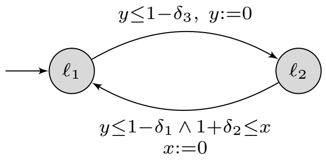

$y \leq 1-\delta_3,\ y:=0$
ℓ_1
ℓ_2
$y \leq 1-\delta_1 \wedge 1+\delta_2 \leq x$
$x:=0$

Figure 5: A shrunk timed automaton that is blocking whenever $\delta_2 > 0$ or $\delta_3 > 0$. To see this, consider any infinite execution and let $d_1, d_3, \ldots$ denote the delays at location l_1, and $d_2, d_4, \ldots$ those at l_2. One can show that $1 \leq d_{2i-1} + d_{2i}$ for all $i \geq 1$, which means that time diverges, but also $\delta_2 + \delta_3 \leq d_{2i+2} - d_{2i}$ and $d_{2i} \leq 1$. The latter means that the sequence $(d_{2i})_i$ increases at least by $\delta_2 + \delta_3$ at each step and is bounded above by 1, which is possible only when $\delta_2 = \delta_3 = 0$. Note that even if $\delta_2 = \delta_3 = 0$, non-blockingness requires consecutive delays to be equal, which is not realistic for digital systems.

A similar example appears in [4] with equality constraints, so it is trivially not shrinkable. The above example shows that such phenomena can occur even without the use of equality.

3.4. Decidability of Shrinkability

Our main result is the decidability of the shrinkability problems.

Theorem 3.5. *The following results hold.*

- *For closed non-blocking timed automata, non-blocking-shrinkability is in* PSPACE*, and in* NP *if the number of outgoing transitions from each location is bounded.*
- *For closed timed automata $\mathcal{A}$ with distinct labels, simulation-shrinkability w.r.t. any $\mathcal{F}$ is decidable in pseudo-polynomial time in the sizes of $\mathcal{A}$ and $\mathcal{F}$.*
- *For closed non-blocking timed automata with distinct labels, strong shrinkability w.r.t. $\mathcal{F}$ is decidable in time exponential in $\mathcal{A}$, and polynomial in $\mathcal{F}$, more precisely, in time $O(2^{|\mathcal{A}|} + p(|\mathcal{F}|, |\mathcal{A}|, M))$, where M is the largest constant of $\mathcal{A}$, and p some polynomial.*

Moreover, we will show that when a given timed automaton is shrinkable, the least shrinking parameters can be computed (in a sense defined in Section 6). We assume distinct labels for simulation-shrinkability, mainly for technical reasons. Nevertheless, it is also meaningful for our purpose. In fact, we compare $\mathcal{A}$ with its shrinking and require these to have approximately the same behaviour. Checking simulation between these systems under this assumption not only requires "observational" equivalence but also a structural one. In fact, if a shrinking simulates $\mathcal{F}$, it should do so by following exactly the same edges as $\mathcal{A}$ does when simulating $\mathcal{F}$. Thus, we require the shrinking and the initial automaton to have the same internal behaviour.

In the rest of the paper, we present the proof of this result. We begin by defining an extension of *difference-bound matrices* (DBMs), called *shrunk DBMs*, which are used to represent the state space of shrunk timed automata, and give tools for solving fixpoint equations on shrunk DBMs by reduction to fixpoint equations in the max-plus algebra. We then explain how these results can be used to solve the shrinkability problems, by expressing them as fixpoint equations. In Section 7, 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

We recall several *elementary operations* on DBMs. Given a DBM M, we let $\mathrm{Pre}_{\mathrm{t}}(M)$ be the normalized DBM that describes the time predecessors of $[\![M]\!]$, *i.e.*, $[\![\mathrm{Pre}_{\mathrm{t}}(M)]\!] = \{v \in \mathbb{R}^{\mathcal{C}}_{\geq 0} \mid \exists t \in \mathbb{R}_{\geq 0} \text{ s.t. } v + t \in [\![M]\!]\}$. 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 [26, 27].

Shrunk DBMs. 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 *shrinking matrix* is an element of $\mathcal{M}_{C+1}(\mathbb{N})$, and a *parameterized shrinking matrix (PSM)* an element of $\mathcal{M}_{C+1}(\mathcal{G}(\mathbf{k}))$. If P is a PSM and ν is a valuation, then $P[\nu]$ is the shrinking matrix defined by replacing each parameter k_i by $\nu(k_i)$, and evaluating the resulting expression.

We introduce a new data structure called *shrunk DBM*, by combining DBMs and PSMs. A shrunk DBM is written in the form $M - \delta P$, where M is a DBM, P a PSM and δ is a fresh parameter. Let us explain how shrunk guards can be expressed using shrunk DBMs. Let M be a DBM, P a shrinking matrix, and $\delta > 0$. Suppose for example that for some i, $M[i, 0] = \alpha$, $M[0, i] = \beta$, $P[i, 0] = k$ and $P[0, i] = l$. Then, M defines the constraint "$-\beta \leq x \leq \alpha$" for some clock x, whereas $M - \delta P$ defines "$-\beta + l\delta \leq x \leq \alpha - k\delta$". Furthermore, if M represents a guard g, the shrunk guard $\langle g \rangle_{-\delta}$ can be represented by the shrunk DBM $M - \mathbf{1}\delta$, where matrix $\mathbf{1}$ has 0's on the diagonal and 1's everywhere else. A property that is crucial for our results is that shrunk DBMs are closed under several elementary operations. These operations are defined below.

When manipulating shrunk DBMs, we will often be interested in the properties of shrunk DBMs $M - \delta P$ for "small enough $\delta > 0$", which means for all $\delta \in [0, \delta_0]$, for some $\delta_0 > 0$. We will use the notation (M, P), which means that we consider $M - \delta P$ for small enough $\delta > 0$. For instance, when we write $\mathrm{Pre}_{\mathrm{t}}((M, P)) = (N, Q)$, we mean that there exists $\delta_0 > 0$ such that $\mathrm{Pre}_{\mathrm{t}}(M - \delta P) = N - \delta Q$ holds for all $\delta \in [0, \delta_0]$. In all the operations, an upper bound on δ_0 will be computable.

Example 4.1. *We consider the timed automaton of Fig. 1, with $g_1 = 1 \leq x, y \leq 3 \wedge 0 \leq x - y \leq 3$ the guard of the edge from ℓ_1 to ℓ_2. Let G_1 denote the DBM that represents this guard. We consider the following shrinking of this guard by parameters $(k_i)_i\delta$:*

$$g_1' = 1 + k_1\delta \leq x \leq 3 - k_2\delta \wedge 1 + k_3\delta \leq y \leq 3 - k_4\delta \wedge k_5\delta \leq x - y \leq 2 - k_6\delta.$$

This shrunk guard can be expressed by the shrunk DBM (G_1, P), where G_1 and P are defined as follows.

$$G_1 = \begin{array}{c} \\ 0 \\ x \\ y \end{array} \begin{array}{c} \begin{array}{ccc} 0 & x & y \end{array} \\ \begin{pmatrix} 0 & -1 & -1 \\ 3 & 0 & 3 \\ 3 & 0 & 0 \end{pmatrix} \end{array} \qquad P = \begin{array}{c} \\ 0 \\ x \\ y \end{array} \begin{array}{c} \begin{array}{ccc} 0 & x & y \end{array} \\ \begin{pmatrix} 0 & k_1 & k_3 \\ k_2 & 0 & k_6 \\ k_4 & k_5 & 0 \end{pmatrix} \end{array}$$

Note that our definition of PSM is different from parametric DBMs considered for instance in [28], since we use max-plus polynomials instead of linear expressions and only consider natural number valuations.

Operations on shrunk DBMs. We are going to use shrunk DBMs to express the state space of shrunk timed automata. In order to do this, we need to study some algebraic properties of shrunk DBMs. Let us first define symbolic operations on expressions of the form $\alpha - k \cdot \delta$, where $\alpha \in \mathbb{Q}$, k is an expression in $\mathcal{G}(\mathbf{k})$ (bound to take values in $\mathbb{N}$). In such expressions, δ is a variable (which at the end will take *small* non-negative real values, see Lemma 4.2). More precisely, given two pairs (α, k) and (β, l) in $\mathbb{Q} \times \mathcal{G}(\mathbf{k})$, we define the addition $(\alpha, k) + (\beta, l) = (\alpha + \beta, k + l)$, where $(k + l)[\nu] = k[\nu] + l[\nu]$ for all $\nu\colon \mathbf{k} \to \mathbb{N}$. We also define the following relation:

$$(\alpha, k) \preceq (\beta, l) \quad \Leftrightarrow \quad \alpha < \beta \text{ or } (\alpha = \beta \text{ and } k[\nu] \geq l[\nu] \text{ for all } \nu\colon \mathbf{k} \to \mathbb{N}).$$

An important (but quite straightforward) property of this definition is the following. Notice that we are again interested in comparing (α, k) and (β, l) for small positive values of δ.

Lemma 4.2. *For all (α, k) and (β, l) in $\mathbb{Q} \times \mathcal{G}(\mathbf{k})$, the following two statements are equivalent:*

- $(\alpha, k) \prec (\beta, l)$
- *for all $\nu\colon \mathbf{k} \to \mathbb{N}$, there exists $\delta_0 > 0$ s.t. for all $0 \leq \delta < \delta_0$, it holds $\alpha - k[\nu] \cdot \delta \leq \beta - l[\nu] \cdot \delta$.*

Proof. Assume that the second statement holds. In particular when $\delta = 0$, we get $\alpha \leq \beta$. Hence either $\alpha < \beta$, and we are done, or $\alpha = \beta$. In the latter case, we get that for all ν, it holds $k[\nu] \cdot \delta \geq l[\nu] \cdot \delta$ for small positive values of δ. Hence for all ν, $k[\nu] \geq l[\nu]$.

Conversely, we consider several cases:

- if $\alpha < \beta$: then for all $\nu\colon \mathbf{k} \to \mathbb{N}$,
  - either $k[\nu] = l[\nu]$ and any non-negative δ satisfies $\alpha - k[\nu] \cdot \delta < \beta - l[\nu] \cdot \delta$ (so that δ_0 can be any positive real);
  - or $|k[\nu] - l[\nu]| \geq 1$, in which case we let $\delta_0 = \frac{\beta - \alpha}{|k[\nu] - l[\nu]|}$. Then any nonnegative $\delta < \delta_0$ satisfies $|k[\nu] - l[\nu]| \cdot \delta < \beta - \alpha$. In particular, $(l[\nu] - k[\nu]) \cdot \delta < \beta - \alpha$, as required.
- if $\alpha = \beta$ and $k[\nu] \geq l[\nu]$ for all ν: then for any non-negative δ, $k[\nu] \cdot \delta \geq l[\nu] \cdot \delta$, and the result follows.

□

Notice that $\preceq$ is not an ordering relation: indeed, $(\alpha, k) \preceq (\beta, l) \preceq (\alpha, k)$ implies $\alpha = \beta$ and $k[\nu] = l[\nu]$ for all ν, but the latter does not imply equality of k and l (syntactically). In the sequel, we (silently) consider the quotient of $\mathcal{G}(\mathbf{k})$ by this equivalence relation, so that $\preceq$ is a partial order. To see that not all elements are comparable, consider $(1, k_1)$ and $(1, k_2 + k_3)$. For some valuations, the former is smaller than the latter, and for some valuations the inverse is true.

We now define the *minimum* of two elements of $\mathbb{Q} \times \mathcal{G}(\mathbf{k})$:

$$\min((\alpha, k), (\beta, l)) = \begin{cases} (\alpha, k) & \text{if } \alpha < \beta, \\ (\beta, l) & \text{if } \beta < \alpha, \\ (\alpha, \max(k, l)) & \text{otherwise.} \end{cases}$$

This definition enjoys the following property:

Lemma 4.3. *For all (α, k) and (β, l) in $\mathbb{Q} \times \mathcal{G}(\mathbf{k})$, $\min((\alpha, k), (\beta, l))$ belongs to $\mathbb{Q} \times \mathcal{G}(\mathbf{k})$ and is the greatest lower bound of (α, k) and (β, l) in that set.*

Proof. The first statement is obvious, since $\max(k, l)$ belongs to $\mathcal{G}(\mathbf{k})$ as soon as k and l do. That $\min((\alpha, k), (\beta, l)) \preceq (\alpha, k)$ is easily obtained by considering three cases:

- if $\alpha < \beta$, then $\min((\alpha, k), (\beta, l)) = (\alpha, k)$, which implies our result;
- if $\beta < \alpha$, then $\min((\alpha, k), (\beta, l)) = (\beta, l)$, and $(\beta, l) \preceq (\alpha, k)$ since $\beta < \alpha$;
- if $\alpha = \beta$, then $\min((\alpha, k), (\beta, l)) = (\alpha, \max(k, l))$, and $(\alpha, \max(k, l)) \preceq (\alpha, k)$ since $(\max(k, l))[\nu] \geq k[\nu]$ for all ν (by definition of max).

Symmetrically, $\min((\alpha, k), (\beta, l)) \preceq (\beta, l)$.

Now, consider any (γ, m) s.t. $(\gamma, m) \preceq (\alpha, k)$ and $(\gamma, m) \preceq (\beta, l)$. We show that then $(\gamma, m) \preceq \min((\alpha, k), (\beta, l))$. This follows immediately from the observation that $\min((\alpha, k), (\beta, l))$ is either equal to (α, k), or to (β, l), since the assumption on (γ, m) then implies the desired inequality. We have, in fact, if $\alpha < \beta$, then $\min((\alpha, k), (\beta, l)) = (\alpha, k)$, and symmetrically for $\alpha > \beta$. If $\alpha = \beta$ and, say, $k > l$, then the minimum is (α, k). □

The following set of propositions explains how elementary operations can be computed on shrunk DBMs. In particular, it shows how shrinking parameters (*i.e.*, the PSMs) can be propagated in a backward analysis while staying in the max-plus theory. All operations are illustrated in Example 4.10 at the end of this section.

We first show that shrunk DBMs also have a *normal form* in the following sense. Just like for DBMs, the normalization of a shrunk DBM (M, P) is obtained simply by computing the shortest path between all indices i and j, by interpreting (M, P) as the adjacency matrix of a directed graph. The algorithm for normalization is the Floyd-Warshall all-pairs shortest path algorithm applied to regular DBMs, but we simply apply it in our algebra over $\mathbb{Q} \times \mathcal{G}(\mathbf{k})$ where the sum, the order $\preceq$ and min are defined as above.

Algorithm 1 Normalization procedure for shrunk DBMs.

```
Given a shrunk DBM $(M, P)$,
for $i = 0..n$ do
  for $j = 0..n$ do
    for $k = 0..n$ do
      $(M_{i,j}, P_{i,j}) \leftarrow \min\big((M_{i,j}, P_{i,j}), (M_{i,k}, P_{i,k}) + (M_{k,j}, P_{k,j})\big)$.
    end for
  end for
end for
```

Lemma 4.4. *Let M be any DBM and P be a PSM. Then, there exists a PSM P' such that for all valuations $\nu\colon \mathbf{k} \to \mathbb{N}$, there exists $\delta_0 > 0$ for which $\mathsf{norm}(M - \delta \cdot P[\nu]) = M' - \delta \cdot P'[\nu]$ for all $\delta \in [0, \delta_0]$, where $M' = \mathsf{norm}(M)$. Moreover, one can compute δ_0.*

Proof. We initialize the matrix (N, Q) as the matrix of $\mathcal{M}_{C+1}(\mathbb{Q} \times \mathcal{G}(\mathbf{k}))$ whose element in cell (i, j) is $(M_{i,j}, P_{i,j})$. We first prove that each step of the normalization algorithm applied to $R = (N, Q)$ in $\mathcal{M}_{C+1}(\mathbb{Q} \times \mathcal{G}(\mathbf{k}))$ yields a matrix in $\mathcal{M}_{C+1}(\mathbb{Q} \times \mathcal{G}(\mathbf{k}))$. More precisely, given $R = (N, Q)$ and integers i, j and k less than or equal to C, we prove that the matrix R' obtained from R by replacing $R_{i,j}$ with $\min(R_{i,j}, R_{i,k} + R_{k,j})$ can be written as (N', Q') for some DBM N' and $Q' \in \mathcal{M}_{C+1}(\mathcal{G}(\mathbf{k}))$.

Assume that $N_{i,j} \leq N_{i,k} + N_{k,j}$. If the inequality is strict, then $\min(R_{i,j}, R_{i,k} + R_{k,j})$ is $(N_{i,j}, Q_{i,j})$. Otherwise $N_{i,j} = N_{i,k} + N_{k,j}$: then $R'_{i,j}$ is set to $(N_{i,j}, \max(Q_{i,j}, Q_{i,k} + Q_{k,j}))$, which satisfies our requirement. Assume now that $N_{i,k} + N_{k,j} < N_{i,j}$. In this case, $R'_{i,j} = R_{i,k} + R_{k,j}$. The upper bound on δ, under which the inequalities hold can be computed using Lemma 4.2.

Let us write (M', P') the shrunk DBM returned by the normalization algorithm. By the definition of the minimum on pairs $(N_{i,j}, Q_{i,j})$, the algorithm applies normalization on the DBM N, so $M' = \mathsf{norm}(M)$. Pick any valuation $\nu \colon \mathbf{k} \to \mathbb{N}$. We define a negative cycle of a shrunk DBM $(M, P[\nu])$ as $i_1, i_2, \ldots, i_k \in \mathcal{C}_0$, where $i_1 = i_k$, and $(M_{i_1,i_2}, P_{i_1,i_2}) + (M_{i_2,i_3}, P_{i_2,i_3}[\nu]) + \ldots + (M_{i_{k-1},i_k}, P_{i_{k-1},i_k}[\nu]) \prec (0,0)$. if $(M, P[\nu])$ contains a negative cycle, then for any $\delta > 0$, $M - \delta P[\nu]$ is empty since this would imply the constraint $x - x \leq M'_{i_1,i_1} - \delta P'_{i_1,i_1}[\nu] < 0$, for some clock x. Otherwise, we have $\mathsf{norm}(M - \delta \cdot P[\nu]) = M' - \delta \cdot P'[\nu]$ for all small enough $\delta > 0$. In fact the operations performed on the parameterized expressions during the normalization algorithm correspond to the normalization algorithm applied on the DBM $M - \delta P$, by Lemma 4.3. Last, by applying Lemmas 4.2 and 4.3 to the (finitely many) cells of the matrix (M', P') we get $\delta_0 > 0$ such that such that for all $\delta \in [0, \delta_0]$ and for all indices i, j and k,

$$M_{i,j} - P'_{i,j}[\nu] \cdot \delta \leq M_{i,k} - P'_{i,k}[\nu] \cdot \delta + M_{k,j} - P'_{k,j}[\nu] \cdot \delta,$$

so that $M - P'[\nu] \cdot \delta$ is in normal form. Also, quite obviously, for any ν and δ as above, the DBM obtained at each step of the normalization algorithm defines the same set of valuations as the original DBM. □

Note that even if a DBM M is not empty, a shrunk DBM (M, P) can be empty. For instance, the shrunk DBM corresponding to constraints $1 + \delta \leq x \leq 1 - \delta$ is empty for all $\delta > 0$, although the set $[\![1 \leq x \leq 1]\!]$ is not. The emptiness of a shrunk DBM (M, P), *i.e.* whether $M - \delta P$ is empty for all $\delta > 0$, can be determined by first applying normalization, then checking whether all diagonal components are 0.

The intersection of two shrunk DBMs can also be written as a shrunk DBM:

Lemma 4.5. *Let M, N^1, N^2 be normalized DBMs such that $M = N^1 \cap N^2$. Then, for all PSMs P^1 and P^2, there exists a PSM P' such that for all valuations $\nu \colon \mathbf{k} \to \mathbb{N}$, there exists $\delta_0 > 0$ for which $M - \delta \cdot P'[\nu] = (N^1 - \delta \cdot P^1[\nu]) \cap (N^2 - \delta \cdot P^2[\nu])$ for all $\delta \in [0, \delta_0]$.*

Proof. For all i, j, define $(N_{i,j}, P_{i,j}) = \min((N^1_{i,j}, P^1_{i,j}), (N^2_{i,j}, P^2_{i,j}))$. Then by definition, $N_{i,j} = \min(N^1_{i,j}, N^2_{i,j})$, so that $[\![N]\!] = [\![M]\!]$. Applying Lemma 4.2 (several times), for any valuation ν, there is a positive δ_0 for which $N_{i,j} - P_{i,j}[\nu] \cdot \delta = \min({N^1}_{i,j} - {P^1}_{i,j}[\nu] \cdot \delta, {N^2}_{i,j} - {P^2}_{i,j}[\nu] \cdot \delta)$, for all $0 \leq \delta < \delta_0$, so that the DBM $N - P[\nu] \cdot \delta$ corresponds to the intersection of $N^1 - P^2[\nu] \cdot \delta$ and $N^2 - P^2[\nu] \cdot \delta$.

To conclude, it suffices to apply Lemma. 4.4 to turn the resulting shrunk DBM in normal form. □

We now describe how we compute the shrinking matrix for the *unreset* operation: given a DBM M and a set of clocks Z, $\text{Unreset}_Z(M)$ is the (normal-form) DBM defining all clock valuations v s.t. $v[Z := 0]$ belongs to M. It is obtained by intersecting with the DBM representing the set $Z = 0$, and removing contraints involving clocks in Z.

Lemma 4.6. *Let M and N be two normalized DBMs such that $M = \mathit{Unreset}_Z(N)$ for some $Z \subseteq \mathcal{C}$. Then, for any PSM P, there exists a PSM P' such that for all valuations $\nu\colon \mathbf{k} \to \mathbb{N}$, there exists $\delta_0 > 0$ for which $M - P'[\nu] \cdot \delta = \mathit{Unreset}_Z(N - P[\nu] \cdot \delta)$ for all $\delta \in [0, \delta_0]$.*

Proof. We first consider the intersection of the shrunk DBM (N, P) and the DBM representing the set $Z = 0$. From Lemma. 4.5, this can be written as a shrunk DBM (N', P'), where N' is the normalized DBM obtained after intersecting N with the DBM for $Z = 0$. The second step consists in removing the constraints involving clocks of Z. This is achieved on DBMs by turning all the values in the corresponding columns to ∞; for shrunk DBMs, we change $(M_{i,j}, P_{i,j})$ into $(\infty, 0)$ whenever $i \in Z$ or $j \in Z$ except when $i \neq j$, the value of $P_{i,j}$ being actually not relevant in that case. This results in a shrunk DBM (M, Q) with the required properties.

Note that if (N, P) is empty, then so is (M, Q) since we did not change the diagonal, and that the components of the shrinking matrix can only increase during normalization. □

Finally, we compute the time-predecessor set of a DBM by dropping the lower-bound constraints on single clocks (*i.e.*, the first row of the DBM), while preserving all other constraints.

Lemma 4.7. *Let M and N be two normalized DBMs such that $M = \mathit{Pre}_t(N)$. Then for any PSM P, there exists a PSM P' such that for all valuations $\nu\colon \mathbf{k} \to \mathbb{N}$, there exists $\delta_0 > 0$ for which $M - \delta \cdot P'[\nu] = \mathit{Pre}_t(N - \delta \cdot P[\nu])$ for all $\delta \in [0, \delta_0]$.*

Proof. P' is obtained from P by also changing all the elements of the first row into 0 (except for the component $(0, 0)$), and applying normalization. Note that if (N, P) is empty, so is (M, Q), as in the previous lemma, since the diagonal of the shrinking matrix can only increase. □

By combining the previous lemmas, we immediately get the following proposition.

Proposition 4.8. *Let $f\colon \mathcal{M}_{C+1}(\mathbb{Q}_\infty)^n \to \mathcal{M}_{C+1}(\mathbb{Q}_\infty)$ be an elementary function and $M_0, M_1, \dots, M_n$ normalized DBMs satisfying $M_0 = f(M_1, \dots, M_n)$. For any PSMs $P_1, \dots, P_n$, one can compute a PSM P_0 s.t. for all $\nu\colon \mathbf{k} \to \mathbb{N}$, there exists a (computable) $\delta_0 > 0$ such that*

$$M_0 - \delta P_0[\nu] = f(M_1 - \delta P_1[\nu], \dots, M_n - \delta P_n[\nu]),$$

for all $\delta \in [0, \delta_0]$. These computations can be carried out in polynomial time, and in particular P_0 has size polynomial in the size of $P_1, \dots, P_n$ and f.

Observe that in the above equation, for any valuation ν, $(M, P_0[\nu])$ is empty whenever some $(M_i, P_i[\nu])$ is empty.

Proposition 4.8 gives a polynomial-size PSM P_0 provided that we represent the max-plus polynomials by sharing subexpressions. For instance, we assume that if we have max-plus polynomials ϕ_1 and ϕ_2 in memory, then the expression $\max(\phi_1, \phi_2)$ is represented by a new node max() that points to ϕ_1 and ϕ_2. More precisely, these expressions can be given as max-plus graphs defined in Section 5.2. Then, each elementary operation adds a polynomial number of nodes (in the number of clocks), thus yielding a polynomial-size representation.

The bound δ_0 can be computed when successively applying Lemmas 4.4–4.7, using Lemma 4.2. In fact, such a choice of δ ensures that all shrunk DBMs encountered in the computations are non-empty and normalized or $\delta \in [0, \delta_0)$. We show that δ_0 can also be chosen simply as (roughly) the inverse of the maximal component of all shrinking matrices that appear in all computations.

Remark 4.9. *If the equation of Proposition 4.8 has a non-empty shrunk solution, then one can choose $\delta_0 = \frac{1}{3m}$, where m is the maximum of all components of the shrinking matrices that appear in the computations.*

Proof. Assume that for some valuation ν, $M_i - \delta Q_i$ satisfies the equation for all small enough $\delta > 0$, where $Q_i = P_i[\nu]$. We will show that the equation is then satisfied for all $\delta \in [0, \frac{1}{3m}]$. To prove this, we need to show that for all $\delta \in [0, \frac{1}{3m})$, all shrunk DBMs are non-empty and satisfy the equations they are involved in.

Since all shrunk DBMs are non-empty (for small enough $\delta > 0$), all diagonals of the shrinking matrices are equal to 0. Thus, it suffices to show that all shrunk DBMs $M - \delta Q$ are normalized. Normalization condition requires

$$\forall i, j, k \in \mathcal{C}_0^3, \quad M_{i,j} - \delta Q_{i,j} \leq M_{i,k} - \delta Q_{i,k} + M_{k,j} - \delta Q_{k,j},$$

for all $\delta \in [0, \delta_0]$. If $M_{i,j} = M_{i,k} + M_{k,j}$, then we must have $Q_{i,j} \geq Q_{i,k} + Q_{k,j}$. So, for these components, the condition holds for all $\delta > 0$. If $M_{i,j} < M_{i,k} + M_{k,j}$, then the condition holds if $\delta_0 \leq |\frac{M_{i,k}+M_{k,j}-M_{i,j}}{Q_{i,k}+Q_{k,j}-Q_{i,j}}|$, but this is already the case since $\delta_0 \leq \frac{1}{3m}$ (the upper bound is infinity if the denominator is 0).

It remains to show that all equations that appear in the computations hold. For an equation of the form $(M, Q) = (N_1, R_1) \cap (N_2, R_2)$, we have either $(N_1)_{i,j} = (N_2)_{i,j}$ and $Q_{i,j} = \max((R_1)_{i,j}, (R_2)_{i,j})$, or for example $(N_1)_{i,j} < (N_2)_{i,j}$ and $(M_{i,j}, Q_{i,j}) = ((N_1)_{i,j}, (Q_1)_{i,j})$. In the former case, the equation holds for all $\delta > 0$. In the latter case, it holds for all $\delta < |\frac{(N_2)_{i,j}-(N_1)_{i,j}}{(Q_2)_{i,j}-(Q_1)_{i,j}}|$. This is already the case since $\delta < \frac{1}{3m}$. For equations of the form $(M, Q) = \mathrm{Pre}_{\mathrm{t}}((N, R))$, it suffices to choose δ small enough to ensure that normalization holds. For $(M, Q) = \mathrm{Unreset}_r((N, R))$, one only needs to ensure that the intersection with $r = 0$ and normalization holds. Both conditions were already shown to hold. ☐

How much can m grow? A rough estimation shows that it is at most exponential in the size of the equation. We haven't observed such a growth in practice.

Let us give an insight into the use of operations on shrunk DBMs to treat shrinkability problems. The following example explains the computation of the shrinking parameters on the timed automata of Fig. 1 and 3.

Example 4.10. *We consider the timed automaton of Fig. 1, with $g_1 = 1 \leq x, y \leq 3 \wedge 0 \leq x - y \leq 3$ the guard of the edge from ℓ_1 to ℓ_2, $R_1 = \{y\}$ its reset set, and $g_2 = 1 \leq x \leq 4 \wedge x - y \leq 3$ the guard of the edge from ℓ_2 to ℓ_3. As in Example 4.1, we shrink the guard g_1 into*

$$g_1' = 1 + k_1\delta \leq x \leq 3 - k_2\delta \wedge 1 + k_3\delta \leq y \leq 3 - k_4\delta \wedge k_5\delta \leq x - y \leq 2 - k_6\delta,$$

and g_2 into

$$g_2' = 1 + k_7\delta \leq x \leq 4 - k_8\delta \wedge k_9\delta \leq y \wedge x - y \leq 3 - k_{10}\delta.$$

Assume that we are looking for a valuation of $\mathbf{k} = (k_i)_{1 \leq i \leq 10}$ in $\mathbb{N}_{>0}$ for which the resulting shrunk automaton $\mathcal{A}_{-\mathbf{k}\delta}$ witnesses the shrinkability w.r.t. simulation, i.e. it can time-abstract simulate $\mathcal{A}$ for small enough $\delta > 0$. According to our definition of shrinkability, the simulator sets of $\mathcal{A}_{-\mathbf{k}\delta}$ must be shrinkings of the simulator sets of $\mathcal{A}$. Let us concentrate on three interesting simulation classes: all states $\ell_3 \times \mathbb{R}_{\geq 0}^{\mathcal{C}}$ are simulation-equivalent and can be extended to an infinite run, the set of states $X = [\![x \leq 4 \wedge 0 \leq x - y \leq 3]\!]$ at location ℓ_2 are those that can go to ℓ_3 by a b action, and the set of states $Y = [\![x, y \leq 3 \wedge 0 \leq x - y \leq 3]\!]$ at location ℓ_1 can go to X by an a action. One can see that X is precisely the time-predecessors of g_2, that is

$$X = Pre_t(g_2) \tag{1}$$

Further,

$$Y = Pre_t(g_1 \cap Unreset_y(X)) \tag{2}$$

expresses the fact that some point of $\ell_2 \times [\![X]\!]$ can be reached in one step starting from $\ell_1 \times [\![Y]\!]$, and this defines Y. Now, we use these equations to compute the simulator sets when guards are shrunk. Let X' denote the shrunk DBM describing time-predecessors of g_2', as given by Lemma 4.7. We have $X' = [\![x \leq 4 - k_8\delta \wedge x - y \leq 3 - k_{10}\delta]\!]$, and $\ell_2 \times X'$ is indeed the set of states of $\mathcal{A}_{-\mathbf{k}\delta}$ that can simulate the states $\ell_2 \times X$ of $\mathcal{A}$, for any given valuation and small enough $\delta > 0$. Let us now compute Y', the simulator set in $\mathcal{A}_{-\mathbf{k}\delta}$ of the states Y of $\mathcal{A}$, applying Lemmas 4.4–4.7 to the equation relating Y to X. In the figure below, the union of dark gray and light gray areas illustrate this equation for $\delta = 0$, while the dark gray areas illustrate the equation between shrunk *zones, i.e. between X' and Y'. We have:*

$$\begin{aligned}
Unreset_y(X') \quad &= [\![x \leq 3 - k_{10}\delta]\!],\\
[\![g_1']\!] \cap Unreset_y(X') \quad &= [\![1 + \max(k_1, k_3 + k_5)\delta \leq x \leq 3 - \max(k_2, k_{10})\delta\\
&\quad \wedge 1 + k_3\delta \leq y \leq 3 - \max(k_4, k_5 + \max(k_2, k_{10}))\delta\\
&\quad \wedge k_5\delta \leq x - y \leq 2 - \max(k_6, k_3 + \max(k_2, k_{10}))\delta]\!],\\
Y' = Pre_t([\![g_1']\!] \cap Unreset_y(X')) \quad &= [\![k_5\delta \leq x \leq 3 - \max(k_2, k_{10})\delta\\
&\quad \wedge y \leq 3 - \max(k_4, k_5 + \max(k_2, k_{10}))\delta\\
&\quad \wedge k_5\delta \leq x - y \leq 2 - \max(k_6, k_3 + \max(k_2, k_{10}))\delta]\!].
\end{aligned}$$

The calculations are illustrated in Example 4.10. We now have at hand both parameterized expressions for the simulator sets X' and Y', given parameterized shrunk guards g_1' and g_2'. It remains to choose a valuation, and check that X' and Y' are non-empty for small enough $\delta > 0$. We choose the valuation that sets $k_1 = k_4 = k_6 = 2$ and other parameters to 1. Note here that some parameters are set to 2 so that the shrunk guards are normalized.[4] *We get that under this valuation, $X' = [\![x \leq 4 - \delta \wedge x - y \leq 3 - \delta]\!]$ and $Y' = [\![\delta \leq x \leq 3 - \delta \wedge y \leq 3 - 2\delta \wedge \delta \leq x - y \leq 2 - 2\delta]\!]$. These sets and the guards are non-empty, and all equations above hold for all $\delta \in [0, \frac{1}{6}]$. This bound can be derived by looking at each application of Lemma 4.4–4.7 in our computations. Hence, we obtained a shrunk timed automaton $\mathcal{A}_{-\mathbf{k}\delta}$ that can time-abstract simulate $\mathcal{A}$, and expressions for the simulator sets parameterized by δ.* [5]

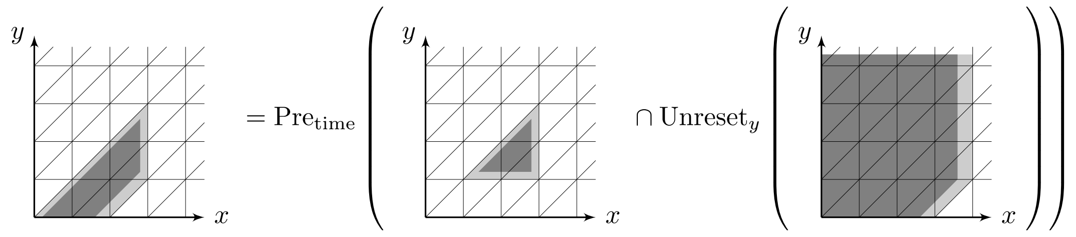


Figure 6: The calculations of Example 4.10. The light and dark gray areas combined describe the equation $Y = \mathrm{Pre}_t(g_1 \cap \mathrm{Unreset}_y(X))$, while the dark areas describe the shrunk version: $Y' = \mathrm{Pre}_t(g_1' \cap \mathrm{Unreset}_y(X'))$.

In Example 4.10, we guessed a valuation $\mathbf{k}$ that witnessed the shrinkability of the considered timed automaton. The fact that the timed automaton did not contain cycles simplified the constraints on

[4]We could instead apply normalization to g_1' and g_2' and set all parameters to 1.

[5]Note that the initial state $\ell_1 \times \mathbf{0}$ is not included in X' unless we set $k_5 = 0$. This can be tested easily. Although Definition 2.1 requires simulation to hold at initial states, this requirement can be relaxed depending on the application.

this valuation. The aim of the next section is to express systematically all constraints on parameters $\mathbf{k}$ induced by the given equations, even in presence of cyclic dependencies, and provide an algorithm to compute a valuation satisfying these constraints.

4.2. Equations on shrunk DBMs

We consider fixpoint equations on DBMs and study whether by "shrinking" a given solution, one can still satisfy the equation. Our goal is to generalize the arguments of Example 4.10 where we were able to compute a shrinking of the solutions to Equations (1) and (2). We consider fixpoint equations of the following form.

$$M_i = f_i(M_1, \dots, M_n, M_{n+1}, \dots, M_{n+n'}), \quad \forall 1 \le i \le n, \tag{3}$$

where $M_1, \dots, M_{n+n'}$ are unknown normalized DBMs and f_i's are elementary functions. Notice that $M_{n+1}, \dots, M_{n+n'}$ are *unconstrained*, *i.e.* they do not appear in the left hand side of the equation. Let us write $m = n + n'$. For instance, in Example 4.10, DBMs g_1 and g_2 are unconstrained since they only appear in the right hand side of the equations, where as X and Y do appear in the left hand side.

We assume we are given a solution $(M_i)_{1\le i\le m}$ to Equation (3), and we are interested in *shrunk solutions* defined as follows.

Definition 4.11. *Consider a solution $(M_i)_{1\le i\le m}$ of* (3)*. A* shrunk solution of (3) w.r.t. $(M_i)_{1\le i\le m}$ *is a triple*

$$\big((M_i)_{1\le i\le m}, (Q_i)_{1\le i\le m}, \delta_0\big),$$

where $\delta_0 > 0$ and Q_i's are shrinking matrices such that for all $\delta \in [0, \delta_0]$, $(M_i - \delta Q_i)_{1\le i\le m}$ is a solution of (3)*. A shrunk solution is called the* greatest shrunk solution *if $(Q_i)_{1\le i\le m}$ are the least shrinking matrices that define a shrunk solution w.r.t. $(M_i)_{1\le i\le m}$.*

Notice that we define shrunk solutions with shrinking matrices, not with parameterized ones. Here, the "least shrinking matrices" refer to the order on $\mathbb{N}^k$ defined by $\mathbf{a} \le \mathbf{b}$ if, and only if $a_i \le b_i$ for all $1 \le i \le k$. Clearly, the sets X' and Y' we computed in Example 4.10 are a shrunk solution (and in fact the least one) with respect to the solution (X, Y). A non-empty shrunk solution is a shrunk solution whose all shrunk DBMs are non-empty.

We will now show that the problem of finding shrunk solutions can be reduced to solving fixpoint equations in the max-plus algebra. For a non-empty solution $(M_i)_{1\le i\le m}$ of (3), consider parameterized shrinking matrices P_i, for all $1 \le i \le m$, where each cell of each P_i is a unique parameter in $\mathbf{k}$. By Proposition 4.8, there exist matrices $(\phi_i)_{1\le i\le n}$ of max-plus polynomials such that

$$\big(M_i, \phi_i(P_1, \dots, P_m)[\nu]\big) = f_i\big((M_1, P_1[\nu]), \dots, (M_m, P_m[\nu])\big),$$

for all $1 \le i \le n$, and any valuation ν. Here, $\phi_i(P_1, \dots, P_m)$ denotes a matrix whose components are max-plus polynomials that combine the components of matrices P_j. The above equation suggests that we study the following fixpoint equation on PSMs P_i's:

$$\begin{aligned} &P_i = \phi_i(P_1, \dots, P_m), \quad \forall 1 \le i \le n, \\ &[P_i]_{j,j} = 0, \quad \forall 1 \le i \le m, 1 \le j \le |\mathcal{C}_0|. \end{aligned} \tag{4}$$

If some valuation ν satisfies (4), then if we denote by $Q_i = P_i[\nu]$ for all i, we get that $\big((M_i)_{1\le i\le m}, (Q_i)_{1\le i\le m}, \delta_0\big)$ is a shrunk solution of (3), for some $\delta_0 > 0$. In fact, the first line means that the fixpoint equation holds (by Prop. 4.8), and the second line means that this is a non-empty shrunk solution. Note that requiring the diagonals to be zero is sufficient since we assume all shrunk DBMs to be normalized. The converse also holds: the shrinking matrices of any shrunk solution of (3) satisfies (4). By Prop. 4.8, the size of the equation (4) is polynomial in the size of (3).

Note that we will be often interested in solutions of (4) where some parameters are positive. These parameters will, for instance, correspond to the shrinking of the guards. In order to enforce positive values to some parameters, one can augment Equation (4) with the following constraint, for any matrix cell $[P_i]_{j,k}$ that is to be positive:

$$[P_i]_{j,k} = \max(1, [P_i]_{j,k}). \tag{5}$$

Remark 4.12. *The choice of the positive parameters depend on the problem at hand. For instance, in Section 6, we will shrink all the atomic constraints of given timed automata by positive amounts, so all parameters will be positive. One can nonetheless be interested in shrinking only the guards of some of the edges, in which case, one would add the constraints* (5) *only for those parameters that need to be positive.*

Let us define $\mathcal{P}(x) = 0$ if $x = 0$ and $\mathcal{P}(x) = \infty$ otherwise. We extend $\mathcal{P}(\cdot)$ to matrices, by componentwise application. The correspondence between shrunk solutions and max-plus equations is formally stated in the following lemma.

Lemma 4.13. *Consider any non-empty solution $(M_i)_{1\le i\le m}$ of* (3)*, and the max-plus polynomial matrices $(\phi_i)_{1\le i\le n}$ as defined above. Then,*

- *For all shrinking matrices $(Q_i)_{1\le i\le m}$, there exists $\delta_0 > 0$ such that $\big((M_i)_{1\le i\le m}, (Q_i)_{1\le i\le m}, \delta_0\big)$ is a non-empty shrunk solution of* (3) *if, and only if, $(Q_i)_{1\le i\le m}$ is a solution of* (4) *in $\mathbb{N}$.*
- *$\big((M_i)_{1\le i\le m}, (Q_i)_{1\le i\le m}, \delta_0\big)$ is the greatest shrunk solution of* (3) *if, and only if $(Q_i)_{1\le i\le m}$ is the least solution of* (4) *in $\mathbb{N}$.*
- *If* (4) *has a solution $(Q_i)_{1\le i\le m}$, then for any shrinking matrices $R_{n+1}, \ldots, R_m$ such that $\mathcal{P}(R_j) \ge \mathcal{P}(Q_j)$ for $n+1 \le j \le m$, there exist $R_1, \ldots, R_n$ such that $(R_i)_{1\le i\le m}$ is the least shrunk solution of* (4)*, computable in polynomial time.*

The first two statements follow directly from the previous paragraph and Prop. 4.8. The third statement follows easily from Theorem 5.1 given in the next section, where we study the efficient computation of the solutions to fixpoint equations in the max-plus algebra, as in Equation (4).

Example 4.14. *As we noted earlier, the equation system of Example 4.10 was simple to solve manually because it was not recursive (we were able to first solve* (1) *and then* (2)*). We consider here a simple example to illustrate the need for general fixpoint equation systems.*

Consider a timed automaton with clocks x, y, and a self-loop guarded by $0 \le x, y \le 1$ that resets x. Clearly, whenever the lower bound on x is shrunk (say, to $\delta \le x$) then the system contains no infinite runs, thus the timed automaton is not simulation-shrinkable. Let us show how this can be seen by our results. Consider X the set of states that can take (infinitely many times) the self-loop. This set satisfies the following fixpoint equation

$$X = \mathit{Pre}_t(G \cap \mathit{Unreset}_x(X)), \tag{6}$$

where G stands for the guard. The greatest fixpoint can be seen to be the set $X = [\![0 \le x, y \le 1]\!]$. We see the set G as a variable, so (G, X) is a solution of this equation. We will now compute shrunk solutions w.r.t. (G, X). Let consider the following shrunk DBMs with fresh parameters:

$$(X, P) = \left(\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{pmatrix}, \begin{pmatrix} k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 \\ k_7 & k_8 & k_9 \end{pmatrix} \right)$$
$$(G, Q) = \left(\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{pmatrix}, \begin{pmatrix} k_{10} & k_{11} & k_{12} \\ k_{13} & k_{14} & k_{15} \\ k_{16} & k_{17} & k_{18} \end{pmatrix} \right)$$

We will instantiate (4) for this equation. We calculate $Pre_t((G, Q) \cap Unreset_x((X, P)))$, which yields (X, R) where R is

$$\begin{pmatrix} m(k_1, k_2, k_{10}) & 0 & 0 \\ m(k_{13}, k_{15} + m(k_{16}, k_2 + m(k_7, k_8))) & m(k_5, k_{14}) & m(k_{15}, k_{13} + m(k_{12}, k_2 + k_3)) \\ m(k_{16}, k_2 + m(k_7, k_8)) & m(k_{17}, k_{11} + m(k_{16}, k_2 + m(k_7, k_8))) & m(k_9, k_{18}) \end{pmatrix}$$

where we m represents max. *Since we are looking for non-empty solutions, we must set $k_1 = k_5 = k_9 = k_{10} = k_{14} = k_{18} = 0$. Furthermore, we would like the parameters $k_{13}, k_{16}, k_{11}, k_{12}$ to be positive so that the guard will be shrunk by a positive amount on all sides. So Equation (4) is written, for this case, as follows.*

$$\begin{aligned}
&k_1 = \max(k_1, k_2, k_{10}) \\
&k_4 = \max(k_{13}, k_{15} + \max(k_{16}, k_2 + \max(k_7, k_8))) \\
&k_5 = \max(k_5, k_{14}) \\
&k_6 = \max(k_{15}, k_{13} + \max(k_{12}, k_2 + k_3)) \\
&k_7 = \max(k_{16}, k_2 + \max(k_7, k_8)) \\
&k_8 = \max(k_{17}, k_{11} + \max(k_{16}, k_2 + \max(k_7, k_8))) \\
&k_9 = \max(k_9, k_{18}) \\
&k_i = \max(1, k_i), \forall i \in \{1, 2, 3, 5, 9, 10, 14, 18\}.
\end{aligned} \tag{7}$$

Is there a solution to this equation? If we concentrate on the two following lines, it is easy to see that there is none:

$$\begin{aligned}
&k_8 = \max(k_{17}, k_{11} + \max(k_{16}, k_2 + \max(k_7, k_8))) \\
&k_{11} = \max(1, k_{11})
\end{aligned}$$

In fact, the first line implies that k_8 should be no less than $k_{11} + k_8$, while the second line requires $k_{11} \ge 1$, a contradiction.

In the next section, we will study general polynomial fixpoint equations as (7), and show how to check for solutions and solve them. These results will then allow us to define our algorithms for shrinkability.

5. Max-Plus Algebra

5.1. Max-plus equations

In PSMs, formal expressions using maximization and sum are manipulated. The set $\mathbb{R}_{\ge 0}$ 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 [29]. The purpose of this section is to show how to solve polynomial fixpoint equations in the max-plus algebra.

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 are interested in computing solutions of fixpoint equations of the following form:

$$k_i = \phi_i(k_1, \ldots, k_n, k_{n+1}, \ldots, k_{n+n'}), \quad \forall 1 \leq i \leq n. \tag{8}$$

Notice that variables $k_{n+1}, \ldots, k_{n+n'}$ only appear at the right hand side of the equation. Equation (8) defines a *non-linear* equation (polynomials ϕ_i have arbitrary degrees). We call these equations *max-plus polynomial fixpoint equations.* The equations between parameters that we derived in the previous section fall into this category. Although Tarski's Theorem [30] 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. Note that some algorithms have been studied to compute fixpoint equations of polynomial equations on general ω-continuous semirings, see [31]. Our specialized algorithm provides a more efficient solution, and can be used to show additional properties such as the second part of the following theorem.

Theorem 5.1. *The existence of a solution of a given max-plus polynomial fixpoint equation is decidable in polynomial time in the size of the equation.*

Moreover, if there is a solution $\mathbf{v}$ *in* $\mathbb{N}$ *to a given equation* E*, then for any values* $v'_{n+1}, \ldots, v'_{n+n'} \in \mathbb{N}$ *where* $v_{n+i} > 0 \Rightarrow v'_{n+i} > 0$ *for all* $1 \leq i \leq n'$*, equation* E *with the additional constraints* $\{k_{n+i} = v_{n+i}\}_{1 \leq i \leq n'}$ *has a least solution, computable in polynomial time.*

As in the previous section, we assume that expressions can be shared in equations given as input to the above theorem. Such a data structure is detailed in the next subsection. The second point of the theorem states that the existence of solutions does not depend on the exact values of the unconstrained variables, but only on their positiveness.

These results rely on an analysis of max-plus graphs, that we associate to max-plus equations. The rest of this section defines these graphs and gives an algorithm to solve these equations.

5.2. Max-Plus Graphs

Let $\mathbf{k}$ be a set of parameters. A *max-plus graph* G *with parameters* $\mathbf{k}$ is a directed graph (V, E), where V is the set of *nodes*, and $E \subseteq V \times V$ the set of *arcs*. The node set V is partitioned into $V = \mathbf{k} \cup \mathbf{N} \cup \mathbf{Max} \cup \mathbf{Plus}$. There is one node for each parameter, and also some additional nodes labelled by natural numbers, and others labelled either by max or by plus. The graphs satisfy the constraint that each (directed) cycle contains at least one node in $\mathbf{k}$. We identify nodes $n \in \mathbf{N}$ with the natural number they represent. We will have at most one node labelled by each natural number.

Intuitively, a max-plus graph encodes the relations between the parameters $\mathbf{k}$, where a directed path from parameter k to k' means that k' is greater than or equal to k in any solution. Formally, given a max-plus graph G, a mapping $\nu \colon \mathbf{k} \to \mathbb{N}$ is called a *valuation* of G. An *extended valuation* $\overline{\nu}$ is the extension of a valuation ν to all nodes of G. An example is given in Fig. 7.

Definition 5.2. *A valuation* $\nu : \mathbf{k} \to \mathbb{N}$ *is a* solution *of a max-plus graph* G *with parameters* $\mathbf{k}$ *if there exists some extended valuation* $\overline{\nu}$*, that satisfies the following conditions.*

- *For all* $k \in \mathbf{k}$, $\overline{\nu}(k) = \max(\overline{\nu}(k'))$ *where the max is over all predecessors* k' *of* k*,*
- *For all* $n \in \boldsymbol{N}$, $\overline{\nu}(n) = n$.

- *For all $p \in$ **Plus**, $\overline{\nu}(p) = \sum_{p'} \overline{\nu}(p')$, where the sum is over the predecessors p' of p.*
- *For all $m \in$ **Max**, $\overline{\nu}(m) = \max_{m'}(\overline{\nu}(m'))$ where the max is over the predecessors m' of m.*

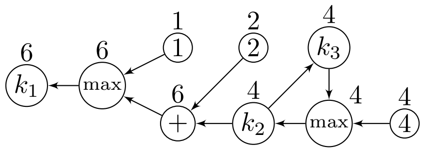


Figure 7: The figure shows a max-plus graph containing the parameter nodes k_1, k_2, k_3. The numbers given above the nodes are an extended valuation. One can notice that at all nodes but the plus node, the values are at least as large as the values of the successors of the node. At the plus node, the given value 6 is the sum of the values of the successors. One can check that the given extended valuation is the least extended valuation in this graph, thus defines the least solution.

Note that if we are interested in a positive integer solution of a max-plus graph, we can simply add one edge from node 1 to each parameter k. We extend the order $\leq$ to vector of numbers as $(a_i)_{i\in I} \leq (b_i)_{i\in I}$ iff $a_i \leq b_i$ for all $i \in I$.

Given a graph $G = (V, E)$, a *path* from node v_1 to node v_k is a sequence $v_1 v_2 \dots v_k$ of nodes where $(v_i, v_{i+1}) \in E$ for all $1 \leq i \leq k-1$. A *simple cycle* is a path $v_1 \dots v_k$ such that $v_1 = v_k$ and nodes $v_1, \dots, v_{k-1}$ are pairwise distinct. A node v' is *reachable* from a node v if there is a path starting at v and ending at v'. For two nodes $v, v' \in V$, we write $v \xrightarrow{+} v'$, if there is a path $v = v_1 \dots v_k = v'$ such that $k \geq 2$, and $v_i \in$ **Plus** for some $i \in \{1, \dots, k\}$.

The following lemma gives a graph theoretical characterization of max-plus graphs that have solutions. A simple cycle of a max-plus graph is a *bad cycle* if it contains at least one **Plus** node, and at least one node that is reachable from a node $n \in \mathbf{N}$ with $n \geq 1$. A *contradicting path* is a path from k to l, for some $k, l \in \mathbf{N}$ with $k > l$.

Lemma 5.3. *A max-plus graph G with parameters $\mathbf{k}$ has a solution if and only if it has no bad cycle or contradicting path. Moreover, if G has a solution, then it has a least solution which can be computed in polynomial time in the size of G.*

Proof. Clearly, if G has contradicting paths, it has no solution. Let us show that if G has a bad cycle, then it does not have any solution.

Consider a simple bad cycle c in G that contains a plus node p, and suppose there is an extended solution $\overline{\nu}$. We know that p has one predecessor in p' in c, and another one outside, let us call it p''. By hypothesis, all nodes of c are reachable from a node $n \geq 1$, which implies $\overline{\nu}(p'') \geq 1$. We have $\overline{\nu}(p) \geq \overline{\nu}(p') + \overline{\nu}(p'') \geq \overline{\nu}(p') + 1$. But since c is a cycle, combining inequalities satisfied by $\overline{\nu}$ along c, we get $\overline{\nu}(p') \geq \overline{\nu}(p)$, therefore $\overline{\nu}(p') \geq \overline{\nu}(p') + 1$, which is a contradiction.

We now prove that if there are no contradicting paths and bad cycles, then G has a solution, which we will construct explicitly. Consider first all nodes W of G that are not reachable from any node $n \in \mathbf{N}$ with $n \geq 1$. We can safely assign the value 0 to all the nodes of W (this doesn't contradict the max-plus graph since none of these nodes are reachable from a positive number). Then, we replace all nodes of W by the node $0 \in$ **Nat**. Let us call G' the graph obtained from G in this manner. Clearly, if G' has a solution, so does G, by extending the solution with $W \cap \mathbf{k} \mapsto 0$. Moreover, G' can be computed in polynomial time.

If $\mathbf{k} \subseteq W$ then we are done. Otherwise, let us assume w.l.o.g. that all the nodes of G are reachable from a positive number. We are now looking for a solution in $\mathbb{N}_{>0}$. Then, by hypothesis, the relation $\xrightarrow{+}$ is anti-symmetric. We partition the set of nodes of G into $C_0, C_1, \ldots, C_m$ such that C_0 is the minimum for the relation $\xrightarrow{+}$, and each C_i is the set of nodes whose all $\xrightarrow{+}$-predecessors are in $C_0 \cup \ldots \cup C_{i-1}$ and that has at least one $\xrightarrow{+}$-predecessor in C_{i-1}. This partition can be seen as a topological sort for the relation $\xrightarrow{+}$.

Observe that C_0 is not empty by hypothesis (in fact, there is at least one node that is reachable from a positive integer and there is no bad cycle) and that any plus node of C_i has both its predecessors in $C_0 \cup \ldots \cup C_{i-1}$. Also, there is no path in G from C_i to C_j for $j < i$ by construction. Let $G[C_i]$ denote the graph G restricted to nodes of C_i. For all $1 \leq i \leq m$, given a *lower bound* $\iota_i \colon C_i \to \mathbb{N}_{>0}$, there is a unique solution ν_i of $G[C_i]$ such that $\iota_i(v) \leq \nu_i(v)$ for all $v \in C_i$. In fact, the solutions of $G[C_i]$ which respect ι_i are exactly the solutions of the max-plus linear equation $x = Ax \oplus b$, defined by $A_{k,l} = 0$ if there is an arc (l, k) in $G[C_i]$ and $A_{k,l} = -\infty$ otherwise; and vector b is defined as $b_k = \iota_i(k)$ for all k. [6] Solutions to this linear fixpoint equations exist in $\mathbb{N}_{>0}$, and A^*b is one, and is in fact the least solution, where A^* is defined as $A^*_{k,l} = 0$ if there is path from l to k in $G[C_i]$, and $-\infty$ otherwise. This can be computed in polynomial time (see Section 3 of [29]).

We will define ν iteratively for each C_i, $i \geq 0$. For $i = 0$, we let ν_0 be the least solution of $G[C_0]$ for ι_0 defined by $\iota_0(n) = n$ for $n \in \mathbf{N} \cap C_0$ and $\iota_0(v) = 1$ for all $v \in C_0 \setminus \mathbf{N}$. At step $i \geq 0$, suppose we are given valuations $\nu_0, \ldots, \nu_{i-1}$ and a lower bound ι_i. We compute ν_i as the least solution of $G[C_i]$ that respects ι_i, and we define ι_{i+1} on C_{i+1} as follows. For all $u \in C_{i+1} \cap \mathbf{k}$, we let $\iota_{i+1}(u) = \max_{v:(v,u)\in E \wedge v \in C_0 \cup \ldots \cup C_i} \nu(v)$; for all $u \in C_{i+1} \cap \mathbf{Max}$, we let $\iota_{i+1}(u) = \max(\nu(v_1), \nu(v_2))$ where $v_1, v_2 \in C_0 \cup \ldots \cup C_i$ by construction; and for all $u \in C_{i+1} \cap \mathbf{Plus}$, we let $\iota_{i+1}(u) = \nu(v_1) + \nu(v_2)$ where $v_1, v_2 \in C_0 \cup \ldots \cup C_i$. (Note that $\mathbf{N} \subseteq C_0$). Clearly, any solution of G that extends $\nu|_{C_0 \cup \ldots \cup C_i}$ must satisfy this lower bound.

By construction, ν defines a solution. Moreover, we show that this is the least solution. In fact, since this is a finite solution, the least solution ν' given by Tarski's theorem is also finite. But we defined ν_0 as the least solution in C_0, so that there is no solution of G whose values in C_0 is less than ν_0. This means $\nu'|_{C_0} \geq \nu_0$. We show by induction that the lower bound ι_i is satisfied by $\nu'|_{C_i}$, for all i, so that we get $\nu'|_{C_i} \geq \nu_i$ by construction of ν_i. Hence, $\nu' = \nu$. $\square$

We now define max-plus graphs that encode the solutions of equations of the form (8). First, for a max-plus polynomial ϕ, let us define graph $G(\phi)$ associated to ϕ, as follows. Graph $G(\phi)$ is the syntactic binary tree of the expression ϕ, where the leaves are either constants or elements of $\mathbf{k}$, and each internal node corresponds to either plus or max, and joins the subtrees of the corresponding subexpressions. The root corresponds to the whole expression ϕ. We direct the edges of the tree bottom up. Now, define the *max-plus graph G associated to equation (8)* as the union of graphs $G(\phi_i)$ associated to each ϕ_i, where all nodes corresponding to same parameters are merged together. Moreover, an arc is added from the root node of each $G(\phi_i)$ to node k_i, and from k_i to the root. Notice that any cycle we create in this manner contains a parameter node. Observe that in our construction, each max- or plus-node of G corresponds to a unique subexpression in one of the ϕ_i's. The following lemma states that this graph encodes precisely the solutions of equation (8).

Lemma 5.4. *Let G be the max-plus graph associated to Equation* (8). *A valuation $\nu \colon \mathbf{k} \to \mathbb{N}$ is a solution of G if, and only if it is a solution of* (8). *Therefore, ν is the least solution of G if, and only if it is the least finite solution of* (8).

[6] In max-plus algebra, max is the addition and is denoted by $\oplus$, whereas the sum is the multiplication and is denoted by $\otimes$. The product of a matrix and a vector is defined as usual, where multiplication is $\otimes$ and addition is $\oplus$.

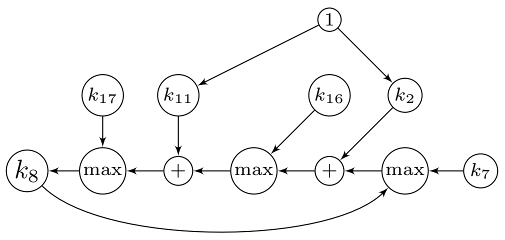
Proof. As noted above, there is a correspondence between the nodes labelled max or plus with the subexpressions. We use this to show how a solution of G can be seen as a solution of (8) and vice versa. Suppose $\overline{\nu}$ is an extended solution that proves that ν is a solution of G. Then, this defines a solution of (8) where $\nu(\mathbf{k})$ is the values given to $\mathbf{k}$, and $\nu(V \setminus \mathbf{k})$ define the values of the subexpressions in ϕ_i's. In fact, by construction, these are the "intermediate" values calculated in the subexpressions when one evaluates each ϕ_i. Using the same correspondence, one can see that a solution of (8) yields a solution of G whose extension correspond to these intermediate values. □

Proof of Theorem 5.1. The first statement of Theorem 5.1 now follows directly from Lemmas 5.3 and 5.4. For the second statement, define equation (E) from equation (8) by adding equality constraints $k_{n+i} = v_{n+i}$ for all $1 \leq i \leq n'$, for any $v_{n+1}, \ldots, v_{n+n'} \in \mathbb{N}_{>0}$. Assume that (8) has a solution where $k_{n+1}, \ldots, k_{n+n'}$ are given positive values. We show that then, the max-plus graph G' associated with equation (E), does not have bad cycles or contradicting paths, which means that it has a solution by above lemmas. In fact, in the max-plus graph G of (8), no cycle that is reachable from a node k_{n+i} contains a plus node, since this would contradict the existence of a solution with a positive value for k_{n+i}. Therefore, when we add the additional constraint $k_{n+i} = v_{n+i}$ we do not create any new bad cycles or contradicting paths. The result follows. □

Example 5.5. *In Example 4.14, we proved manually, by contradiction, that the considered equation had no solution. The contradiction was obtained by the following lines:*

$$\begin{aligned} &k_8 = \max(k_{17}, k_{11} + \max(k_{16}, k_2 + \max(k_7, k_8))) \\ &k_{11} = \max(1, k_{11}) \end{aligned}$$

We can apply the results of this section to detect the same contradiction. The graph below shows the max-plus graph associated to these two lines (this is a subgraph of the max-plus graph associated to the full equation). One can see that the graph contains a bad cycle: k_8 belongs to a cycle containing a plus node which is reachable from node 1.



6. Deciding shrinkability

We now apply the results we developed in previous sections to shrinkability.

6.1. Simulation-Shrinkability

We fix a closed timed automaton $\mathcal{A} = (\mathcal{L}, l_0, \mathcal{C}, \Sigma, E)$ with distinct labels, and a finite automaton $\mathcal{F}$ on the same alphabet Σ such that $[\![\mathcal{F}]\!] \sqsubseteq_{\text{t.a.}} [\![\mathcal{A}]\!]$. 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 be the reset set.

Computation of the Simulator Sets. Since all edge labels are distinct, the simulator set of each state f of $\mathcal{F}$ in $[\![\mathcal{A}]\!]$ can be expressed as the greatest fixpoint of the following equation:

$$S_f = \bigcap_{\sigma\in\Sigma} \bigcap_{f \xrightarrow{\sigma} f'} \mathrm{Pre}_{\mathrm{t}}(\mathrm{Unreset}_{R_\sigma}(S_{f'}) \cap [\![G_\sigma]\!]), \tag{9}$$

for all states f of $\mathcal{F}$, where $(S_f)_{f\in\mathcal{F}}$ are unknown sets of states of $\mathcal{A}$. The greatest fixpoint is well-defined since the operator on the right, denoted Ω, is non-decreasing.

We will use properties of the region equivalence in a timed automaton without formalizing this well-known construction (for that, we refer to [1]). Regions refine the largest time-abstract bisimulation in a timed automaton; therefore, if $(S_f)_f$ are finite unions of regions, then $(S'_f)_f = \Omega((S_f)_f)$ are also finite unions of regions. In particular, the largest fixpoint of (9) can be computed by a finite number of iterations of operator Ω, after having initialized all sets with the full set of states of $\mathcal{A}$. We also notice that if $(S_f)_f$ are convex sets, then so are $(S'_f)_f = \Omega((S_f)_f)$.

For every i, we write $(S^i_f)_f$ for the sets of states obtained after i iterations of Ω in the above-mentioned iterative computation. From the above discussion, the two following properties hold:

- for every state f of $\mathcal{F}$, $\text{ta-sim}_{[\![\mathcal{A}]\!]}(f) = \lim_{i\to\infty} S^i_f = S^{i_0}_f$ for some i_0;
- for every i, for every f, S^i_f is a finite convex union of regions.

We therefore deduce that the simulator set of each state f of $\mathcal{F}$ in $[\![\mathcal{A}]\!]$ can be expressed as the greatest fixpoint of the following equation on DBMs:

$$[\![M_f]\!] = \bigcap_{\sigma\in\Sigma} \bigcap_{f \xrightarrow{\sigma} f'} \mathrm{Pre}_{\mathrm{t}}(\mathrm{Unreset}_{R_\sigma}([\![M_{f'}]\!]) \cap [\![G_\sigma]\!]), \tag{10}$$

for all states f of $\mathcal{F}$, where $(M_f)_f$ are unknown DBMs.

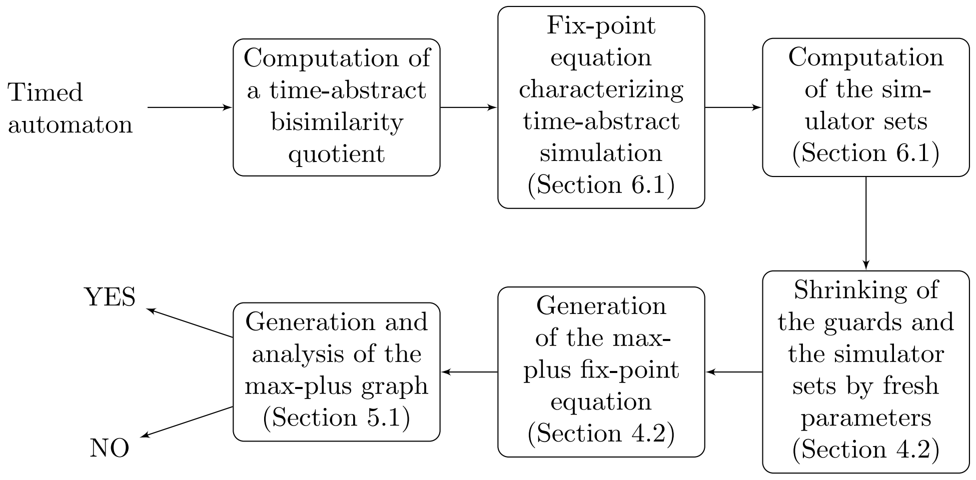
We now argue that the simulator sets can be computed in time pseudo-polynomial in $\mathcal{A}$ and $\mathcal{F}$. As argued above, one can compute this greatest fixpoint by initializing all M_f to unconstrained DBMs, and then iteratively computing approximations applying (10) until a fixpoint is reached. If M^i_f is the i-th iterative DBM for state f, we obviously have that $[\![M^i_f]\!] = S^i_f$. As each computed DBM represents a finite union of regions, it can be defined with constraints using integer constants between $-C$ and C, and all the computed normalized DBMs only use integer constants in $[-C\cdot|\mathcal{C}_0|, C\cdot|\mathcal{C}_0|] \cup \{-\infty, \infty\}$.

So, at each iteration, either the M_f's do not change, in which case the fixpoint has been reached, or some constant is decreased by at least one. Thus, fixpoint must be reached in at most $O((C\cdot|\mathcal{C}_0|)\cdot|\mathcal{F}|\cdot|\mathcal{C}_0|^2)$ iterations, and each computation takes time $O(|\mathcal{F}|\cdot|\mathcal{C}_0|^3)$, since the expression inside the intersection in (10) is computed, and all DBMs need be normalized for each edge of $\mathcal{F}$.

Globally, the simulator sets for all f's can therefore be computed in time $O(C\cdot|\mathcal{F}|^2\cdot|\mathcal{C}_0|^6)$, which is then pseudo-polynomial.

Computing Shrunk Simulator Sets When They Exist. Consider the greatest solution $(M_f)_f$ of (10). Including the G_σ's in the unknown DBMs, Equation (10) can be seen as an instantiation of Equation (3) (page 18) over DBMs $(M_f)_f \cup (G_\sigma)_\sigma$.

Solving simulation-shrinkability w.r.t. $\mathcal{F}$ consists in deciding if for some shrinking of the guards G_σ, there exist simulator sets that are shrinkings of the sets M_f's. So, solving simulation-



Timed automaton
Computation of a time-abstract bisimilarity quotient
Fix-point equation characterizing time-abstract simulation (Section 6.1)
Computation of the simulator sets (Section 6.1)
Shrinking of the guards and the simulator sets by fresh parameters (Section 4.2)
Generation of the max-plus fix-point equation (Section 4.2)
Generation and analysis of the max-plus graph (Section 5.1)
YES
NO

Figure 8: Summary of the simulation-shrinkability algorithm. If shrinkability holds (YES), then the analysis of the max-plus graph yields the least solution, thus, the shrinking of the guards and the simulator sets in the shrunk timed automaton. Note that the computation of a time-abstract bisimilarity quotient can be replaced by an input requiring a finite automaton.

shrinkability w.r.t. $\mathcal{F}$ means deciding whether (10) has a shrunk solution with respect to $(M_f)_f \cup (G_\sigma)_\sigma$ where the shrinking matrices of G_σ's are positive.[7] This can be decided by Lemma 4.13.

Simulation-shrinkability does not depend on how much the guards are shrunk. In fact, since G_σ's are unconstrained in (9), if there is a shrunk solution to (9) with positive shrinking matrices for G_σ's, then for any shrinking matrices $(K_\sigma)_\sigma$, Lemma 4.13 provides a (greatest) shrunk solution where the shrinking matrices for $(G_\sigma)_\sigma$ are fixed to $(K_\sigma)_\sigma$. Therefore, either all positive integer vectors $\mathbf{k}$, which yield normalized guards, witness the shrinkability of $\mathcal{A}$ (into $\mathcal{A}_{-\mathbf{k}\delta}$), or $\mathcal{A}$ is not simulation-shrinkable w.r.t. $\mathcal{F}$ for any value of $\mathbf{k}$.

Furthermore, one can also require *initial* simulation between $\mathcal{F}$ and a shrinking of $\mathcal{A}$, *i.e.* the initial state of $\mathcal{A}_{-\mathbf{k}\delta}$ should simulate the initial state f_0 of $\mathcal{F}$. In this case, it suffices to compute the shrunk simulator sets as above (if these exist), and then check whether the shrinking of the set M_{f_0} contains the valuation $\mathbf{0}$. This can be done efficiently, since it suffices to intersect the shrinking with this valuation and check for emptiness.

Now, solving simulation-shrinkability w.r.t. $\mathcal{F}$ only takes time polynomial in the size of the equation. So the overall complexity is pseudo-polynomial in $\mathcal{A}$ and $\mathcal{F}$. Note that, one can check simulation-shrinkability w.r.t. $\mathcal{A}$, which requires the shrinking to time-abstract simulate $\mathcal{A}$, by chosing $\mathcal{F}$ as a time-abstract bisimulation quotient of $\mathcal{A}$. In this case, $\mathcal{F}$ has exponential size, so the procedure takes exponential time. Nevertheless, minimization can be used to compute the coarsest bisimulation quotient as in [25], which yields small equations in practice.

A summary of the several steps of the algorithm is given in Fig. 8.

[7]Let us call a shrinking matrix *positive*, if all its off-diagonal components are positive.

6.2. Non-blocking-Shrinkability

We fix a closed non-blocking timed automaton $\mathcal{A} = (\mathcal{L}, l_0, \mathcal{C}, \Sigma, E)$. 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 be the reset set. The following equation characterizes non-blockingness:

$$\forall \sigma \in \Sigma, \quad [\![G_\sigma]\!] \subseteq \bigcup_{\sigma' : (\sigma,\sigma') \in \Sigma_{E \circ E}} \mathrm{Unreset}_{R_\sigma}(\mathrm{Pre}_{\mathrm{t}}([\![G_{\sigma'}]\!])), \tag{11}$$

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

$$\forall \sigma \in \Sigma, \quad [\![G_\sigma]\!] = \bigcup_{\sigma' : (\sigma,\sigma') \in \Sigma_{E \circ E}} \mathrm{Unreset}_{R_\sigma}(\mathrm{Pre}_{\mathrm{t}}([\![G_{\sigma'}]\!])) \cap [\![G_\sigma]\!], \tag{12}$$

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:

$$\begin{aligned} \forall \sigma \in \Sigma, \quad [\![G_\sigma]\!] &= \bigcup_{\sigma' : (\sigma,\sigma') \in \Sigma_{E \circ E}} [\![M_{\sigma,\sigma'}]\!] \\ \forall \sigma, \sigma' \in \Sigma, \quad [\![M_{\sigma,\sigma'}]\!] &= \mathrm{Unreset}_{R_\sigma}(\mathrm{Pre}_{\mathrm{t}}([\![G_{\sigma'}]\!])) \cap [\![G_\sigma]\!] \end{aligned} \tag{13}$$

Fix a solution $(G_\sigma)_\sigma \cup (M_{\sigma,\sigma'})_{\sigma,\sigma'}$, which exists again by the non-blockingness assumption. We will solve the max-plus equation corresponding to the second part of (13) by Lemma 4.13, but we first add to this equation some inequalities which “encode” the first part of (13). We use the following technical lemma to choose these inequalities.

Lemma 6.1. *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. $[\![D - \delta Q]\!] = \bigcup_{1 \le i \le b} [\![C_i - \delta P_i]\!]$ for all $0 < \delta < \min(\delta_0, \delta_1)$, in polynomial space and in time $O(|\mathcal{C}_0|^{2b} p(|\mathcal{C}| + b))$, 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 $[\![D - \delta Q']\!] = \bigcup_{1 \le i \le b} [\![C_i - \delta P'_i]\!]$ for all small enough $\delta > 0$.

Proof. One can verify in polynomial time, whether for all $i \in \{1, \ldots, b\}$, $[\![C_i - \delta P_i]\!] \subseteq [\![D - \delta Q]\!]$ for all small enough $\delta \ge 0$: It suffices to check, for all x, y, whether $(C_i - \delta P_i)_{x,y} \le (D - \delta Q)_{x,y}$ for small enough $\delta \ge 0$, which holds if either $(C_i)_{x,y} < D_{x,y}$ or $(C_i)_{x,y} = D_{x,y}$ and $(P_i)_{x,y} \ge Q_{x,y}$. In the former case, we need to choose δ_1 so that $\delta(Q_{x,y} - (C_i)_{x,y}) < D_{x,y} - (C_i)_{x,y}$ for all $0 \le \delta \le \delta_1$, whereas the latter always holds.

It remains to verify that $[\![D - \delta Q]\!] \subseteq \bigcup_{1 \le i \le b} [\![C_i - \delta P_i]\!]$. This holds if and only if

$$\Big(\bigcup_{1 \le i \le b} [\![C_i - \delta P_i]\!] \Big)^c \cap [\![D - \delta Q]\!] = \bigcap_{1 \le i \le b} \bigcup_{(x,y) \in (\mathcal{C} \cup \{0\})^2} [\![x - y > (C_i - \delta P_i)_{x,y}]\!] \cap [\![D - \delta Q]\!] = \emptyset.$$

where $(\cdot)^c$ denotes the set complement, which is true if and only if for all $(x^{(1)}, y^{(1)}), \dots, (x^{(b)}, y^{(b)}) \in (\mathcal{C} \cup \{0\})^2$, $\bigcap_{1 \le i \le b} (\llbracket x^{(i)} - y^{(i)} > (C_i - \delta P_i)_{x^{(i)},y^{(i)}} \rrbracket \cap \llbracket D - \delta Q \rrbracket) = \emptyset$. But there are less than $(|\mathcal{C}_0|)^{2b}$ such terms, which are conjunctions of $b+1$ DBMs, so the emptiness of each term can be checked in polynomial time, as described above. The overall time complexity is then $O((|\mathcal{C}_0|)^{2b} p(|\mathcal{C}| + b))$. But this verification can be carried out in polynomial space since each term can be checked independently. Note that each term gives an upper bound on δ_1, so the equality holds choosing δ_1 as the minimum of these.

The last statement follows from the fact that the emptiness, for all small enough $\delta \ge 0$, of the disjuncts above only depends on the order between the parameters. □

Although we do not know whether the polynomial space complexity is optimal for the above problem, the high complexity is of little surprise since checking equality between a zone and a union of zones is a difficult problem in general even in the exact setting [32].

The second point of the lemma says that the satisfaction of the first part of (13) by a shrunk solution only depends on the relative ordering of the components 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 second part of (13) augmented with these guessed (in)equalities.

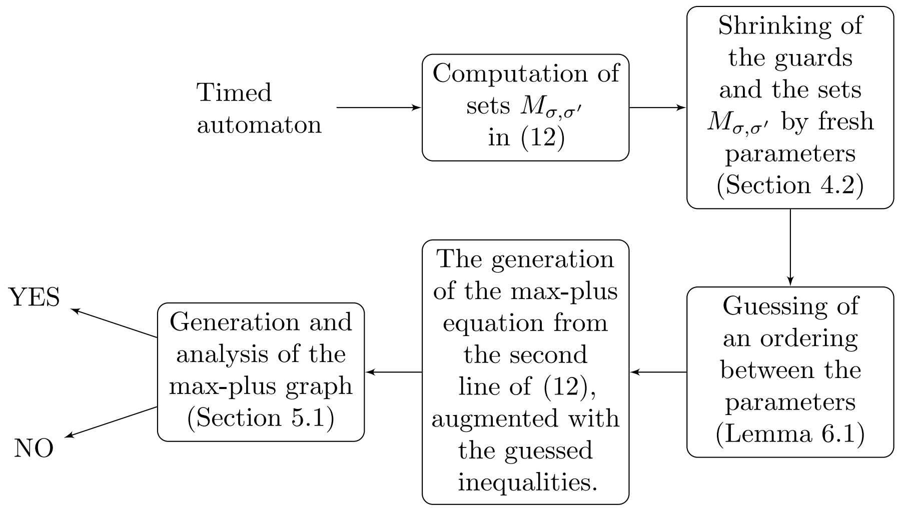
Formally, let Φ be the max-plus equation corresponding to the second part of (13), as defined in Section 4.2. Let $\mathbf{k}'$ denote the set of all parameters that appear in Φ (there is one parameter per element of each matrix G_σ and $M_{\sigma,\sigma'}$). Notice that $\mathbf{k}'$ has size $O\big((|\mathcal{C}_0| \cdot |\mathcal{L}| \cdot b)^2\big)$, 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 5.1. If we find a solution, say $(P_\sigma)_\sigma \cup (P_{\sigma,\sigma'})_{\sigma,\sigma'}$, we verify that $\llbracket G_\sigma - \delta P_\sigma \rrbracket = \bigcup_{\sigma'} \llbracket M_{\sigma,\sigma'} - \delta P_{\sigma,\sigma'} \rrbracket$ 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 6.1. We accept if all verifications succeed and reject otherwise. If accepted, any solution provides a shrunk solution of (13), by Lemma 4.13. Conversely, if there is a shrunk solution of (13), then, Φ' can be constructed for the guesses corresponding to this solution, and by Lemma 6.1, 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. A summary of the several steps of the algorithm is given in Fig. 9.

Finally, to decide strong shrinkability, one can first compute the least parameters $\mathbf{k}$ and δ_0 for non-blocking-shrinking, then check simulation-shrinkability since the latter does not depend on exact values of $\mathbf{k}$ and δ_0.

6.3. In practice

We implemented the algorithm for the shrinkability with respect to simulation in a tool called `Shrinktech`. The program is written in C++ and extends the Uppaal DBM library [8] to shrunk DBMs and uses Kronos [33] to compute a bisimulation quotient of a given timed automaton.

[8]http://people.cs.aau.dk/ adavid/UDBM/



Timed automaton
Computation of sets $M_{\sigma,\sigma'}$ in (12)
Shrinking of the guards and the sets $M_{\sigma,\sigma'}$ by fresh parameters (Section 4.2)
Guessing of an ordering between the parameters (Lemma 6.1)
The generation of the max-plus equation from the second line of (12), augmented with the guessed inequalities.
Generation and analysis of the max-plus graph (Section 5.1)
YES
NO

Figure 9: Summary of the non-blocking-shrinkability algorithm. If shrinkability holds (YES), then the analysis of the max-plus graph yields the least solution, thus, the least shrinking of the guards that yield a non-blocking timed automaton.

Let us return to Example 3.4. We have a timed automaton $\mathcal{A}$ describing alternating events with adjustable periods with constraints. Remember that the system is not simulation-shrinkable because an infinite run intuitively requires the periods of both events to be exactly 2. To detect this, we define the finite automaton $\mathcal{F}$ as the simple (untimed) cycle of $\mathcal{A}$. When run on $\mathcal{A}$ and $\mathcal{F}$ the tool `Shrinktech` answers that $\mathcal{A}$ is not shrinkable w.r.t. $\mathcal{F}$. In fact, the tool finds a bad cycle (see Section 5), in the generated max-plus graph, which can then be traced back to the cycle $\texttt{cons} \cdot \texttt{prod}$. This means that there is no possible value for the shrinking parameters along this cycle under which the timed automaton simulates the finite automaton.

The system can be fixed by allowing the productions to slow down further when the consumptions also slow down. In fact, if we define timed automaton $\mathcal{A}'$ by replacing the guard $x \leq 2$, by $x \leq 3$, then the tool answers that $\mathcal{A}'$ is shrinkable w.r.t. $\mathcal{F}$ with $k_1 = k_2 = 1$ and $\delta = 0.1$.

In order to evaluate the performance of our tool, we applied it to several benchmarks found in the literature. Some of these results are summarized in Table 1. In all models, all atomic guards were shrunk by positive-valued parameters except for equality constraints. This means that the shrinking parameters of the guards that contain equality were not required to be positive (see Remark 4.12).

The circuit models were studied with Imitator, a parameter synthesis tool for timed automata[9]. The guards are used to encode timing constraints on the input and output of the gates. The results show that the chosen constants are robust in the sense that all time-abstract behaviors are preserved when the guards are shrunk. The phone protocol, and the Philips Audio protocol are case studies done with Kronos. The former contains several equality constraints, and even though the model contains non-equality guards, mixing these turn out to imply punctual simulator sets which are no longer valid when the (non-equality) guards are shrunk. We believe that shrinkability analysis is

[9]`http://www.lsv.ens-cachan.fr/Software/imitator/`

Table 1: Experimental results. Shrinkability results marked with an asterisk indicates that the finite automaton $\mathcal{F}$ was not the full bisimulation graph.

Model	**states**	**trans**	**clocks**	$\lvert\mathcal{F}\rvert$	**time**	**result**
$\mathcal{A}$	2	2	2	2/2	0.1s	No
$\mathcal{A}'$	2	2	2	2/2	0.1s	Yes
Phone Prot.	11	16	5	63/271	0.1s	No
Philips Audio	446	2097	2	437/2734	46s	Yes
Flip-Flop Circuit	22	34	5	30/64	1s	Yes
Latch Circuit	32	77	7	105/364	1.6s	Yes
Fischer's Protocol 3	152	464	3	472/4321	20s	Yes
Fischer's Protocol 4	752	2864	4	4382/65821	310min	Yes
Fischer's Protocol 4	752	2864	4	4/4*	0.1s	Yes
Train Gate Controller	320	1136	4	36135/1099414	MEMOUT	
Train Gate Controller	320	1136	4	4/4*	0.1s	Yes

more interesting for models without equality constraints.

In the train gate controller example from [34], and the model describes a gate with constraints on opening and closing times, trains with time bounds on arrival rate, and a controller. The full bisimilarity quotient is too large to process, but the design is shrinkable w.r.t. a simple cycle requiring the liveness of all trains. A possible interpretation of the shrinkability is that the train gate controller does not require trains to go through the gate at the last moment, or immediately after opening.

Fischer's protocol shows the limit of the approach when the finite automaton is the full bisimilarity quotient. Nevertheless, the verification time is significantly smaller when the finite automaton is a simple cycle requiring all agents to make progress. We believe that significant properties can be expressed with small automata, and one rarely needs the full bisimilarity quotient.

The tool is open source and available at `http://www.lsv.ens-cachan.fr/Software/shrinktech`.

7. Application to Implementability

In this section, we demonstrate how shrinkability can be used to ensure that the behaviour is preserved in implementation. To this end, we first present an *implementation semantics* for timed automata, which takes into account nonzero reaction times, synchronization delays and clock imprecisions. Given a shrinkable timed automaton, we show how the parameters of the implementation semantics can be chosen so that the semantics is preserved. Our semantics corresponds to the execution of timed automata by a digital system that has a single digital clock and nonzero reaction time. It is similar to the one studied in [10] with minor differences (corresponding to different abstraction choices), and we prove additional properties besides the one given there.

Implementability is in general a difficult problem, and an exact answer needs to take into account a detailed model of the platform, such as the worst and best case execution times for each instruction in a given microprocessor (*e.g.* [35]). Such a modeling is out of the scope of this paper. It is nonetheless useful to carry the formal verification as far as possible in the design of a system, so as to gain confidence in the design at hand. Our goal here is to show that some properties of the system (simulation and non-blockingness) are preserved in a semantics that is closer to a real implementation than the idealized abstract semantics of timed automata.

We first define our semantics and state its properties, then compare it with [10], 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. Thus, in our semantics, the system does not have a buffer to store incoming signals; it either responds immediately to a signal or ignores it. 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 $\lfloor T \rfloor_{\Delta_c} = \max_{k\in\mathbb{N}}\{k\Delta_c \mid k\Delta_c \le T\}$.

Definition 7.1. *Let $\mathcal{A} = (\mathcal{L}, \ell_0, \mathcal{C}, \Sigma, E)$ be a timed automaton 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 global time*[10]*, $v \in \mathbb{R}^{\mathcal{C}}_{\ge 0}$ 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 $T' \ge T$, we let,*

- *if $\sigma \in \Sigma_{\mathsf{in}}$, $(\ell, T, v, u_0) \xrightarrow{\sigma(T')} (\ell', T' + \epsilon, v[R \leftarrow T' + \epsilon], u'_0)$, whenever $\lfloor T' + \epsilon \rfloor_{\Delta_c} - \lfloor v \rfloor_{\Delta_c} \models g$ and $T' + \epsilon \ge T + u_0$, 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 $T' - \epsilon \ge T + u_0$, where $(\epsilon, u'_0) \in [0, \Delta_t] \times [0, \Delta_r]$ is chosen non-deterministically.*

The transitions should be interpreted as follows. If the environment generates an event at time T', then the system responds to it at time $T' + \epsilon$, provided that the reaction time from the previous event is over ($T' + \epsilon \ge T + u_0$), and the guard is approximately satisfied. If the system generates an event at time $T' - \epsilon$, similar constraints apply but the timestamp is registered as T', which is the time the environment receives the event. 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 $[\![\mathcal{A}]\!]^{\mathsf{Impl}}_{\rho}$ 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

[10]Although we define the semantics with respect to an exact global time, the behaviour of this TTS will only depend on an approximate measure of this time.

write $\mathcal{T}^{\geq\alpha}$, the TTS obtained from $\mathcal{T}$ where consecutive transitions are restricted to be separated by at least α time units. More precisely, the states of $\mathcal{T}^{\geq\alpha}$ are pairs (s,u) where s is a state of $\mathcal{T}$, and u is the time elapsed since the latest transition, $(s_0,0)$ being the initial state if s_0 is that of $\mathcal{T}$. There is a transition $(s,u)\xrightarrow{\sigma(T)}(s',0)$ if, and only if $s\xrightarrow{\sigma(T)}s'$ in $\mathcal{T}$ and $u\geq\alpha$. For general TTS, this can be seen as a semantic modification. For timed automata, this property can be easily obtained by adding a new clock u reset at each transition and guards $u\geq\alpha$ to every transition.

Proposition 7.2. *Let $\mathcal{A}$ be a closed non-blocking timed automaton, 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,*

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

Before proving this proposition, let us give our main result on implementability. As in Section 3.3, for any timed automaton $\mathcal{A}$, we denote by $\mathcal{A}'$ the timed automaton obtained from $\mathcal{A}$ by adding a new clock u reset at each edge. We have the following result.

Theorem 7.3. *Let $\mathcal{A}$ be a timed automaton that is strongly shrinkable w.r.t. the region automaton of $\mathcal{A}$ and let $\mathcal{A}_{-\mathbf{k}\delta}$ be its witnessing shrinking for $\delta\in[0,\delta_0]$. Let $\Delta_r,\Delta_c,\Delta_t>0$ be parameters such that $4\Delta_c+8\Delta_t+2\Delta_r\leq\delta_0$. Then for all $\Delta\in[2\Delta_c+4\Delta_t+\Delta_r,\delta_0-2\Delta_c-4\Delta_t]$, for any scheduler ρ, $[\![\mathcal{A}_\Delta]\!]_\rho^{\mathsf{Impl}}$ is a non-blocking timed refinement of $\mathcal{A}$ and time-abstract simulates $\mathcal{A}$.*

Proof. By Proposition 7.2 applied to $\mathcal{A}'_{-\mathbf{k}\delta}$, we have

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

and $[\![\mathcal{A}'_{-\mathbf{k}\delta+\Delta}]\!]_\rho^{\mathsf{Impl}}$ is non-blocking 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 $\delta\geq 2\Delta_r+\Delta_t$ due to the shrinking of the additional clock constraints in $\mathcal{A}'$. The rightmost simulation is due to the fact that $-\mathbf{k}\delta+\Delta'\leq 0$. □

Thus, given δ_0, the parameters $\Delta_c,\Delta_t,\Delta_r$ and Δ can be chosen so that the implementation semantics of the automaton $\mathcal{A}'_{-\mathbf{k}\delta+\Delta}$ is a timed 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, shrinkable timed automata can be implemented so as to preserve non-blockingness and the behaviour upto time-abstract simulation.

We prove Proposition 7.2 through Lemmas 7.4 – 7.8. In the proofs we use the standard supremum distance on $\mathbb{R}^n$ defined by $d_\infty(\nu,\nu')=\max_{1\leq i\leq n}(|\nu_i-\nu'_i|)$, where $\nu,\nu'\in\mathbb{R}^n$. For any vector ν and real α, we denote by $\nu+\alpha$ the vector obtained by adding α to all components of ν.

Lemma 7.4. *Let $\mathcal{A}$ be a timed automaton, $\Delta_r,\Delta_c,\Delta_t>0$ denote the parameters. Then, for any $\Delta\geq 2\Delta_c+4\Delta_t$, and any scheduler ρ,*

$$[\![\mathcal{A}]\!]_\rho^{\mathsf{Impl}}\sqsubseteq[\![\mathcal{A}_\Delta]\!].$$

Proof. We show that the relation $\mathcal{R}$ defined by $(\ell,T,v,u_0)\mathcal{R}(\ell,\nu,T')$ such that $T'\in[T-\Delta_t,T]$, and $d_\infty(\nu+T-T',T-v)\leq\Delta_t$ is a timed simulation. Notice that we do not require these two states to be at the same time instant, but the difference between these instants must be bounded and the clock valuations must be close when the second system delays to time T. Consider such a pair of states.

Consider the following transition.

$$(\ell, T, v, u_0) \xrightarrow{\sigma(T+\tau)} (\ell', T+\tau+\epsilon, v[R \leftarrow T+\tau+\epsilon], u_0'),$$

for some $\sigma \in \Sigma_{\mathsf{in}}$, and $(u_0', \epsilon) \in [0, \Delta_r] \times [0, \Delta_t]$ is given by ρ, and $\lfloor T+\tau+\epsilon \rfloor_{\Delta_c} - \lfloor v \rfloor_{\Delta_c} \models g$. We show that the following transitions are realizable in $[\![\mathcal{A}_\Delta]\!]$.

$$(\ell, \nu, T') \xrightarrow{\sigma(T+\tau)} (\ell', (\nu+T+\tau-T')[R \leftarrow 0], T+\tau).$$

We need to show that $\nu + T + \tau - T' \models \langle g \rangle_{2\Delta_c+4\Delta_t}$. Using the fact that $\lfloor \alpha \rfloor_{\Delta_c} \in [\alpha - \Delta_c, \alpha]$ for any $\alpha \in \mathbb{R}_{\geq 0}$, we get $d_\infty\big((T+\tau+\epsilon-v), (\lfloor T+\tau+\epsilon \rfloor_{\Delta_c} - \lfloor v \rfloor_{\Delta_c})\big) \leq \Delta_c$. Then $T+\tau+\epsilon-v \models \langle g \rangle_{2\Delta_c}$ (the factor 2 is due to diagonal constraints), and, $T+\tau-v \models \langle g \rangle_{2\Delta_c+2\Delta_t}$. Since $d_\infty(\nu+T-T', T-v) \leq \Delta_t$, we also have $d_\infty(\nu+T-T'+\tau, T-v+\tau) \leq \Delta_t$, so $\nu+T+\tau-T' \models \langle g \rangle_{2\Delta_c+4\Delta_t}$. We now show that the new states are related by $\mathcal{R}$. Let $\nu' = (\nu+T+\tau-T')[R \leftarrow 0]$ and $v' = v[R \leftarrow T+\tau+\epsilon]$. We have $d_\infty(\nu'+\epsilon, T+\tau+\epsilon-v') \leq \Delta_t$. In fact, for any clock $x \notin R$, this follows from the assumption that $d_\infty(\nu+T-T', T-v) \leq \Delta_t$, and for all $x \in R$, we have $\nu'(x)+\epsilon = \epsilon \leq \Delta_t$ and $(T+\tau+\epsilon-v')(x) = 0$.

Now, consider the following transition.

$$(\ell, T, v, u_0) \xrightarrow{\sigma(T+\tau)} (\ell', T+\tau, v[R \leftarrow T+\tau-\epsilon], \epsilon, u_0'),$$

where $\sigma \in \Sigma_{\mathsf{out}}$, $(u_0', \epsilon) \in [0, \Delta_r] \times [0, \Delta_t]$ is given by ρ and $\lfloor T+\tau-\epsilon \rfloor_{\Delta_c} - \lfloor v \rfloor_{\Delta_c} \models g$. We show that the following transition is realizable in $[\![\mathcal{A}_\Delta]\!]$.

$$(\ell, \nu, T') \xrightarrow{\sigma(T+\tau)} (\ell', (\nu+T+\tau-T')[R \leftarrow 0], T+\tau).$$

We show that $\nu+T+\tau-T' \models \langle g \rangle_{2\Delta_c+4\Delta_t}$. As in the previous case, we have, $T+\tau-\epsilon-v \models \langle g \rangle_{2\Delta_c}$ and using the fact that $d_\infty(\nu+T-T', T-v) \leq \Delta_t$, we get $\nu+T+\tau-T' \models \langle g \rangle_{2\Delta_c+4\Delta_t}$. Let $\nu' = (\nu+T+\tau-T')[R \leftarrow 0]$ and $v' = v[R \leftarrow T+\tau-\epsilon]$. We have $d_\infty(\nu', T+\tau-v') \leq \Delta_t$. In fact, for any clock $x \notin R$, this follows from $d_\infty(\nu+T-T', T-v) \leq \Delta_t$, and for all $x \in R$, we have $\nu'(x) = 0$ and $(T+\tau-v')(x) = \epsilon \leq \Delta_t$. □

For any timed automaton $\mathcal{A}$ and $\Delta' \geq 0$, we denote the states of $[\![\mathcal{A}]\!]^{\geq \Delta'}$ as triples (ℓ, ν, u), where ℓ is a location, ν a clock valuation and u the time elapsed since the latest action (and it is 0 initially).

Lemma 7.5. *Let $\mathcal{A}$ be a timed automaton, and $\Delta_r, \Delta_c, \Delta_t > 0$ parameters. Then, for any $\Delta \geq 2\Delta_c + 4\Delta_t$, and any scheduler ρ,*

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

Proof. We show that the relation $\mathcal{R}$ defined by $(\ell, \nu, T)\mathcal{R}(\ell, T', v, u_0)$ such that $T' \in [T, T+\Delta_t]$ and $d_\infty(\nu + (T'-T), T'-v) \leq \Delta_t$ is a timed simulation. Consider such a pair of states.

Suppose that $(\ell, \nu, T) \xrightarrow{\sigma(T+\tau)} (\ell', \nu', T+\tau)$ for some $\sigma \in \Sigma_{\mathsf{in}}$, where $\nu' = (\nu+\tau)[R \leftarrow 0]$. For any $(u_0, \epsilon) \in [0, \Delta_r] \times [0, \Delta_t]$ given by ρ, this is simulated by the following transition.

$$(\ell, T', v, u_0) \xrightarrow{\sigma(T+\tau)} (\ell', T+\tau+\epsilon, v[R \leftarrow T+\tau+\epsilon], u_0').$$

In fact, by hypothesis, $\tau \geq 2\Delta_r + \Delta_t$, so $(T+\tau) - T' \geq \Delta_r \geq u_0$, hence the reaction time is over when the action happens. Let us show that the guard is satisfied. We have $\nu + \tau \models g$. Since $d_\infty(\nu + T' - T, T' - v) \leq \Delta_t$, we have $d_\infty(\nu+\tau, T+\tau-v) \leq \Delta_t$ (in fact, we add $\tau - (T'-T) = \tau - T' + T$ to the first vector, and $(T+\tau) - T' = \tau - T' + T$ to the second). Hence $T + \tau + \epsilon - v \models \langle g \rangle_{4\Delta_t}$. Then, $\lfloor T + \tau + \epsilon \rfloor_{\Delta_c} - \lfloor v \rfloor_{\Delta_c} \models \langle g \rangle_{4\Delta_t + 2\Delta_c}$, so the guard is satisfied. Let us write $v' = v[R \leftarrow T + \tau + \epsilon]$. It remans to show that $d_\infty(\nu' + \epsilon, T + \tau + \epsilon - v') \leq \Delta_t$. In fact, for all $x \notin R$ this follows from hypothesis since the difference between the values of a clock in the two systems is unchanged when both systems delay to time instant $T + \tau + \epsilon$. For all $x \in R$, we have $(\nu' + \epsilon)(x) = \epsilon \leq \Delta_t$ and $(T + \tau + \epsilon - v')(x) = 0$. Hence, $(\ell', \nu', T+\tau)\mathcal{R}(\ell', T+\tau+\epsilon, v', u'_0)$.

Suppose now that $(\ell, \nu, T) \xrightarrow{\sigma(T+\tau)} (\ell', \nu', T+\tau)$ for some $\sigma \in \Sigma_{\mathsf{out}}$, where $\nu' = (\nu+\tau)[R \leftarrow 0]$. For any $(u'_0, \epsilon) \in [0, \Delta_r] \times [0, \Delta_t]$ given by ρ, this is simulated by the following.

$$(\ell, T', v, u_0) \xrightarrow{\sigma(T+\tau)} (\ell', T+\tau, v[R \leftarrow T + \tau - \epsilon], u'_0),$$

In fact, by hypothesis, $\tau \geq 2\Delta_r + \Delta_t$, so $T + \tau - \epsilon - T' \geq \Delta_r$. Let us show that the guard is satisfied. We have $\nu + \tau \models g$. Since $d_\infty(\nu + (T'-T), T' - v) \leq \Delta_t$, we have $d_\infty(\nu + \tau, T + \tau - v) \leq \Delta_t$ as in the previous case, hence $T + \tau - \epsilon - v \models \langle g \rangle_{4\Delta_t}$. Then, $\lfloor T + \tau - \epsilon \rfloor_{\Delta_c} - \lfloor v \rfloor_{\Delta_c} \models \langle g \rangle_{4\Delta_t + 2\Delta_c}$. It remans to show that $d_\infty(\nu', T + \tau - v') \leq \Delta_t$. This follows from hypothesis for clocks $x \notin R$ since both systems delay to time instant $T + \tau$. For all $x \in R$, we have $(\nu')(x) = 0$ and $(T + \tau - v')(x) = \epsilon \leq \Delta_t$. Hence, $(\ell', \nu', T+\tau)\mathcal{R}(\ell', T+\tau, v', u'_0)$. □

We are now interested in the preservation of non-blockingness in the implementation semantics. Note that this is not a consequence of the simulation relations. We first prove a property on the enlarged zones.

Definition 7.6. *Let Z be any closed convex subset of $\mathbb{R}^{\mathcal{C}}_{\geq 0}$. The* lower boundary *of Z is the set $lb(Z) = \{v \in Z \mid \forall \tau > 0, v - \tau \notin Z\}$. The* width *of Z is defined as* $\inf\{\tau \mid \exists v \in lb(Z), v + \tau \notin Z)$. *In other terms, the width of Z is the least delay necessary to go out of Z starting inside the lower boundary.*

A guard is said to be normalized if the corresponding DBM is normalized.

Lemma 7.7. *Let g be a normalized guard such that $[\![g]\!] \neq \emptyset$. Then for any $\Delta > 0$, $[\![\langle g \rangle_\Delta]\!]$ has width greater than or equal to Δ.*

Proof. Let M be a DBM that describes $[\![g]\!]$. Then $M' = M + \Delta\mathbf{1}$ describes $[\![\langle g \rangle_\Delta]\!]$, where $\mathbf{1}$ is the matrix with same dimension as M in which all off-diagonal entries are 1's and diagonal entries are 0. Let $v \in lb([\![M']\!])$. We will show that $v + \tau \in [\![M']\!]$ for all $\tau \in [0, \Delta]$. First, observe that clock differences are constant during delay transitions. So, whenever $-M_{y,x} - \Delta \leq v(x) - v(y) \leq M_{x,y} + \Delta$, we have $-M_{y,x} - \Delta \leq (v(x) + \tau) - (v(y) + \tau) \leq M_{x,y} + \Delta$ for all τ. We now show that rectangular constraints are also satisfied for at least Δ time units. Since $v \in lb([\![M']\!])$, there exists $x \in \mathcal{C}$, such that $-M_{0,x} - \Delta = v(x)$ (otherwise valuation v can be decremented by some positive amount). For this clock, obviously $-M_{0,x} - \Delta \leq v(x) + \tau \leq M_{x,0} + \Delta$ for $\tau \in [0, \Delta]$, since $-M_{0,x} \leq M_{x,0}$ (in fact, the set is not empty). Now, consider any $y \in \mathcal{C}$. DBM M' implies the following diagonal constraint (and possibly a tighter one).

$$-\Delta - M_{0,x} - M_{y,0} \leq v(x) - v(y) \leq M_{x,0} + M_{0,y} + \Delta.$$

But, combining the above inequality with $v(x) = -M_{0,x} - \Delta$, we get that $v(y) \leq M_{y,0}$, so $v(y) + \tau \leq M'_{y,0} = M_{y,0} + \Delta$ for $\tau \in [0, \Delta]$. □

The following lemma shows that if the exact semantics is non-blocking, then the implementation semantics is also non-blocking.

Lemma 7.8. *Let $\mathcal{A}$ be a timed automaton, $\Delta_r, \Delta_c, \Delta_t > 0$ be parameters and assume that $[\![\mathcal{A}]\!]$ is non-blocking. Then for any $\Delta \geq \Delta_r + 2\Delta_c + \Delta_t$, $[\![\mathcal{A}_\Delta]\!]^{\mathsf{Impl}}$ is non-blocking.*

Proof. First, observe that if $[\![\mathcal{A}]\!]$ is non-blocking, then so is $[\![\mathcal{A}_\Delta]\!]$ for any $\Delta > 0$. In fact, consider an edge $\ell \xrightarrow{g,\sigma,R} \ell'$ and any corresponding transition $(\ell, v_1, T) \xrightarrow{\sigma(T+\tau)} (\ell', v_1', T+\tau)$ in $[\![\mathcal{A}_\Delta]\!]$. Since $v_1 + \tau \models_\Delta g$, there exists v_2 such that $d_\infty(v_2, v_1) \leq \Delta$ and $v_2 + \tau \models g$. Let v_2' be such that $(\ell, v_2, T) \xrightarrow{\sigma(T+\tau)} (\ell', v_2', T+\tau)$. Since $[\![\mathcal{A}]\!]$ is non-blocking, for some $\tau \geq 0$, there exists an edge with guard g' such that $v_2' + \tau \models g'$. But $d_\infty(v_1', v_2') \leq \Delta$, so we also have $v_1' + \tau \models_\Delta g'$, and the edge is enabled from $(\ell', v_1', T+\tau)$ as well. Hence $[\![\mathcal{A}_\Delta]\!]$ is non-blocking.

Let $(\ell, T, v, u_0) \xrightarrow{\sigma(T+\tau)} (\ell', T+\tau+\epsilon, v', u_0')$ denote a transition in $[\![\mathcal{A}_\Delta]\!]^{\mathsf{Impl}}$ with $v' = v[R \leftarrow T+\tau+\epsilon]$ for some $\sigma \in \Sigma_{\mathsf{in}}$, and $\ell \xrightarrow{\sigma,g,R} \ell'$ the corresponding edge in $\mathcal{A}$. We have $\lfloor T+\tau+\epsilon \rfloor_{\Delta_c} - \lfloor v \rfloor_{\Delta_c} \models_\Delta g$, so for $\nu = \lfloor T+\tau+\epsilon \rfloor_{\Delta_c} - \lfloor v \rfloor_{\Delta_c}$, we have $(\ell, \nu, 0) \xrightarrow{\sigma(0)} (\ell', \nu', 0)$ in $[\![\mathcal{A}_\Delta]\!]$, where $\nu' = \nu[R \leftarrow 0]$. Notice that $\nu' = \lfloor T+\tau+\epsilon \rfloor_{\Delta_c} - \lfloor v' \rfloor_{\Delta_c}$. Since $[\![\mathcal{A}_\Delta]\!]$ is non-blocking, there exists $\tau \geq 0$ such that $\nu' + \tau \models_\Delta g'$, where g' is the guard of some edge with label σ' outgoing from ℓ'.

Consider $Z = [\![\langle g' \rangle_\Delta]\!]$, whose width is at least Δ by Lemma 7.7. Then, there exists $\alpha, \beta \in \mathbb{R}_{\geq 0}$ such that $\alpha + \Delta \leq \beta$ and $\nu' + \tau' \in [\![\langle g' \rangle_\Delta]\!]$ for all $\tau' \in [\alpha, \beta]$. We show that $(\ell', T+\tau+\epsilon, v', u_0')$ can delay some amount τ' and take this transition. For any $\max(\alpha, \Delta_r) + \Delta_c \leq \tau' \leq \beta - \Delta_c - \Delta_t$, we have

$$\lfloor T+\tau+\epsilon \rfloor_{\Delta_c} + (\max(\alpha, \Delta_r) + \Delta_c) - \Delta_c \leq \lfloor T+\tau+\epsilon+\tau' \rfloor_{\Delta_c} \leq \lfloor T+\tau+\epsilon \rfloor_{\Delta_c} + (\beta - \Delta_c - \Delta_t) + \Delta_c.$$

Thus,

$$\nu' + \max(\alpha, \Delta_r) \leq \lfloor T+\tau+\epsilon+\tau' \rfloor_{\Delta_c} - \lfloor v' \rfloor_{\Delta_c} \leq \nu' + \beta - \Delta_t. \tag{14}$$

Notice that such a τ' exists since $\Delta_r + 2\Delta_c + \Delta_t \leq \Delta$. Hence, if $\sigma' \in \Sigma_{\mathsf{out}}$, then, for any $\epsilon' \in [0, \Delta_t]$ given by ρ, the transition

$$(\ell', T+\tau+\epsilon, v', u_0') \xrightarrow{\sigma'(T+\tau+\epsilon+\tau'+\epsilon')} (\ell'', T+\tau+\epsilon+\tau'+\epsilon', \cdot, \cdot),$$

is valid. If $\sigma' \in \Sigma_{\mathsf{in}}$, then

$$(\ell', T+\tau+\epsilon, v', u_0') \xrightarrow{\sigma'(T+\tau+\epsilon+\tau')} (\ell'', T+\tau+\epsilon+\tau'+\epsilon', \cdot, \cdot)$$

is valid thanks to the right hand side term in (14) (since $\epsilon' \in [0, \Delta_t]$.

The proof is similar when $\sigma \in \Sigma_{\mathsf{out}}$. □

7.1. Related Work on Implementation Semantics

A similar semantics, called the *program semantics*, was defined in [10] and was proven to be simulated by the enlarged semantics (as in the rightmost simulation in Proposition 7.2). Our definition follows their ideas, but we define a somewhat more abstract semantics by concentrating on different aspects. The main difference is that we do not insist on the semantics to produce events *almost as soon as possible (almost ASAP).* Thus, transitions are not urgent in our semantics; only the reactions to input events are. Also, our semantics is not *input-enabled*, that is, it can ignore signals during the treatment of another signal, since it has no buffer. Both assumptions are applicable to different platforms (see [36, 37] for examples of systems that ignore any signal unless it is maintained long enough). Moreover, instead of giving a detailed model of 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*. Due to these differences, both semantics are uncomparable: the program semantics is not simulated, in general, by our implementation semantics, and vice versa. Note that two results in Proposition 7.2 are new compared to [10]: the leftmost simulation and the preservation of non-blockingness.

A similar effort to define an implementation semantics for *programmable logic controllers* as *PLC automata*, was studied in [37]. However, one cannot use the full power of timed automata in that framework since PLC automata correspond to a restricted class of timed automata. A different line of work considers the implementability of timed automata extended with tasks under sampling of time [38] but no detailed semantics is studied (see also [39]). Another recent work considers the variation in execution times in systems modeled as timed automata, when actions have long execution times [5].

8. Conclusion

In this work, we introduced the technique of shrinking the guards of timed automata, in order to obtain correct implementations. We showed that shrinking the guards always yields implementations that are refinements of the abstract models, and we gave algorithms to synthesize the shrinking parameters for all atomic guards to ensure that the shrunk automaton is non-blocking, and can time-abstract simulate (some part of) the initial automaton. Our framework allows one to design a system with the usual timed automata semantics, use existing tools to run simulations and do verification, and then synthesize the shrinking parameters before implementing the system. If the automaton is shrinkable, then several properties proved for the initial automaton are preserved in the implementation. If it is not shrinkable, then this suggests that the model is vulnerable to the slightest variations in the measure of time and should thus be considered as non-robust, and the design should be reviewed.

Future work includes developing techniques to solve a larger class of fixpoint equations on shrunk DBMs, such as those including arbitrary use of the union operation. One could then solve, for instance, the simulation-shrinkability problem without the hypothesis of distinct edge labels. We believe the shrunk DBMs we introduced can be useful in solving other problems in timed automata, involving small imprecisions. We recently used this data structure to solve a different kind of robustness problem on timed automata, where the control is modelled as a game, and the imprecisions are controlled by one of the players [16, 18]. We plan to investigate such a game semantics for richer objectives and under probabilistic perturbations.

[1] R. Alur, D. L. Dill, A theory of timed automata, Theoretical Computer Science 126 (2) (1994) 183–235.

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

[3] M. De Wulf, L. Doyen, N. Markey, J.-F. Raskin, Robust safety of timed automata, Formal Methods in System Design 33 (1-3) (2008) 45–84.

[4] F. Cassez, T. A. Henzinger, J.-F. Raskin, A comparison of control problems for timed and hybrid systems, in: C. Tomlin, M. R. Greenstreet (Eds.), Proceedings of the 5th International Workshop on Hybrid Systems: Computation and Control (HSCC'02), Vol. 2289 of Lecture Notes in Computer Science, Springer, 2002, pp. 134–148.

[5] T. Abdellatif, J. Combaz, J. Sifakis, Model-based implementation of real-time applications, in: Proceedings of the tenth ACM international conference on Embedded software, ACM, New York, NY, USA, 2010, pp. 229–238.

[6] P. Bouyer, N. Markey, P.-A. Reynier, Robust model-checking of linear-time properties in timed automata, in: J. R. Correa, A. Hevia, M. Kiwi (Eds.), Proceedings of the 7th Latin American Symposium on Theoretical INformatics (LATIN'06), Vol. 3887 of Lecture Notes in Computer Science, Springer, 2006, pp. 238–249.

[7] P. Bouyer, N. Markey, P.-A. Reynier, Robust analysis of timed automata via channel machines, in: R. Amadio (Ed.), Proceedings of the 11th International Conference on Foundations of Software Science and Computation Structure (FoSSaCS'08), Vol. 4962 of Lecture Notes in Computer Science, Springer, 2008, pp. 157–171.

[8] P. Bouyer, N. Markey, O. Sankur, Robust model-checking of timed automata via pumping in channel machines, in: U. Fahrenberg, S. Tripakis (Eds.), Proceedings of the 9th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'11), Vol. 6919 of Lecture Notes in Computer Science, Springer, Aalborg, Denmark, 2011, pp. 97–112. `doi:10.1007/978-3-642-24310-3_8`.

[9] O. Sankur, Untimed language preservation in timed systems, in: F. Murlak, P. Sankowski (Eds.), Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science (MFCS'11), Vol. 6907 of Lecture Notes in Computer Science, Springer, Warsaw, Poland, 2011, pp. 556–567. `doi:10.1007/978-3-642-22993-0_50`.

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

[11] D. L. Dill, Timing assumptions and verification of finite-state concurrent systems, in: J. Sifakis (Ed.), Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems (AVMFSS'89), Vol. 407 of Lecture Notes in Computer Science, Springer, 1990, pp. 197–212.

[12] M. Swaminathan, M. Fränzle, J.-P. Katoen, The surprising robustness of (closed) timed automata against clock-drift, in: G. Ausiello, J. Karhumäki, G. Mauri, L. Ong (Eds.), Fifth IFIP International Conference On Theoretical Computer Science TCS 2008, Vol. 273 of IFIP International Federation for Information Processing, Springer US, 2008, pp. 537–553. `doi:10.1007/978-0-387-09680-3_36`.

[13] C. Dima, Dynamical properties of timed automata revisited, in: J.-F. c. Raskin, P. Thiagarajan (Eds.), Formal Modeling and Analysis of Timed Systems, Vol. 4763 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2007, pp. 130–146.

[14] C. Daws, P. Kordy, Symbolic robustness analysis of timed automata, in: E. Asarin, P. Bouyer (Eds.), Proceedings of the 4th International Conferences on Formal Modelling and Analysis of

Timed Systems, (FORMATS'06), Vol. 4202 of Lecture Notes in Computer Science, Springer, 2006, pp. 143–155.

[15] R. Jaubert, P.-A. Reynier, Quantitative robustness analysis of flat timed automata, in: Proceedings of the 14th international conference on Foundations of software science and computational structures: part of the joint European conferences on theory and practice of software, Vol. 6604 of Lecture Notes in Computer Science, Springer, 2011, pp. 229–244.

[16] P. Bouyer, N. Markey, O. Sankur, Robust reachability in timed automata: A game-based approach, in: A. Czumaj, K. Mehlhorn, A. Pitts, R. Wattenhofer (Eds.), Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP'12) – Part II, Vol. 7392 of Lecture Notes in Computer Science, Springer, 2012, pp. 128–140.

[17] K. Chatterjee, T. A. Henzinger, V. S. Prabhu, Timed parity games: Complexity and robustness, Logical Methods in Computer Science 7 (4).

[18] O. Sankur, P. Bouyer, N. Markey, P.-A. Reynier, Robust controller synthesis in timed automata, in: CONCUR'13, Lecture Notes in Computer Science, Springer, 2013.

[19] P. Bouyer, K. G. Larsen, N. Markey, O. Sankur, C. Thrane, Timed automata can always be made implementable, in: J.-P. Katoen, B. König (Eds.), Proceedings of the 22nd International Conference on Concurrency Theory (CONCUR'11), Vol. 6901 of Lecture Notes in Computer Science, Springer, Aachen, Germany, 2011, pp. 76–91. `doi:10.1007/978-3-642-23217-6_6`.

[20] K. Altisen, S. Tripakis, Implementation of timed automata: An issue of semantics or modeling?, in: P. Pettersson, W. Yi (Eds.), Proceedings of the 3rd International Conferences on Formal Modelling and Analysis of Timed Systems, (FORMATS'05), Vol. 3829 of Lecture Notes in Computer Science, Springer, 2005, pp. 273–288.

[21] H. Bowman, G. Faconti, J.-P. Katoen, D. Latella, M. Massink, Automatic verification of a lip-synchronisation protocol using uppaal, Formal Aspects of Computing 10 (1998) 550–575.

[22] F. Heidarian, J. Schmaltz, F. W. Vaandrager, Analysis of a clock synchronization protocol for wireless sensor networks, Theor. Comput. Sci. 413 (1) (2012) 87–105.

[23] V. Gupta, Th. A. Henzinger, R. Jagadeesan, Robust timed automata, in: Proc. International Workshop on Hybrid and Real-Time Systems (HART'97), Vol. 1201 of Lecture Notes in Computer Science, Springer, 1997, pp. 331–345.

[24] N. Markey, Robustness in real-time systems, in: Proceedings of the 6th IEEE International Symposium on Industrial Embedded Systems (SIES'11), IEEE Computer Society Press, Västerås, Sweden, 2011, pp. 28–34.

[25] S. Tripakis, S. Yovine, Analysis of timed systems using time-abstracting bisimulations, Form. Methods Syst. Des. 18 (1) (2001) 25–68.

[26] J. Bengtsson, W. Yi, Timed automata: Semantics, algorithms and tools, in: J. Desel, W. Reisig, G. Rozenberg (Eds.), Lectures on Concurrency and Petri Nets, Vol. 2098 of Lecture Notes in Computer Science, Springer-Verlag, 2004, pp. 87–124.

[27] P. Chamuczyński, Algorithms and data structures for parametric analysis of real time systems, Ph.D. thesis, University of Göttingen, Germany (2009).

[28] T. Hune, J. Romijn, M. Stoelinga, F. Vaandrager, Linear parametric model checking of timed automata, in: T. Margaria, W. Yi (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, Vol. 2031 of Lecture Notes in Computer Science, Springer, 2001, pp. 189–203.

[29] F. Baccelli, G. Cohen, G. J. Olsder, J.-P. Quadrat, Synchronization and Linearity – An Algebra For Discrete Event Systems, John Wiley & Sons, 1992.

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

[31] J. Esparza, M. Luttenberger, Solving fixed-point equations by derivation tree analysis, in: Proceedings of the 4th international conference on Algebra and coalgebra in computer science, CALCO'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 19–35.

[32] A. David, J. Håkansson, K. G. Larsen, P. Pettersson, Model checking timed automata with priorities using DBM subtraction, in: E. Asarin, P. Bouyer (Eds.), Proceedings of the 4th International Conferences on Formal Modelling and Analysis of Timed Systems, (FORMATS'06), Vol. 4202 of Lecture Notes in Computer Science, Springer, 2006, pp. 128–142.

[33] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, S. Yovine, Kronos: A model-checking tool for real-time systems, in: CAV '98: Proceedings of the 10th International Conference on Computer Aided Verification, Springer-Verlag, London, UK, 1998, pp. 546–550.

[34] T. Aaltonen, M. Katara, R. Pitkänen, Verifying real-time joint action specifications using timed automata, in: 16th World Computer Congress, 2000, pp. 516–525.

[35] V. Bandur, W. Kahl, A. Wassyng, Microcontroller assembly synthesis from timed automaton task specifications, in: M. Stoelinga, R. Pinger (Eds.), Formal Methods for Industrial Critical Systems, Vol. 7437 of Lecture Notes in Computer Science, Springer, 2012, pp. 63–77.

[36] E. Asarin, O. Maler, A. Pnueli, On discretization of delays in timed automata and digital circuits, in: D. Sangiorgi, R. de Simone (Eds.), CONCUR'98 Concurrency Theory, Vol. 1466 of Lecture Notes in Computer Science, Springer, 1998, pp. 470–484. doi:10.1007/BFb0055642.

[37] H. Dierks, PLC-automata: a new class of implementable real-time automata, Theoretical Computer Science 253 (2001) 61–93. doi:10.1016/S0304-3975(00)00089-X.

[38] P. Krćal, L. Mokrushin, P. Thiagarajan, W. Yi, Timed vs. time-triggered automata, in: P. Gardner, N. Yoshida (Eds.), CONCUR 2004 - Concurrency Theory, Vol. 3170 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2004, pp. 340–354.

[39] T. A. Henzinger, B. Horowitz, C. M. Kirsch, Giotto: A time-triggered language for embedded programming, in: EMSOFT, Vol. 2211 of Lecture Notes in Computer Science, Springer, 2001, pp. 166–184.