Robust Model-Checking of Timed Automata via Pumping in Channel Machines

Patricia Bouyer, Nicolas Markey, and Ocan Sankur

LSV, CNRS & ENS Cachan, France

Abstract. Timed automata are governed by a mathematical semantics which assumes perfectly continuous and precise clocks. This requirement is not satisfied by digital hardware on which the models are implemented. In fact, it was shown that the presence of imprecisions, however small they may be, may yield extra behaviours. Therefore correctness proven on the formal model does not imply correctness of the real system.

The problem of robust model-checking was then defined to circumvent this inconsistency. It consists in computing a bound on the imprecision under which the system will be correct.

In this work, we show that robust model-checking against ω-regular properties for timed automata can be reduced to standard model-checking of timed automata, by computing an adequate bound on the imprecision. This yields a new algorithm for robust model-checking of ω-regular properties, which is both optimal and valid for general timed automata.

1 Introduction

Timed automata [1] are a well-established model in real-time system design. These are finite automata augmented with *clocks*, which are used to measure the time elapsed between events, and to constrain the runs of the automaton. Timed automata provide a powerful way of modelling and verifying real-time systems. However, timed automata make idealistic assumptions on the system, such as the perfect continuity of clocks and instantaneous reaction time, which are known not to be preserved in implementation even in digital hardware with arbitrarily small imprecisions. It was shown that even the smallest imprecisions on the clocks yield a different semantics than the *exact* one [14,8] (see Fig. 2 for an example). This suggests that even if the exact semantics is proven correct, the implementation on a physical machine is not guaranteed to respect the specification. In order to prove the correctness of implementations, a framework was proposed in [9], where a detailed model of the implementation of timed automata is given, as programs executed on a simple micro-processor. A simpler over-approximation, the so-called *enlarged semantics* was also studied, which models the imprecisions by *relaxing* all clock constraints of the automaton of the form $x \in [a, b]$ to $x \in [a - \delta, b + \delta]$ for some $\delta > 0$. The problem of *robust model-checking*, that is determining whether for *some* $\delta > 0$, the enlarged semantics satisfies a given property, was first solved for safety properties [14,8], then for linear temporal logic (LTL) [4] (both in PSPACE, which is the complexity of the problem in the exact semantics), and for a timed extension of LTL [5].

U. Fahrenberg and S. Tripakis (Eds.): FORMATS 2011, LNCS 6919, pp. 97–112, 2011.

These robust model-checking algorithms are all valid for a particular class of timed automata, namely, those in which all cycles are *progress cycles*. Roughly, a progress cycle is a cycle of the timed automaton which resets all clocks that are below the maximal constant at least once. We argue that this can be restrictive for modeling. In fact, a timed automaton model of a system under this assumption cannot measure the time spent in a cycle. As an example, consider a simple system which waits for a special signal, while ignoring any other signal, and triggers a time-out action if the expected signal is not received after one second (Fig. 1). In order to ignore any number of signals during this time, we need a cycle in the automaton. But if all clocks are reset on this cycle, then we cannot measure the time spent in it in order to issue the time-out. One could model such a system using progress cycles by explicitly defining an upper bound m on the number of events that can be treated by the system in one time unit, and unfolding the cycle for m iterations. This would remove the cycle. However this requires the prior knowledge of m which may not be obvious in the design phase, and moreover, this may increase the size of the model and render model-checking infeasible.

```
bool timeout := false;
clock x;
...
x := 0;
while ( x <= 1 ){
   ...
   if ( signal() == A )
      break;
}
if ( x >= 1 )
   timeout := true;
```

Fig. 1. A program that waits for a signal A and issues a time-out if it is not received in one time unit. A timed automaton model of this program naturally contains a non-progress cycle.

Our contribution. We propose a new algorithm for robust model-checking timed automata against ω-regular properties, with optimal complexity (PSPACE). Our algorithm consists in reducing the problem to classical model-checking of timed automata and is valid for general timed automata: we do not assume progress cycles, nor any upper bound on the clocks (Assuming bounded clocks is not restrictive in terms of expressivenes but has a negative effect on the size of the models [2]). We prove that any timed automaton satisfies a given ω-regular property under enlargement by *some* value $\delta > 0$ if, and only if, it satisfies the formula under enlargement by δ_0, where δ_0 only depends on the size of the timed automaton. Then the algorithm simply consists in model-checking the automaton enlarged by δ_0, which can be done using well-known algorithms and tools for timed automata. An algorithm was given in [4] for this problem but only for timed automata with progress cycles and bounded clocks, and because it is based on a modification of the region automaton construction, one cannot use directly the existing model-checking tools. For safety properties, an algorithm similar to ours can be derived from [8], but the complexity would be exponentially higher due to the bound given for δ_0. For automata without nested cycles, [12] gives an algorithm to compute the greatest δ under which a safety property holds, but does not provide a bound on δ.

Although the worst-case complexity of our algorithm is not higher than classical model-checking, in practice, the timed automaton enlarged by δ_0 can yield a model with a state space that is much larger than that of the initial automaton (see Section 3 for the precise value). However, we observed that this does not always increase the time and space necessary for verification. In fact, we used Uppaal [13] to test our algorithm on some benchmarks given with safety specifications.[1] We were able to show the non-robustness of the Fischer protocol upto three agents, and Uppaal returned almost immediately. With more than three agents, the problem is not due to time or space resources but to the fact that Uppaal only allows 32-bit integers as constants in timed automata; when δ_0 requires more precision, the model does not compile. However, Uppaal found counter-examples in less than one minute, for the protocol upto thirty agents, when enlarged by 10^{-8}. We believe that extending Uppaal with arbitrary precision integers, one should be able to use our algorithm for larger models. We could assess the robustness of the CSMA/CD protocol described in [15], the Bang & Olufsen Collision Detection Protocol [11], and the Token Ring Protocol upto thirty agents in less than one minute. Note that the correctness of a model under an enlargement δ implies the correctness for all $0 \leq \delta' < \delta$, so we only verified the above robust models under enlargement that we chose arbitrarily as $\delta = 10^{-6}$ (see [9]). All verification queries returned almost as fast as for the non-enlarged models.

In order to establish our results, we develop proof techniques based on the encoding of the states of timed automata with *channel machines*, introduced in [3], and used in [5] in the context of robustness. In this encoding, a word represents the content of a FIFO channel, which roughly contains all clock symbols ordered by their fractional parts. Time delays are simulated by sequences of read and writes on this channel, whereas action transitions also use a special renaming operation. It turns out that the finitary representation by these words capture well the behaviour of timed automata under enlargement. This was used in [5] to design a robust model-checking algorithm for a timed extension of LTL. We further develop these techniques and prove a pumping lemma for those channel machines, which preserves ω-regular properties. This enables us to prove new properties on the runs of enlarged timed automata, to refine some previously known results and obtain our algorithm. The proof follows the ideas of [14,8] but the techniques are different, and moreover, our analysis is finer since it yields an exponentially better bound for δ_0, as we also noted above.

By lack of space, technical proofs are not included. They can be found in [6].

2 Preliminaries

2.1 Timed Automata

A *labelled timed transition system (LTTS)* 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 $\rightarrow \subseteq S \times (\Sigma \cup \mathbb{R}_{\geq 0}) \times S$ the *transitions*.

[1] See `http://www.uppaal.org/benchmarks/`

Given a finite set of clocks $\mathcal{C}$, we call *valuations* the elements of $\mathbb{R}_{\geq 0}^{\mathcal{C}}$. For a subset $R \subseteq \mathcal{C}$, a real $\alpha \in \mathbb{R}_{\geq 0}$ 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. We write $\mathbf{0}$ for the valuation which assigns 0 to every clock.

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

$$\langle k \leq x \leq l \rangle_\delta \;=\; k - \delta \leq x \leq l + \delta.$$

The enlargement of a guard g, denoted by $\langle g \rangle_\delta$, is obtained by enlarging all its 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.

Definition 1. *A timed automaton $\mathcal{A}$ is a tuple $(\mathcal{L}, \mathcal{C}, \Sigma, l_0, E)$, consisting of 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, and where $l_0 \in \mathcal{L}$ is 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.*

A timed automaton is *integral* if all constants that appear in its guards are integers. For any $\delta \in \mathbb{Q}$, $\mathcal{A}_\delta$ denotes the timed automaton where all guards are enlarged by δ. In the sequel, we only consider integral timed automata as input, and only their enlarged counterparts might not be integral.

Definition 2. *The semantics of a timed automaton $\mathcal{A} = (\mathcal{L}, l_0, \mathcal{C}, \Sigma, E)$ is an LTTS over alphabet Σ, denoted $[\![\mathcal{A}]\!]$, whose state space is $\mathcal{L} \times \mathbb{R}_{\geq 0}^{\mathcal{C}}$. The initial state is $(l_0, \mathbf{0})$.* Delay transitions *are defined as $(l, v) \xrightarrow{\tau} (l, v + \tau)$ for any state (l, v) and $\tau \geq 0$.* Action transitions *are defined as $(l, v) \xrightarrow{\sigma} (l', v')$, for any edge $l \xrightarrow{g,\sigma,R} l'$ in $\mathcal{A}$ such that $v \models g$ and $v' = v[R \leftarrow 0]$.*

Consider any timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \Sigma, l_0, E)$ and let $[\![\mathcal{A}]\!] = (S, s_0, \Sigma, \rightarrow)$. A *run* of $[\![\mathcal{A}]\!]$ is a finite or infinite sequence $\rho = (s_i, \sigma_i, \tau_i)_{i \geq 0}$, where $s_i = (l_i, v_i) \in S$, $\sigma_i \in \Sigma$, $\tau_i \in \mathbb{R}_{\geq 0}$ and $s_i \xrightarrow{\tau_i, \sigma_i} s_{i+1}$ for all $i \geq 0$. The word $l_0 l_1 \ldots$ is the *trace* of the run ρ, denoted $\mathsf{trace}(\rho)$. The i-th state s_i of a run ρ is denoted by $(\rho)_i$.

We define the usual notion of *regions* [1]. Pick a timed automaton $\mathcal{A}$ with clock set $\mathcal{C}$, and let M be the largest constant that appears in its guards. For any $(l, u), (l', v) \in \mathcal{L} \times \mathbb{R}_{\geq 0}^{C}$, we let $(l, u) \simeq (l', v)$ if, and only if, $l = l'$ and for all $x, y \in \mathcal{C}$, the following conditions are satisfied:

- either $\lfloor u(x) \rfloor = \lfloor v(x) \rfloor$ or $u(x), v(x) > M$;
- if $u(x) \leq M$, $\mathsf{frac}(u(x)) = 0$ iff $\mathsf{frac}(v(x)) = 0$;
- if $u(x), u(y) \leq M$, $\mathsf{frac}(u(x)) < \mathsf{frac}(u(y))$ iff $\mathsf{frac}(v(x)) < \mathsf{frac}(v(y))$,

where $\mathsf{frac}(\cdot)$ denotes the fractional part. The equivalence class of a state (l, v) for the relation $\simeq$ is denoted by $\mathsf{reg}((l, v))$, and called a *region* of $\mathcal{A}$. The *region automaton* of $\mathcal{A}$ is a finite automaton $\mathcal{R}(\mathcal{A})$ defined as follows. The states of $\mathcal{R}(\mathcal{A})$ are regions r of $\mathcal{A}$. There is a transition from r to r' labelled by $\sigma \in \Sigma$ if there is an edge (l, g, σ, R, l') such that for some $(l, u) \in r$ and $d \geq 0$, $u + d \models g$, and $(l', u[R \leftarrow 0]) \in r'$. This automaton is known to be time-abstract bisimilar to $[\![\mathcal{A}]\!]$ [1]. The number W of regions is bounded by $|\mathcal{L}| \cdot (2M + 2)^{|\mathcal{C}|} \cdot |\mathcal{C}|! \cdot 2^{|\mathcal{C}|}$. A *progress cycle* in $\mathcal{A}$ is a cycle in $\mathcal{R}(\mathcal{A})$ along which each clock $x \in \mathcal{C}$ is either reset or remains larger than M.

2.2 Robust Model-Checking of Timed Automata

It has been remarked long ago that the semantics of timed automata is not realistic: while this was first exemplified by the so-called *Zeno runs*, the problem goes far beyond, and includes other convergence phenomena [7], or isolated traces [10].

Among the possible approaches to circumvent this problem, *robust model checking* was introduced in [14]: it consists in checking a given property on the extended version of the timed automaton under study; here, *extended* includes clock drifts (clocks may evolve at different rates between $1 - \epsilon$ and $1 + \epsilon$) and guard enlargement. Robust model checking consists in deciding the existence of positive values for ϵ and/or δ for which the property holds in the extended timed automaton. In this paper, we only focus on guard enlargement (*i.e.*, we assume $\epsilon = 0$, so that clocks won't drift); in that setting, robust model checking amounts to deciding the existence of a positive δ for which $\mathcal{A}_\delta$ satisfies a given property.

Take the timed automaton depicted on Fig. 2, and the property that the rightmost location ℓ_3 is never reached. While this property can be checked to hold under the classical semantics, any positive enlargement of the clock constraints will make location ℓ_3 reachable (see [8]); this timed automaton does not *robustly* fulfill the safety property.

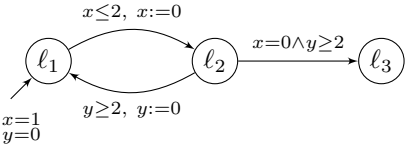

Fig. 2. A (non-robust) timed automaton

Robust model checking has been revisited recently in the setting of *implementability* [9]. Implementability also involves a new semantics for timed automata, the so-called *program semantics*, which simulates the execution of timed automata on a simplified hardware (with digital clock and finite-frequency CPU). This semantics can be over-approximated by the enlarged semantics, so that robust model checking provides an approximate technique to check implementability of timed automata [8].

Robust model checking was proved decidable for safety properties in [14], for timed automata in which all cycles are progress cycles. This was then extended to ω-regular properties [4], and then to timed properties [5].

3 Results

The following theorem is our main result.

Theorem 3. *Let $\mathcal{A}$ be a timed automaton and W be the number of regions of $\mathcal{A}$. Consider any $0 < \delta_0 < \left(8|\mathcal{C}|^2 \cdot (W+1)\right)^{-1}$ if $\mathcal{A}$ has only progress cycles, and $0 < \delta_0 < \left(5(W+1) \cdot |\mathcal{C}|^3 \cdot (2 \cdot |\mathcal{L}| \cdot |\mathcal{C}|! \cdot 4^{|\mathcal{C}|} + 4)^2\right)^{-1}$ otherwise. For any ω-regular property[2] ϕ, if $\mathcal{A}_\delta \models \phi$ for some positive δ, then $\mathcal{A}_{\delta_0} \models \phi$.*

Thus, one can decide robust satisfaction of any ω-regular property by checking whether the property holds for some fixed δ_0, which only depends on the size of the automaton. Now, using the usual model-checking algorithms, one can analyze $\mathcal{A}_{\delta_0}$ in polynomial space. In fact, the greatest constant in $\mathcal{A}$ is now multiplied by $\frac{1}{\delta_0}$ and the regions of $\mathcal{A}_{\delta_0}$ can still be encoded in polynomial space. The problem is PSPACE-hard since it is already for timed automata with progress cycles [5].

Corollary 4. *Robust model-checking of general timed automata against ω-regular properties is* PSPACE*-complete.*

The proof of Theorem 3 uses the encoding by channel machines proposed in [5]. The complex mechanism of the channel machine is not required for our purpose. We therefore hide it as much as possible and focus on the underlying transition system. The transition system and its relation to timed automata is presented in section 4. In section 5, we state our main technical results (namely, the pumping lemma and the cycling lemma), which we use to prove Theorem 3. The rest of the paper is then devoted to the proof of these lemmas.

Remark 1. The results of [8] can be lifted to the region-automaton construction, by adding extra transitions representing (progress) cycles [4]. Using our results, this can be further adapted by adding transitions corresponding to weak cycles, which can be detected on the transition system of the channel automaton.

4 Encoding by Channel Machines

In this section, we show how we encode the behaviour of $\mathcal{A}_\delta$ (where $\mathcal{A}$ is a timed automaton and $\delta > 0$) as the transition system of a channel machine. Channel machines are finite-state automata equipped with a FIFO channel. Intuitively, a state of $\mathcal{A}_\delta$ is encoded as follows: the location and the integer parts of the clocks are stored in a *discrete* location, while the channel contains the clock symbols, ordered according to their fractional parts. When a clock is popped out from the tail of the channel, it is (almost) immediately pushed back to the head of the

[2] With *ω-regular property*, we mean state-based properties whose truth value only depends on the set of locations that are visited infinitely often. By an adequate product, we could handle properties expressed by, say, deterministic Muller automata (hence including LTL properties); we omit these details to keep focus on the main objectives of this paper. For an ω-regular property ϕ, we write $\mathcal{A} \models \phi$ when all the runs of automaton $\mathcal{A}$ satisfy ϕ.

channel (hence it is assumed to have small fractional part). This corresponds to a delay transition along which that clock has changed integer value. Some additional symbols (Δ's) will appear on the channel, which serve for refining the region equivalence, and for approximating the values of the clocks. Our encoding is a slightly simplified version of [5], ignoring technicalities such as non-deterministic renaming and occurence testing operations. This is sufficient since the transition system will have access to the whole content of the channel, not only to the head and the queue (as this is the case for the standard mechanism of the channel machines).

We fix for the rest of this section a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \Sigma, l_0, E)$, and a symbol $\Delta \notin \mathcal{C}$.

4.1 Channel Machine Associated to a Timed Automaton

For any word w over alphabet $2^{\mathcal{C}} \setminus \{\emptyset\} \cup \{\Delta\}$, $|w|_\Delta$ denotes the number of occurences of symbol Δ in w, and for any $x \in \mathcal{C}$, $|w|_x$ denotes the number of times x appears inside the symbols of $2^{\mathcal{C}}$ in w. For any integer $N > 0$, let Γ_N be the set of words w over alphabet $2^{\mathcal{C}} \setminus \{\emptyset\} \cup \{\Delta\}$ such that $|w|_\Delta = N$ and $|w|_x \leq 1$ for all $x \in \mathcal{C}$. For any $w \in \Gamma_N$, we define $\mathsf{right}^w_\Delta(x)$ as 0 if $|w|_x = 0$, and as the number of symbols Δ that appears on the right of the (unique) symbol containing x in w. We define $\mathsf{left}^w_\Delta(x)$ symmetrically.

Let M denote the largest constant that appears in $\mathcal{A}$. We assume that clocks are indexed by $\{1, \ldots, n\}$ for some $n > 0$, and we write $\mathcal{C} = \{x_1, \ldots, x_n\}$. We define *the channel machine associated with* $\mathcal{A}$ as the transition system $\mathcal{C}_\mathcal{A}(\Delta^N)$, parameterized by an integer $N \geq 0$, as follows. The states of $\mathcal{C}_\mathcal{A}(\Delta^N)$ are elements of $(\mathcal{L} \times 2^{\mathcal{C}} \times \{0, \ldots, M, \infty\}^{\mathcal{C}}) \times \Gamma_N$. The first component of a state q is the *discrete state*, made of a location, denoted by $\mathsf{loc}(q)$, the set of clocks that have integer values, and a mapping from clocks to their integer parts which is denoted by $\mathsf{int}(q)$ (we write ∞ if it is larger than M); the second component is the *channel content* where clocks are ordered according to their fractional parts. For a state $q = (d, w)$, we extend $\mathsf{right}_\Delta(\cdot)$ as $\mathsf{right}^q_\Delta(x) = \mathsf{right}^w_\Delta(x)$, and similarly for $\mathsf{left}^q_\Delta(x)$. The initial state of $\mathcal{C}_\mathcal{A}(\Delta^N)$ is $((l_0, \mathcal{C}, \mathbf{0}), \Delta^N)$, where l_0 is the initial location of $\mathcal{A}$. Forgetting Δ's, each state q of $\mathcal{C}_\mathcal{A}(\Delta^N)$ naturally encodes a region of $\mathcal{A}$, which we denote by $\mathsf{reg}(q)$: if $\mathcal{C} = \{x, y, z\}$, the state $((l, \{y\}, \binom{\lfloor x \rfloor = \lfloor y \rfloor = 2}{\lfloor z \rfloor = 1}), \Delta^2\{x\}\Delta\{z\}\Delta^4)$ encodes the region where $y = 2$, $\lfloor x \rfloor = 2$, $\lfloor z \rfloor = 1$ and $0 < \mathsf{frac}(x) < \mathsf{frac}(z)$. We will explain the role of the Δ's later.

Transitions of $\mathcal{C}_\mathcal{A}(\Delta^N)$ are labelled by $\Sigma \cup \{\tau\}$, where $\tau \notin \Sigma$. *Elementary delay transitions* are defined as follows, for any state $((l, Z, \iota), w)$:

$$\textit{(i)}\ ((l, Z, \iota), w) \xrightarrow{\tau} ((l, \emptyset, \iota), Z \cdot w) \qquad \textit{(ii)}\ ((l, \emptyset, \iota), w \cdot \Delta) \xrightarrow{\tau} ((l, \emptyset, \iota), \Delta \cdot w)$$
$$\textit{(iii)}\ ((l, \emptyset, \iota), w \cdot X) \xrightarrow{\tau} ((l, X', \iota'), w), \text{ where } \iota'(x) = \iota(x) + 1 \text{ for } x \in X,$$
$$\text{and } \iota'(y) = \iota(y) \text{ for } y \notin X, \text{ and } X' = X \cap \iota'^{-1}([0, M]),$$

where we write $M + 1 = \infty$ (all clocks whose integral part reaches $M + 1$ are abstracted to ∞ and they do not appear anymore in the word of Γ_N—this is the role of $\iota'^{-1}([0, M])$). *Delay transitions* are defined as the reflexive and transitive

closure of $\xrightarrow{\tau}$, and we also write $\xrightarrow{\tau}$ whenever $\xrightarrow{\tau}^*$. Viewing w as the content of a channel (the head being the first letter and the tail being the last letter), the delay transitions correspond to sequences of reads and writes at the channel, while the discrete state is changed to keep track of the integer parts, whenever a clock subset symbol is read. We say that a clock y *disappears* during a delay transition whenever the rule (iii) is applied, with $y \in X$ and $y \notin X'$. Obviously, delay transitions in $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ correspond to time elapsing in $\mathcal{A}$.

We now define when a state of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ satisfies a guard. A clock formula $x \leq k$ is *exactly satisfied* by a state $q = ((l, Z, \iota), w)$ if either $\iota(x) \leq k-1$, or $\iota(x) = k$ and $x \in Z$ (this is equivalent to say that $\mathsf{reg}(q)$ satisfies $x \leq k$). The formula is *satisfied* if either it is exactly satisfied or $\iota(x) = k$ and $\mathsf{left}^w_{\Delta}(x) \leq 1$. Intuitively, the value of x is then a bit larger than k (this will be made clearer when explaining the role of the Δ's). A formula $x \geq k$ is exactly satisfied if $\iota(x) \geq k$, and satisfied if it is exactly satisfied or if $\iota(x) = k-1$ and $\mathsf{right}^w_{\Delta}(x) \leq 1$. *Action transitions* are defined as follows. For any edge $l \xrightarrow{g,\sigma,R} l'$ of $\mathcal{A}$, we let $((l, Z, \iota), w) \xrightarrow{\sigma} ((l, Z \cup R, \iota'), w')$ if $((l, Z, \iota), w)$ satisfies g, $\iota'(x) = 0$ if $x \in R$ and $\iota'(x) = \iota(x)$ if $x \notin R$, and w' is obtained from w by removing the occurences of all clocks in R. This rule is not a valid operation in a channel machine, since some symbols may be removed from w, and checking guards requires reading the tail of w. However this can be simulated using rewriting and occurrence testing, see [5]. Action transitions in $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ where guards are exactly satisfied correspond to action transitions in $\mathcal{A}$. Non-exact satisfaction of guards represents enlarged timing constraints. We will see the precise correspondence later.

A *path* of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ is a sequence $\pi = (q_i, \sigma_i)_{i \geq 1}$ where q_i's are states of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$, and $\sigma_i \in \Sigma \cup \{\tau\}$, and there is a transition labelled by σ_i from q_i to q_{i+1}. We only consider w.l.o.g paths that are alternations of delay and action transitions. The *length* of π, denoted $|\pi|$, is the length of the sequence π. We denote by $\mathsf{trace}(\pi)$ the sequence of locations $\mathsf{loc}(q_0)\mathsf{loc}(q_2)\ldots$ visited by π, and by $\pi_{i\ldots j}$ the path defined by the subsequence between indices i and j. A path is *exact* if all guards in its transitions are satisfied exactly. The i-th state of a path π is denoted by $(\pi)_i$.

Representation of the states of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$. In the sequel, to help manipulate the transition system of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$, we use a *flat representation* of the states. We say that $((l, Z, \iota), \widetilde{w})$ is a *flat representation* of state $(d, w) = ((l, Z, \iota), w)$ whenever $\widetilde{w} \in \left(\mathcal{C} \cup \Delta \cup \Delta^{-1}\right)^*$ can be written as

$$\widetilde{w} = \Delta^{n_0} x_{i_1} \Delta^{n_1} \ldots x_{i_m} \Delta^{n_m}, \tag{1}$$

where $\{x_{i_j} \mid 1 \leq j \leq m\} = \{x \in \mathcal{C} \mid \iota(x) \leq M\}$ is the set of clocks whose integral part is no more than M in (d, w) (some appear in w, some, whose values is integral, do not appear in w), $n_0, \ldots, n_m \geq -1$, and:

- if we remove the maximal prefix of the form $\Delta^{-1} y_1 \Delta^{-1} \ldots \Delta^{-1} y_p$,
- if we remove the maximal suffix of the form $y_p \Delta^{-1} \ldots \Delta^{-1} y_1 \Delta^{-1}$, and
- if we replace all maximal factors of the form $y_1 \Delta^{-1} \ldots \Delta^{-1} y_p$ by $\{y_1, \ldots, y_p\}$,

then we obtain w. Such a flat representation $(d, \widetilde{w})$ contains exactly the same information as w (though with some redundancy) but will be easier to manipulate.

Note that there can be several flat representations for a given state (since both $x\Delta^{-1}y$ and $y\Delta^{-1}x$ can be used to represent $\{x,y\}$). Two clocks x_{i_j} and $x_{i_{j+1}}$ which are separated by Δ^{-1} in $\widetilde{w}$ will belong to the same set in w, hence the corresponding clocks will have the same fractional part in $\mathsf{reg}((d,w))$. If $n_0 = -1$, then x_{i_1} has an integer value in (d,c), and similarly for x_{i_m} if $n_m = -1$. Notice that all clocks whose values are (strictly) less than $M+1$ are present in this word, even those having an integral value. When clock indices $i_1,\ldots,i_m$ are clear from the context, or implicit, we also represent the channel content (1) by its block sizes $(n_0,n_1,\ldots,n_m)$. The channel content in (1) defines $m+1$ *blocks*, which are words of $\Delta^* \cup \{\Delta^{-1}\}$ separated by the clock symbols. We enumerate these from 0 to m, and say, for example, that block i has *size* n_i. In the rest, we only use flat representations, for which we easily infer the transition relation.

Example 1. The following is a path in $\mathcal{C}_{\mathcal{A}}(\Delta^{14})$, for the timed automaton $\mathcal{A}$ depicted on Fig. 2. This path simulates the run of the automaton that enters location ℓ_1 with $x=1$ and $y=0$; delays in ℓ_1 for $1+\delta$ time units, and then moves to ℓ_2, resetting x along that transition. It then waits for $1-2\delta$ time units, until $y = 2-\delta$, and goes back to ℓ_1, and so on.

$$\Big((\ell_1,\{x,y\},\binom{\lfloor x\rfloor=1}{\lfloor y\rfloor=0}),\Delta^{-1}x\Delta^{-1}y\Delta^{14}\Big) \xrightarrow{\tau} \Big((\ell_1,\emptyset,\binom{\lfloor x\rfloor=2}{\lfloor y\rfloor=1}),\Delta x\Delta^{-1}y\Delta^{13}\Big) \xrightarrow[x:=0]{x\le 2}$$
$$\Big((\ell_2,\{x\},\binom{\lfloor x\rfloor=0}{\lfloor y\rfloor=1}),\Delta^{-1}x\Delta y\Delta^{13}\Big) \xrightarrow{\tau} \Big((\ell_2,\emptyset,\binom{\lfloor x\rfloor=0}{\lfloor y\rfloor=1}),\Delta^{12}x\Delta y\Delta\Big) \xrightarrow[y:=0]{y\ge 2}$$
$$\Big((\ell_1,\{y\},\binom{\lfloor x\rfloor=0}{\lfloor y\rfloor=0}),\Delta^{-1}y\Delta^{12}x\Delta^2\Big) \xrightarrow{\tau} \Big((\ell_1,\emptyset,\binom{\lfloor x\rfloor=1}{\lfloor y\rfloor=2}),\Delta x\Delta^2 y\Delta^{11}\Big)\cdots$$

4.2 Relation with Timed Automata

We now define the relation between $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ and $\mathcal{A}$, through a relation on their associated time-abstract transition systems. Next, we write $s \overset{\sigma}{\Longrightarrow} s'$ if $s \xrightarrow{\tau} s'' \xrightarrow{\sigma} s'$ for some state s'' (where $\xrightarrow{\tau}$ is a delay transition).

Definition 5. *Let $\mathcal{S} = (S, s_0, \Sigma, \to)$ be an LTTS. A relation $R \subseteq S \times S$ is a* two-way simulation *if for all $(s_1,s_2) \in R$, if $s_1 \overset{\sigma}{\Longrightarrow} s_1'$ for some $\sigma \in \Sigma$ then $s_2 \overset{\sigma}{\Longrightarrow} s_2'$ for some s_2' with $(s_1',s_2') \in R$, and if $s_1' \overset{\sigma}{\Longrightarrow} s_1$ for some $\sigma \in \Sigma$, then $s_2' \overset{\sigma}{\Longrightarrow} s_2$ for some s_2' with $(s_1',s_2') \in R$. A state s_2* simulates *a state s_1 whenever there exists a two-way simulation R such that $(s_1,s_2) \in R$. In that case we write $s_1 \sqsubseteq s_2$.*

For any state (d,w) of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ with $w \in \Gamma_N$, we define $\mathsf{concrete}\big((d,w)\big)$ as a subset of $\mathsf{reg}((d,w))$ as follows. It contains a state $(l,v) \in \mathsf{reg}((d,w))$, if, and only if, $l = \mathsf{loc}((d,w))$ and there exists $\delta \in \mathbb{R}^N_{\ge 0}$ that satisfies $0 < \delta(1) < \delta(2) < \ldots < \delta(N) < 1$, $\delta(i+1) - \delta(i) = \frac{1}{N}$ for all $1 \le i \le N-1$, and $\delta(i) \neq \mathsf{frac}(v(x))$ for all i and $x \in \mathcal{C}$ that appears in w, and, assuming that $\delta(i)$ is the value of the i-th Δ-symbol in w, the extended valuation $v \cup \{\delta(i)\}_{1\le i\le N}$ is ordered according to w.

For example, consider the state $(d,w) = \big((l,\{x\},(\binom{\lfloor x\rfloor=\lfloor y\rfloor=0}{\lfloor z\rfloor=1})),\Delta^3 z\Delta^3 y\Delta^4\big)$, and valuation v defined by $v(x)=0$, $v(y)=0.6$ and $v(z)=1.3$. We have

$v \in \mathsf{concrete}\big((d,c)\big)$ since for $\delta \in \mathbb{R}^{10}_{\geq 0}$ with $\delta(i) = 0.05 + \frac{i-1}{10}$, the ordering of the fractional parts of $v(x), v(y), v(z), \delta(1), \ldots, \delta(10)$ agree with that given in (d,w).

Lemma 6. *For any timed automaton $\mathcal{A}$ and $N \geq 1$, $[\![\mathcal{A}_{\frac{1}{N}}]\!] \sqsubseteq \mathcal{C}_{\mathcal{A}}(\Delta^N) \sqsubseteq [\![\mathcal{A}_{\frac{2}{N}}]\!]$.*

A weaker version of this lemma was proven in [5]. The two-way simulations in the above lemma are given by relations R defined between states of $[\![\mathcal{A}_\delta]\!]$ and $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ by $(l,v)R(d,w)$ iff $v \in \mathsf{concrete}\big((d,w)\big)$ and $l = \mathsf{loc}(d)$.

4.3 Δ-Distance in $\mathcal{C}_{\mathcal{A}}(\Delta^N)$

If q is a state of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$, we will denote by $[q]$ the topological closure of the region encoded by q. The following lemma characterizes the inclusion of region closures using flat representations and follows from definitions.

Lemma 7. *Let q and q' be two states of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$, and let (d,w) be a flat representation of q, with $w = \Delta^{n_0} x_{i_1} \Delta^{n_1} \ldots x_{i_m} \Delta^{n_m}$. Then $[q] \subseteq [q']$ iff w' has a flat representation of the form (d',w') where $w' = \Delta^{n'_0} x_{i_1} \Delta^{n'_1} \ldots x_{i_m} \Delta^{n'_m}$, s.t.*

- *$\mathsf{loc}(d) = \mathsf{loc}(d')$,*
- *for every $0 \leq i \leq m$, $n'_i = -1$ implies $n_i = -1$, and*
- *there exists $1 \leq r \leq m$ s.t. $n_{r+1} = n_{r+2} = \cdots = n_m = -1$, and:*
  - *for every $1 \leq j \leq r$, $\mathsf{int}(d)(x_{i_j}) = \mathsf{int}(d')(x_{i_j})$,*
  - *for every $r < j \leq m$, $\mathsf{int}(d)(x_{i_j}) = \mathsf{int}(d')(x_{i_j}) + 1$.*

We now define an edit-distance between the states of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$, called the *$\Delta$-distance*. We define the Δ-distance between any pair of states q and q' as infinite unless $[q] \subseteq [q']$ or $[q'] \subseteq [q]$. Fix two flat representations (d,w) and (d',w') that satisfy the conditions in Lemma 7 with block sizes $\boldsymbol{n}$ and $\boldsymbol{n'}$. We define $d_\Delta\big(q,q'\big) = \sum_i (\max({n'_i}^+ - {n_i}^+, 0))$, and notice that this is independent of the choice of the flat representations. This function can be seen to be symmetric (by the fact that both words have the same total number of Δ symbols), and to satisfy the triangular inequality. However, when the function equals 0, this does not imply the equality between states due to the -1-sized blocks. This pseudo-distance has the following important property.

Lemma 8. *For any timed automaton $\mathcal{A}$ and $N \geq |\mathcal{C}| + 2$, for any states q, q' of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$, $d_\Delta(q,q') \leq \frac{N}{|\mathcal{C}|} - 2$ implies that $[q] \cap [q'] \neq \emptyset$.*

5 Proof

5.1 Proof of the Main Theorem

The main theorem is a consequence of the following lemma, using Lemma 6.

Lemma 9. *Let $\mathcal{A}$ be any timed automaton and let W denote its number of regions. Let $K_0 = 2|\mathcal{C}|! \cdot |\mathcal{L}| \cdot 4^{|\mathcal{C}|} + 4$, and $N_1 \geq 8|\mathcal{C}|^2 \cdot (W+1)$ if $\mathcal{A}$ has only progress cycles, and $N_1 \geq 5|\mathcal{C}|^3 \cdot K_0^2 \cdot (W+1)$ otherwise. For any ω-regular property ϕ, if there exists $N > 0$ such that $\mathcal{C}_{\mathcal{A}}(\Delta^N) \models \phi$, then $\mathcal{C}_{\mathcal{A}}(\Delta^{N_1}) \models \phi$.*

We show that if $\mathcal{C}_{\mathcal{A}}(\Delta^{N_1}) \not\models \phi$, then $\mathcal{C}_{\mathcal{A}}(\Delta^{N}) \not\models \phi$ for all $N > 0$. The case where $N < N_1$ is easy, and is implied in the following lemma.

Lemma 10. *For any timed automaton $\mathcal{A}$ and any $N > 0$, $\mathcal{C}_{\mathcal{A}}(\Delta^{N}) \sqsubseteq \mathcal{C}_{\mathcal{A}}(\Delta^{N-1})$.*

The idea is that any path of the channel machine can be carried out when a Δ symbol is removed from the channel. In fact, all guards satisfied in the former system are also satisfied when a Δ symbol is removed. This implies that $\mathcal{C}_{\mathcal{A}}(\Delta^{N_1}) \sqsubseteq \mathcal{C}_{\mathcal{A}}(\Delta^{N})$, hence if the property is violated by a path of $\mathcal{C}_{\mathcal{A}}(\Delta^{N_1})$, then $\mathcal{C}_{\mathcal{A}}(\Delta^{N})$ also has a path violating the property.

In the case where $N_1 < N$, we do not have a simulation between $\mathcal{C}_{\mathcal{A}}(\Delta^{N})$ and $\mathcal{C}_{\mathcal{A}}(\Delta^{N_1})$, but assuming that $\mathcal{C}_{\mathcal{A}}(\Delta^{N_1})$ has a path violating the desired property, we transform it into a path of $\mathcal{C}_{\mathcal{A}}(\Delta^{N})$ also violating the property. This transformation may modify the trace but it does not affect the satisfaction of ϕ (locations that appear infinitely often remain the same). This is stated in the following pumping lemma.

Lemma 11 (Pumping Lemma). *Consider a timed automaton $\mathcal{A}$, and let W denote its number of regions. Let $K_0 = 2|\mathcal{C}|!\cdot|\mathcal{L}|\cdot 4^{|\mathcal{C}|}+4$, and $N_1 \geq 8|\mathcal{C}|^2 \cdot (W+1)$ if $\mathcal{A}$ has only progress cycles, and $N_1 \geq 5|\mathcal{C}|^3 \cdot K_0^2 \cdot (W+1)$ otherwise. Then, for any path π of $\mathcal{C}_{\mathcal{A}}(\Delta^{N_1})$, for any $L \geq 0$, there exists a path π' of $\mathcal{C}_{\mathcal{A}}(\Delta^{N_1+L})$, such that the same set of locations appear infinitely often in π and π'.*

The rest of the paper is devoted to the proof of the pumping lemma. The path of $\mathcal{C}_{\mathcal{A}}(\Delta^{N_1+L})$ is obtained by repeating some factors of the path of $\mathcal{C}_{\mathcal{A}}(\Delta^{N_1})$, then repeating some factors of the resulting word, and repeating this a finite number of times. This operation preserves ω-regular properties.

Overview of the proof. We start by studying, in Subsection 5.2, how the sizes of the blocks evolve along a path. We characterize the blocks whose sizes do not become small along a path; these are called the blocks that *stay united*. We prove a pumping lemma for these blocks (Lemma 12). Then, we study, in Subsection 5.3, exact paths of $\mathcal{C}_{\mathcal{A}}(\Delta^{N_1})$ and show that for any path of bounded length, there is an exact path that follows the same trace and that is close in terms of Δ-distance (Lemma 14). In Subsection 5.4, we apply the above results to bounded paths to prove the pumping lemma for unbounded paths.

5.2 Pumping Lemma: Bounded Case

We fix a timed automaton $\mathcal{A} = (\mathcal{L}, \mathcal{C}, \Sigma, l_0, E)$ and $N > 0$. Let W denote the number of regions of $\mathcal{A}$. For any channel content $w \in \Gamma_N$ with clocks $x_{i_1}, \ldots, x_{i_m}$ and block sizes $(n_0, \ldots, n_m)$, and $L \geq 0$, we define $w[x_{i_j} \leftarrow x_{i_j}\Delta^L]$ as the word of Γ_{N+L} obtained from w by replacing the j-th block Δ^{n_j} with $\Delta^{n_j^+ + L}$. We assume that i_0 is an index fixed to 0. We extend the above definition to the 0-th block, by writing $w[x_{i_0} \leftarrow x_{i_0}\Delta^L] = w[x_{i_1} \leftarrow \Delta^L x_{i_1}]$, obtained by inserting L new Δ symbols in the 0-th block.

We fix the constant $N_0 = 2W + 2$. A block is *small* if it has size $\{-1, 0, 1\}$, *medium* if it has size $\{2, \ldots, N_0 - 1\}$, and *large* otherwise. An important observation about $\mathcal{C}_{\mathcal{A}}(\Delta^{N})$ is that, if a guard is satisfied at some state (d, w), then,

when an arbitrary number of Δ symbols are inserted in medium/large blocks, the same guard is still satisfied. However, if we insert additional Δ symbols inside small blocks, then formulas which are satisfied but not exactly satisfied may not be satisfied anymore. We define a notion of *staying united* along a path for blocks. Intuitively, such blocks are those that are either always at least medium, or are cut into at least one medium/large block along the path. We then show that one can insert any number of Δ symbols inside a block that stays united, and adapt the original path of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ to a path in $\mathcal{C}_{\mathcal{A}}(\Delta^{N+L})$.

We define a relation on pairs of states and clock indices as follows. Let $q = (d, w)$ and $q' = (d', w')$ denote two flat representations of states of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ such that $q \xrightarrow{\tau} q'$ is an elementary delay transition. We let $(q, i) \prec (q', j)$ whenever for every integer $L > 0$, there is a delay transition $q[x_i \leftarrow x_i\Delta^L] \xrightarrow{\tau} q'[x_j \leftarrow x_j\Delta^L]$. This relation can be characterized rather easily by analysing all possible cases for elementary delays $q \xrightarrow{\tau} q'$. This relation is extended to the transitive closure of delay transitions. Similarly, assuming $q \xrightarrow{\sigma} q'$ is an action transition, we write $(q, i) \prec (q', j)$ whenever for every integer L, there is an action transition $q[x_i \leftarrow x_i\Delta^L] \xrightarrow{\sigma} q'[x_j \leftarrow x_j\Delta^L]$. This can be characterized easily as well.

For any finite path π in $\mathcal{C}_{\mathcal{A}}(\Delta^N)$, and any block i_1 in $(\pi)_1$, we say that *block* i_1 *stays united* in π, if block i_1 is large in $(\pi)_1$, and if there exists clock indices $i_2, \ldots, i_m$ such that $((\pi)_1, i_1) \prec ((\pi)_2, i_2) \prec \ldots \prec ((\pi)_n, i_n)$, with $n = |\pi|$.

By definition, the paths along which a block stays united are not sensitive to the precise size of those blocks. This is formalized in the following lemma, which is a pumping lemma for particular finite paths.

Lemma 12. *Let π be a path of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ such that $((\pi)_1, i_1) \prec ((\pi)_2, i_2) \prec \ldots \prec ((\pi)_n, i_n)$, for some $n > 0$. Then for any $L > 0$, $\mathcal{C}_{\mathcal{A}}(\Delta^{N+L})$ has a path π' with $(\pi')_j = (\pi)_j[x_{i_j} \leftarrow x_{i_j}\Delta^L]$ for any $1 \leq j \leq n$ and $\mathsf{trace}(\pi) = \mathsf{trace}(\pi')$.*

We now state a lower bound on the length of a path along which a block does not stay united. The idea is that if a block does not stay united, then whenever it is cut in two parts during a transition, either one of the resulting blocks is small, or none of them stays united in the rest of the path.

Lemma 13. *Let π be a path of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ with $|\pi| = p$. Then all blocks in $(\pi)_1$ that have size at least $p + 1$ stay united along π.*

5.3 Making Exact Paths

In this section, we show how to transform an arbitrary path of bounded length of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ into an exact one. By definition, if a path is not exact, then there are states with small blocks. The idea of our transformation is to replace all small blocks and the blocks that do not stay united by -1-sized blocks, while preserving the ordering of the clocks. Notice that by Lemma 7, the states obtained by this operation define closed subregions of those defined by the original states. Clearly, if all small blocks have size -1, then any guard that is satisfied by a state is satisfied exactly, so the new path is exact.

Take $N \geq (|\mathcal{C}|+1) \cdot N_0$, and fix a path π of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ of length $n \leq N_0 - 1$. For each $1 \leq i < i' \leq n$, we associate with $(\pi)_i$ a state $H(\pi, i, i')$ where any small or medium block that does not stay united along $\pi_{i\ldots i'}$ is replaced by blocks of size -1. Formally, let $j_1, \ldots, j_k$ denote the indices of the blocks of $(\pi)_i$ that do not stay united along the path $\pi_{i\ldots i'}$. Let us write $(\pi)_i = (d, \boldsymbol{n})$. We define $H(\pi, i, i') = (d', \boldsymbol{n}')$ as follows. We let $n'_{j_1} = \ldots = n'_{j_k} = -1$, and $n'_{j_0} = n_{j_0} + n^+_{j_1} \ldots + n^+_{j_k}$, for the large block with the minimal index j_0, which exists by the choice of N. (Notice that the closed region $[H(\pi, i, i')]$ is independent of j_0). We have, by Lemma 7, $[H(\pi, i, i')] \subseteq [(\pi)_i]$. The same lemma implies that any state q with $[q] \subseteq [H(\pi, i, i')]$, has only blocks that stay united along $\pi_{i\ldots i'}$. Observe that because $(\pi)_i$ has at least one large block, by Lemma 13, $H(\pi, i, i')$ is well-defined. Last, we have $d_{\Delta}((\pi)_i, H(\pi, i, i')) \leq |\mathcal{C}| \cdot N_0$ by construction.

We construct exact paths that are "close" to the original ones, as follows.

Lemma 14. *Let $\mathcal{A}$ be any timed automaton having only progress cycles, and $N \geq (|\mathcal{C}|+1) \cdot N_0$. Let π a path of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ of length at most N_0. Then, there exists an exact path π' of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ over trace $\mathsf{trace}(\pi_{1\ldots N_0})$, with $(\pi')_1 = H(\pi, 1, N_0)$ and $[(\pi')_i] \subseteq [H(\pi, i, N_0)]$ and $d_{\Delta}(\pi, \pi') \leq (|\mathcal{C}|+1)N_0$ for all $1 \leq i \leq N_0$.*

A result similar to Lemma 14 was given in [8, Th. 44] for runs of timed automata and distance d_∞ over valuations. The proof there involved approximation of the width of parametric DBMs. Our approach is in some sense closer to the input timed automaton, which may explain why we get an exponentially better distance to the original run.

As one might expect, exact paths satisfy the following property. The idea is that exact paths of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ are not sensitive to the sizes of the blocks.

Lemma 15. *Let $\mathcal{A}$ be any timed automaton, $N \geq 1$ and π an exact path of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$. Then, for any $N' \geq N$, and any state q of $\mathcal{C}_{\mathcal{A}}(\Delta^{N'})$ with $[q] \subseteq [\mathsf{first}(\pi)]$, there exists an exact path π' over the same trace as π, with $\mathsf{first}(\pi') = q$ and $[(\pi')_i] \subseteq [(\pi)_i]$ for all $1 \leq i \leq |\pi|$. The same property holds backwards: for any $q \in [\mathsf{last}(\pi)]$, there exists an exact path π' over the trace of π in $\mathcal{C}_{\mathcal{A}}(\Delta^{N'})$ with $\mathsf{last}(\pi') = q$ and $[(\pi')_i] \subseteq [(\pi)_i]$ for $1 \leq i \leq |\pi|$.*

5.4 Pumping Lemma with Progress Cycles: Unbounded case

The previous sections dealt with the properties of the bounded paths of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$. We now use these to prove the pumping lemma for infinite paths. Let us first define a transformation on the traces of the runs. For any finite trace $w \in \mathcal{L}^*$, we let $\widetilde{w} = \{u_1^+ u_2^+ \ldots u_n^+ \mid u_1 u_2 \ldots u_n = w\}$.

We first need the following lemma, which is an adaptation of Lemma 29 of [8] to channel machines.

Lemma 16 (Cycling Lemma). *Let $\mathcal{A}$ be any timed automaton, $N \geq 1$ and π an exact progress cycle in $\mathcal{C}_{\mathcal{A}}(\Delta^N)$. Then, for all states q with $[q] \subseteq [\mathsf{last}(\pi)]$, there exists a path π' in $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ with $\mathsf{first}(\pi') = \mathsf{first}(\pi)$ and $[\mathsf{last}(\pi')] \subseteq [q]$, and $\mathsf{trace}(\pi') \in \widetilde{\mathsf{trace}(\pi)}$.*

We are now ready to prove the pumping lemma, for timed automata with progress cycles. Figure 3 illustrates a step of the proof.

Proof (of Lemma 11). We prove the result for $1 \leq L \leq |\mathcal{C}|N_0 - 2$. For larger L, one can repeat this construction. Let $N \geq 4|\mathcal{C}|^2 N_0$, and consider an infinite path π of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ where $\mathsf{first}(\pi)$ is the initial state of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$.

Let $n = N_0 - 1$. Let $G^L(\pi, i, i+n)$ denote the state of $\mathcal{C}_{\mathcal{A}}(\Delta^{N+L})$ obtained from $(\pi)_i$ by inserting Δ^L in the block with minimal index that stays united along $\pi_{i\ldots i+n}$ (such a block exists by Lemma 13). We also define $H^L(\pi, i, i+n)$ by inserting Δ^L to the same block in $H(\pi, i, i+n)$ ($H(\cdot)$ is defined right before Lemma 14). Then, by construction, $d_\Delta\big(H^L(\pi, i, i+n), G^L(\pi, i, i+n)\big) \leq |\mathcal{C}|N_0$.

We now define a path π' of $\mathcal{C}_{\mathcal{A}}(\Delta^{N+L})$ over trace $\widetilde{\mathsf{trace}(\pi)}$. At each step $i \geq 1$, we construct $\pi'_{\beta_i \ldots \beta_{i+1}}$, where $(\beta_i)_{i\geq 1}$ is an increasing sequence. Our construction satisfies $\mathsf{trace}(\pi'_{\beta_i\ldots\beta_{i+1}}) \in \widetilde{\mathsf{trace}(\pi_{\alpha_i\ldots\alpha_{i+1}})}$, and $[(\pi')_{\beta_i}] \subseteq [(\pi)_{\alpha_i}]$, for some possibly different increasing sequence $(\alpha_i)_{i\geq 1}$.

We define $(\pi')_1$ as the initial state of $\mathcal{C}_{\mathcal{A}}(\Delta^{N+L})$, which satisfies $[(\pi')_1] \subseteq [(\pi)_1]$. Suppose now that $\pi'_{1\ldots\beta_i}$ has been constructed for some $\beta_i \geq 1$. By Lemma 12, there is a path g of $\mathcal{C}_{\mathcal{A}}(\Delta^{N+L})$ from $G(\pi, \alpha_i, \alpha_i + n)$ over $\mathsf{trace}(\pi_{\alpha_i\ldots\alpha_i+n})$, such that $(g)_j = (\pi)_{\alpha_i+j}[z_j \leftarrow z_j\Delta^L]$ for some clocks z_j. We then apply Lemma 14 to g to get an exact path h with $(h)_1 = H^L(\pi, \alpha_i, \alpha_i + n)$ over the trace of g, with $d_\Delta\big((h)_j, (g)_j\big) \leq (|\mathcal{C}|+1)N_0$, and $[(h)_j] \subseteq [H^L(\pi, \alpha_i + j, \alpha_i + n)]$ for all $1 \leq j \leq n$. Now, h contains at least $n/2 \geq W$ action transitions, so there exist $1 \leq l_0 < l_1 \leq n$ such that $\mathsf{reg}((h)_{l_0}), \mathsf{reg}((h)_{l_0+1}), \ldots, \mathsf{reg}((h)_{l_1})$ is a progress cycle of the region automaton of $\mathcal{A}$. We have $d_\Delta\big((h)_{l_1}, H^L(\pi, \alpha_i + l_1, \alpha_i + l_1 + n)\big) \leq (3|\mathcal{C}|+1)N_0 - 2$ by combining the following inequalities:

$$d_\Delta((h)_{l_1}, (g)_{l_1}) \leq (|\mathcal{C}|+1)N_0,$$
$$d_\Delta\big((g)_{l_1}, G^L(\pi, \alpha_i + l_1, \alpha_i + l_1 + n)\big) \leq L \leq |\mathcal{C}|N_0 - 2,$$
$$d_\Delta\big(G^L(\pi, \alpha_i + l_1, \alpha_i + l_1 + n), H^L(\pi, \alpha_i + l_1, \alpha_i + l_1 + n)\big) \leq |\mathcal{C}|N_0.$$

By Lemma 15, $\mathcal{C}_{\mathcal{A}}(\Delta^{N+L})$ has a path over $\mathsf{trace}(h)$ from $(\pi')_{\beta_i}$ to some state q with $[q] \subseteq [(h)_{l_1}]$. By Lemma 8, we get $[(h)_{l_1}] \cap [H^L(\pi, \alpha_i + l_1, \alpha_i + l_1 + n)] \neq \emptyset$.

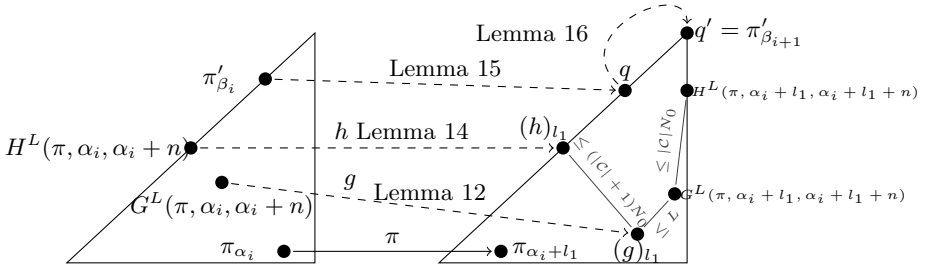

Fig. 3. An induction step of the proof of Lemma 11. Two triangles represent two closed regions. Their sides, interiors and corners are subregions. The proof constructs the dashed paths bottom-up.

Lemma 16 provides a path of $\mathcal{C}_{\mathcal{A}}(\Delta^{N+L})$ from q to some state q' with $[q'] \subseteq [(h)_{l_1}] \cap [H^L(\pi, \alpha_i + l_1, \alpha_i + l_1 + n)]$, over a trace in $\mathsf{trace}(h_{l_0 \ldots l_1})^+$. The concatenation of these two paths define $\pi'_{\beta_i .. \beta_{i+1}}$. This concludes a step of the induction. □

5.5 Pumping Lemma with Non-progress Cycles

In this subsection, we explain the generalization of the proof of the pumping lemma to the case of timed automata with non-progress cycles. Let us call *weak cycle*, a path π of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$ with $[\mathsf{last}(\pi)] \subseteq [\mathsf{first}(\pi)]$ that is not a progress cycle. Thus, π is a cycle along which at least one clock that is present on the channel is not reset. We show that all weak cycles can be transformed into *weak quasi-exact cycles* (defined below). We then define *quasi-exact paths* of $\mathcal{C}_{\mathcal{A}}(\Delta^N)$, which are paths that are exact except along weak quasi-exact cycles. Quasi-exact paths behave very much like exact paths: We adapt each of the previous lemmas involving exact paths for quasi-exact paths, and the proof in the presence of weak cycles is very similar to the proof which assumes progress cycles.

Example 2. The following path of $\mathcal{C}_{\mathcal{A}}(\Delta^{10})$ is an example of weak cycle, in which clock z is not reset.

$$(\ell_1, \{x\}, \cdot), \Delta^3 y \Delta^4 z \Delta^3 \xrightarrow{\tau} (\ell_1, \emptyset, \cdot), \Delta^2 x \Delta^3 y \Delta^4 z \Delta \xrightarrow{\sigma}$$
$$(\ell_2, \{y\}, \cdot), \Delta^2 x \Delta^7 z \Delta \xrightarrow{\tau} (\ell_2, \emptyset, \cdot), y \Delta^2 x \Delta^7 z \Delta \xrightarrow{\sigma'} (\ell_1, \{x\}, \cdot), y \Delta^9 z \Delta$$

It can be seen on this example how clock z prevents from accessing the Δ's that accumulate immediately on its left.

Given a weak cycle, clocks that are reset along the cycle are called *active*, and others *inactive*. Consider a weak cycle π. Suppose that $A = \{i_1, \ldots, i_r\}$ are the indices of the active clocks in π, given in the order of their appearance in $\mathsf{last}(\pi)$. Then, π can be factorized as,

$$\pi = \pi_{i_r} \pi_{i_{r-1}} \ldots \pi_{i_1} \pi', \tag{2}$$

where π_{i_j} ends with the last reset of the clock x_{i_j} in π, and no clock is reset in π'. An important observation that we use is that the size of the block i_j in $\mathsf{last}(\pi_{i_j})$ is determined by the number of Δ's read inside π_{i_j}, for any $1 \leq j \leq r-1$. The block i_r is particular, since its size at the last state is the sum of all the blocks $i_1, \ldots, i_r$ in $\mathsf{first}(\pi)$, plus the Δ's read during π_{i_r}. Among the blocks A of $\mathsf{last}(\pi)$, some will be called *pumpable*. Formally, for $K_0 = |\mathcal{L}| \cdot |\mathcal{C}|! \cdot 4^{|\mathcal{C}|+1} + 4$, we let

$$\mathsf{Pumpable}(\pi) = \{i_j \in A \mid \exists \tau \in \pi_{i_j}, \mathsf{time}_\Delta(\tau) \geq 2 \text{ or } \mathsf{time}_\Delta(\pi_{i_j}) \geq K_0\},$$

where $\mathsf{time}_\Delta(\tau)$ denotes the number of Δ's read from the channel in the delays of a given path τ, and with the abusive notation $\exists \tau \in \pi_{i_j}$ meaning "*for some delay transition in* π_{i_j}". We prove a pumping lemma for pumpable blocks inside weak cycles. In fact, if π_{i_j} has a delay transition which reads at least two Δ symbols, then block i_0 becomes medium or large during this delay, so it can be

extended to read more Δ symbols. But if all delay transitions are too short then this trick cannot be used; in that case when a large number of transitions occur inside π_{i_j}, we show that some factor can be repeated, while additional delays that read Δ's are inserted (this is why we need a large constant K_0).

We then define *quasi-exact paths*, which are paths that are made of delays, exact transitions and weak cycles in which any block that is active is either pumpable, or it ends with size -1. We show that these paths behave very much like exact paths, and we follow step-by-step the same lemmas to construct the proof of the pumping lemma in the general case.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
2. Bouyer, P., Chevalier, F.: On conciseness of extensions of timed automata. Journal of Automata, Languages and Combinatorics 10(4), 393–405 (2005)
3. Bouyer, P., Markey, N., Ouaknine, J., Worrell, J.: The cost of punctuality. In: LICS 2007, Wrocław, Poland, pp. 109–118. IEEE Computer Society Press, Los Alamitos (2007)
4. Bouyer, P., Markey, N., Reynier, P.-A.: Robust model-checking of linear-time properties in timed automata. In: Correa, J.R., Hevia, A., Kiwi, M. (eds.) LATIN 2006. LNCS, vol. 3887, pp. 238–249. Springer, Heidelberg (2006)
5. Bouyer, P., Markey, N., Reynier, P.-A.: Robust analysis of timed automata via channel machines. In: Amadio, R.M. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 157–171. Springer, Heidelberg (2008)
6. Bouyer, P., Markey, N., Sankur, O.: Robust model-checking of timed automata via pumping in channel machines. Research Report LSV-11-19, Laboratoire Spécification et Vérification, ENS Cachan, France (2011)
7. Cassez, F., Henzinger, T.A., Raskin, J.-F.: A comparison of control problems for timed and hybrid systems. In: Tomlin, C.J., Greenstreet, M.R. (eds.) HSCC 2002. LNCS, vol. 2289, pp. 134–148. Springer, Heidelberg (2002)
8. De Wulf, M., Doyen, L., Markey, N., Raskin, J.-F.: Robust safety of timed automata. Formal Methods in System Design 33(1-3), 45–84 (2008)
9. De Wulf, M., Doyen, L., Raskin, J.-F.: Almost ASAP semantics: From timed models to timed implementations. Formal Aspects of Comput. 17(3), 319–341 (2005)
10. Gupta, V., Henzinger, T.A., Jagadeesan, R.: Robust timed automata. In: Maler, O. (ed.) HART 1997. LNCS, vol. 1201, pp. 331–345. Springer, Heidelberg (1997)
11. Havelund, K., Skou, A., Larsen, K.G., Lund, K.: Formal modeling and analysis of an audio/video protocol: an industrial case study using uppaal. In: RTSS 1997. IEEE Computer Society, Los Alamitos (1997)
12. Jaubert, R., Reynier, P.-A.: Quantitative robustness analysis of flat timed automata. In: Hofmann, M. (ed.) FOSSACS 2011. LNCS, vol. 6604, pp. 229–244. Springer, Heidelberg (2011)
13. Larsen, K.G., Pettersson, P., Yi, W.: Uppaal in a nutshell. Int. Journal on Software Tools for Technology Transfer 1, 134–152 (1997)
14. Puri, A.: Dynamical properties of timed systems. Discrete Event Dynamic Systems 10(1-2), 87–113 (2000)
15. Yovine, S.: Kronos: A verification tool for real-time systems. International Journal on Software Tools for Technology Transfer 1, 123–133 (1997)