ENS CACHAN

Robust Analysis of Timed Automata *via* Channel Machines

Patricia Bouyer, Nicolas Markey,
Pierre-Alain Reynier

Research report LSV-07-32

Laboratoire Spécification et Vérification

CNRS CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

École Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Robust Analysis of Timed Automata *via* Channel Machines

Patricia Bouyer[1,2,⋆], Nicolas Markey[1], Pierre-Alain Reynier[3,⋆⋆]

[1] LSV, CNRS & ENS de Cachan, France
[2] Oxford University Computing Laboratory, UK
[3] Université Libre de Bruxelles, Belgium
{bouyer,markey}@lsv.ens-cachan.fr, reynier@ulb.ac.be

Abstract. Whereas formal verification of timed systems has become a very active field of research, the idealised mathematical semantics of timed automata cannot be faithfully implemented. Several works have thus focused on a modified semantics of timed automata which ensures implementability, and robust model-checking algorithms for safety, and later LTL properties have been designed. Recently, a new approach has been proposed, which reduces (standard) model-checking of timed automata to other verification problems on channel machines. Thanks to a new encoding of the modified semantics as a network of timed systems, we propose an original combination of both approaches, and prove that robust model-checking for coFlat-MTL, a large fragment of MTL, is EXPSPACE-Complete.

1 Introduction

Verification of real-time systems. In the last thirty years, formal verification of reactive systems has become a very active field of research in computer science. It aims at checking that (the model of) a system satisfies (a formula expressing) its specifications. The importance of taking real-time constraints into account in verification has quickly been understood, and the model of timed automata [2] has become one of the most established models for real-time systems, with a well-studied underlying theory, the development of mature model-checking tools (Uppaal [20], Kronos [10], ...), and numerous success stories.

Implementation of real-time systems. Implementing mathematical models on physical machines is an important step for applying theoretical results on practical examples. This step is well-understood for many untimed models that have been studied (*e.g.*, finite automata, pushdown automata). In the timed setting, while timed automata are widely-accepted as a framework for modelling real-time aspects of systems, it is known that they cannot be faithfully implemented on finite-speed CPUs [11]. Studying the "implementability" of timed automata is thus a challenging question of obvious theoretical and practical interest.

⋆ Partly supported by a Marie Curie fellowship (European Commission).
⋆⋆ Partly supported by a Lavoisier fellowship (French Ministry of Foreign Affairs).

A semantical approach. In [15], a new semantics for timed automata is defined that takes into account the digital aspects of the hardware on which the automaton is being executed. A timed automaton is then said to be *implementable* if, under this new semantics, the behaviours of this automaton satisfies its specifications. In order to study it efficiently, this semantics is over-approximated by the *AASAP semantics*, which consists in "enlarging" the constraints on the clocks by some parameter δ. For instance, "$x \in [a, b]$" is transformed into "$x \in [a - \delta, b + \delta]$". Implementability can be ensured by establishing the existence of some positive δ for which the AASAP semantics meets the specification. The decidability of this "*robust model-checking*" problem for specifications given as LTL formula has already been solved (first basic safety properties in [13] and then full LTL [9]) using a graph algorithm on the region automaton abstraction.

Timed temporal logics. Until recently [23], linear-time timed temporal logics were mostly considered as undecidable, and only MITL, the fragment without punctuality of MTL [19], was recognised as really tractable and useful [3]. Very recently [7], another fragment of MTL, called coFlat-MTL, whose model-checking is EXPSPACE-Complete. The decidability of this logic relies on a completely original method using channel machines.

Our contribution. Inspired by the channel machine approach of [7], we propose a new techniques to robust model-checking of linear-time timed temporal logics. It is based on the construction of a network of timed systems which captures the AASAP semantics, and which can be expressed as a channel machine. Based on this approach, we prove that the robust model-checking of coFlat-MTL is EXPSPACE-Complete, *i.e.*, not more expensive than under the classical semantics. It is worth noticing that coFlat-MTL includes LTL, our result thus encompasses the previously shown decidability results in that framework.

Related work. Since its definition in [15], the approach based on the AASAP semantics has received much attention, and even other kind of perturbations like the drift of clocks, have been studied [25, 14, 4, 16]. In the case of safety properties and under some natural assumptions, this approach is equivalent to constraint enlargement and relies on similar techniques, as proved in [14]. Also, several works have proposed a symbolic, zone-based approach to the classical region-based algorithm for robustness [12, 26, 16]. This approach using the AASAP semantics contrasts with a modelling-based solution proposed in [1], where the behaviour of the platform is modelled as a timed automaton. Last, many other notions of "robustness" have been proposed in the literature in order to relax the mathematical idealisation of the semantics of timed automata [18, 22, 6, 5]. Those approaches are different from ours, since they roughly consist in dropping "isolated" or "unlikely" executions. Also note that robustness issues have also been handled in the untimed case, but are even further from our approach [17].

Outline of the paper. We introduce the setting in Section 2. Section 3 contains our construction: we first turn the robust semantics of timed automata into networks of timed systems (Section 3.1), which are then encoded as channel automata (Section 3.2). We then explain how the resulting channel automata are

used for Bounded-MTL and coFlat-MTL model-checking (Section 3.3). By lack of space, all proofs have been postponed to the appendix.

2 Preliminaries

We present here the model of timed automata, some linear-time timed temporal logics, and the model of channel automata, which is central to our approach.

2.1 Timed Automata

Let X be a finite set of *clock* variables. We denote by $\mathcal{G}(X)$ the set of *clock constraints* generated by the grammar $g ::= g \wedge g \mid \ x \sim k$, where $x \in X$, $k \in \mathbb{N}$, and $\sim \in \{\leq, \geq\}$. A *(clock) valuation* v for X is an element of $\mathbb{R}_+^X$. If $v \in \mathbb{R}_+^X$ and $t \in \mathbb{R}_+$, we write $v + t$ for the valuation assigning $v(x) + t$ to every clock $x \in X$. If $r \subseteq X$, $v[r \leftarrow 0]$ denotes the valuation assigning 0 to every clock in r and $v(x)$ to every clock in $X \setminus r$.

A *timed automaton* is a tuple $\mathcal{A} = (L, \ell_0, X, I, \Sigma, T)$ where L is a finite set of locations, $\ell_0 \in L$ is an initial location, X is a finite set of clocks, $I \colon L \to \mathcal{G}(X)$ labels each location with its invariant, Σ is a finite set of actions, and $T \subseteq L \times \mathcal{G}(X) \times \Sigma \times 2^X \times L$ is the set of transitions.

Given a parameter value $\delta \in \mathbb{R}_{\geq 0}$, whether a valuation $v \in \mathbb{R}_+^X$ satisfies a constraint g within δ, written $v \models_\delta g$, is defined inductively as follows:

$$\begin{cases} v \models_\delta x \leq k & \text{iff } \ v(x) \leq k + \delta \\ v \models_\delta x \geq k & \text{iff } \ v(x) \geq k - \delta \\ v \models_\delta g_1 \wedge g_2 & \text{iff } \ v \models_\delta g_1 \text{ and } v \models_\delta g_2 \end{cases}$$

Following [15], we define a parameterised semantics for $\mathcal{A}$ as a timed transition system $[\![\mathcal{A}]\!]_\delta = \langle S, S_0, \Sigma, \to_\delta \rangle$. The set S of states of $[\![\mathcal{A}]\!]_\delta$ is $\{(\ell, v) \in L \times \mathbb{R}_+^X \mid v \models_\delta I(\ell)\}$, with $S_0 = \{(\ell_0, v_0) \mid v_0(x) = 0 \text{ for all } x \in X\}$. A transition in $[\![\mathcal{A}]\!]_\delta$ is composed either of a delay move $(\ell, v) \xrightarrow{d}_\delta (\ell, v + d)$, with $d \in \mathbb{R}_+$, when both v and $v + d$ satisfy the invariant $I(\ell)$ within δ, or of a discrete move $(\ell, v) \xrightarrow{\sigma}_\delta (\ell', v')$ when there exists a transition $(\ell, g, \sigma, r, \ell') \in T$ with $v \models_\delta g$, $v' = v[r \leftarrow 0]$, and $v' \models_\delta I(\ell')$. The graph $[\![\mathcal{A}]\!]_\delta$ is thus an infinite transition system. Notice that, in the definitions above, the standard semantics of timed automata can be recovered by letting $\delta = 0$. In that case, we omit the subscript δ.

A *run* of $[\![\mathcal{A}]\!]_\delta$ is an infinite sequence $(\ell_0, v_0) \xrightarrow{d_0}_\delta (\ell_0, v_0 + d_0) \xrightarrow{\sigma_0}_\delta (\ell_1, v_1) \xrightarrow{d_1}_\delta (\ell_1, v_1 + d_1) \dots$ where for each $i \geq 0$, $d_i \in \mathbb{R}_+$. A *timed word* w is an infinite sequence $(\sigma_i, t_i)_{i \in \mathbb{N}}$ where $\sigma_i \in \Sigma$ and $t_i \in \mathbb{R}_+$ for each $i \geq 0$, and such that the sequence $(t_i)_{i \in \mathbb{N}}$ is non-decreasing and diverges to infinity. The timed word w is read on the run $(\ell_0, v_0) \xrightarrow{d_0}_\delta (\ell_0, v_0 + d_0) \xrightarrow{\sigma_0}_\delta (\ell_1, v_1) \xrightarrow{d_1}_\delta (\ell_1, v_1 + d_1) \dots$ whenever $t_i = \sum_{j \leq i} d_j$ for every $i \in \mathbb{N}$. We write $\mathcal{L}([\![\mathcal{A}]\!]_\delta)$ for the set of timed words that can be read on a run of $[\![\mathcal{A}]\!]_\delta$ starting in (ℓ_0, v_0). More generally, we write $\mathcal{L}([\![\mathcal{A}]\!]_\delta^{(\ell, v)})$ for the set of timed words than can be read starting from (ℓ, v).

Since our results rely on the results of [13, 9], we require that our timed automata satisfy the following requirements: *(i)* constraints in guards and invariants only involve non-strict inequalities; *(ii)* all the clocks are always bounded by some constant M; *(iii)* all the cycles in the region graph are *progress cycles*, *i.e.*, all the clocks are reset along those cycles. In addition, we require that the timed automata are *non-blocking*, in the sense that from every state, an action transition will eventually become firable: for every $(\ell, v) \in L \times \mathbb{R}_+^X$ such that $v \models_0 I(\ell)$, there exists $d \in \mathbb{R}_+^X$ and $\sigma \in \Sigma$ such that $(\ell, v) \xrightarrow{d}_0 (\ell, v+d) \xrightarrow{\sigma}_0 (\ell', v')$ for some $(\ell', v') \in L \times \mathbb{R}_+^X$.

2.2 Implementability and Robustness of Timed Automata

The parameterised semantics defined above (referred to as "enlarged semantics" in the sequel), has been defined in [15] in order to study the *implementability* of timed systems. Indeed, timed automata are governed by a mathematical, idealised semantics, which does not fit with the digital, imprecise nature of the hardware on which they will possibly be implemented. An *implementation semantics* has thus been defined in [15] in order to take the hardware into account: that semantics models a digital CPU which, every δ_P time units (at most), reads the value of the digital clock (updated every δ_L time units), computes the values of the guards, and fires one of the available transitions. We write $[\![\mathcal{A}]\!]^{\delta_P,\delta_L}$ for the resulting transition system, and $\mathcal{L}([\![\mathcal{A}]\!]^{\delta_P,\delta_L})$ for the corresponding set of timed words. Given a (linear-time) property $\mathcal{P}$, *i.e.*, a set of accepted timed words, a timed automaton $\mathcal{A}$ is said to be implementable w.r.t. $\mathcal{P}$ iff $\mathcal{L}([\![\mathcal{A}]\!]^{\delta_P,\delta_L}) \subseteq \mathcal{P}$ for some positive values of δ_P and δ_L.

As proved in [15], the enlarged semantics simulates the implementation semantics as soon as $\delta > 4\delta_P + 3\delta_L$. As a consequence, it is sufficient to check the existence of $\delta > 0$ such that $\mathcal{L}([\![\mathcal{A}]\!]_\delta) \subseteq \mathcal{P}$ in order to ensure implementability of $\mathcal{A}$ w.r.t. $\mathcal{P}$. We follow this idea in the sequel, and study the *robust satisfaction* relation: a timed automaton *robustly satisfies* a linear-time property $\mathcal{P}$, written $\mathcal{A} \models\!\!\models \mathcal{P}$, whenever $\mathcal{L}([\![\mathcal{A}]\!]_\delta) \subseteq \mathcal{P}$ for some $\delta > 0$.

2.3 Some Subclasses of Metric Temporal Logic

Linear-time properties are often defined *via* temporal logic formulae. In this paper, we focus on subclasses of MTL (Metric Temporal Logic) [19].

Fix a finite, non-empty alphabet Σ. The syntax of MTL over Σ is defined by the following grammar:

$$\mathsf{MTL} \ni \varphi ::= \sigma \mid \neg\sigma \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \,\mathbf{U}_I\, \varphi \mid \varphi \,\widetilde{\mathbf{U}}_I\, \varphi$$

where $\sigma \in \Sigma$, and I is an interval of $\mathbb{R}^+$ with bounds in $\mathbb{N} \cup \{\infty\}$. MTL formulae are interpreted over timed words. Let $w = (\sigma_i, t_i)_{i \geq 0}$ be a timed word, and $p \in \mathbb{N}$. The (pointwise) semantics of MTL is defined recursively as follows (we omit

Boolean operations):

$$\begin{aligned} w,p \models \sigma &\;\Leftrightarrow\; \sigma_p = \sigma \\ w,p \models \varphi \,\mathbf{U}_I\, \psi &\;\Leftrightarrow\; \exists i > 0 \text{ s.t. } w,p+i \models \psi,\ t_{p+i} - t_p \in I \\ &\qquad\quad \text{and } \ \forall 0 < j < i,\ w,p+j \models \varphi \\ w,p \models \varphi \,\widetilde{\mathbf{U}}_I\, \psi &\;\Leftrightarrow\; w,p \models \neg\Big((\neg\varphi)\,\mathbf{U}_I\,(\neg\psi)\Big). \end{aligned}$$

If $w,0 \models \varphi$, we write $w \models \varphi$. Following the discussion above, we write $\mathcal{A} \models\!\models \varphi$ if, for some $\delta > 0$, we have $w \models \varphi$ for every $w \in \mathcal{L}([\![\mathcal{A}]\!]_\delta)$.

Additional operators, such as $\mathtt{tt}$ (true), $\mathtt{ff}$ (false), $\Rightarrow$, $\Leftrightarrow$, $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{X}$, are defined in the usual way: $\mathbf{F}_I\,\varphi \equiv \mathtt{tt}\,\mathbf{U}_I\,\varphi$, $\mathbf{G}_I\,\varphi \equiv \mathtt{ff}\,\widetilde{\mathbf{U}}_I\,\varphi$, and $\mathbf{X}_I\,\varphi \equiv \mathtt{ff}\,\mathbf{U}_I\,\varphi$. We also use pseudo-arithmetic expressions to denote intervals. For example, '$=1$' denotes the singleton $\{1\}$.

Following [7], we identify the following syntactic fragments of MTL: LTL [24] can be considered as the fragment of MTL in which modalities are not constrained (*i.e.*, where $\mathbb{R}_+$ is the only constraining interval); Bounded-MTL is the fragment of MTL in which all interval constraints are bounded: observe that Bounded-MTL disallows unconstrained modalities, and is in particular not suitable to express invariance properties (the most basic type of temporal specifications). The fragment coFlat-MTL[4] has then been defined to remedy this deficiency:

$$\text{coFlat-MTL} \ni \varphi ::= \sigma \mid \neg\sigma \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi\,\mathbf{U}_J\,\varphi \mid \varphi\,\mathbf{U}_I\,\underline{\psi} \mid \varphi\,\widetilde{\mathbf{U}}_J\,\varphi \mid \underline{\psi}\,\widetilde{\mathbf{U}}_I\,\varphi$$

where J ranges over the set of bounded intervals, I over the set of all intervals, and the underlined formula $\underline{\psi}$ ranges over LTL.

One immediately sees that coFlat-MTL subsumes both LTL and Bounded-MTL, but it is not closed under negation. However, it is closed under invariance, and can then express *e.g.* bounded response-time or even richer formulae such as $\mathbf{G}\,(\mathit{request} \Rightarrow \mathbf{F}_{\leq 5}\,(\mathit{acquire} \wedge \mathbf{F}_{=1}\,\mathit{release}))$.

2.4 Channel Automata

Channel automata are an interesting formalism for reasoning about alternating timed automata, which has been used in [7] to prove the EXPSPACE-membership of the model-checking problem for coFlat-MTL. We only give the definition and necessary results here, and refer to Appendix A for some more intuition.

A *channel automaton with renaming and occurrence testing* (CAROT for short) is a tuple $\mathcal{C} = (S, s_0, \Sigma, \delta, F)$, where S is a finite set of control states, $s_0 \in S$ is the initial control state, $F \subseteq S$ is a set of accepting control states, Σ is a finite alphabet, $\delta \subseteq S \times \mathit{Op} \times S$ is the set of transition rules, where

$$\mathit{Op} = \{\sigma!, \sigma? \mid \sigma \in \Sigma\} \cup \{\mathit{zero}?(\sigma) \mid \sigma \in \Sigma\} \cup \{R \mid R \subseteq \Sigma \times \Sigma\}$$

is the set of operations. Given a rule $\tau = (s, \alpha, s') \in \delta$, we define $\mathit{op}(\tau) = \alpha$.

[4] We do not explain this terminology here, and refer to [7] for deeper considerations.

Intuitively, operations $\sigma!$ and $\sigma?$ are the classical write and read operations, $zero?(\sigma)$ is a guard for testing the occurrence of σ in the channel, and $R \subseteq \Sigma \times \Sigma$ is interpreted as a global renaming. An *end-of-channel* marker can be used to count the number of times the whole channel is read: it suffices to add, from each state of the CAROT, an outgoing transition reading $\triangleright$, immediately followed by a transition writing $\triangleright$. That way, there is always a unique copy of $\triangleright$ on the channel (except when it has just been read). The number of transitions writing $\triangleright$ along a computation ϱ is the number of *cycles* of ϱ, denoted by $cycles(\varrho)$. In the sequel, the CAROTs are assumed to contain the end-of-channel marker $\triangleright$, and to have all their states equipped with a loop reading and writing that symbol.

We consider in the sequel a restricted version of the reachability problem for CAROTs, where we impose a bound on the "time" (measured here as the number of cycles of the channel): the *cycle-bounded reachability problem* for CAROTs is defined as follows: given a CAROT $\mathcal{C}$, an input word $w \in \Sigma^*$, and a cycle bound h, does $\mathcal{C}$ have an accepting computation ϱ on w with $cycles(\varrho) \leqslant h$?

In [7], a non-deterministic procedure is presented to check the existence of an accepting cycle-bounded computation using only polynomial space in the *value* of the cycle bound and in the size of the CAROT:

Theorem 1. *The cycle-bounded reachability problem for CAROTs is solvable in polynomial space in the size of the channel automaton, the size of the input word, and the value of the cycle bound.*

Remark. Note that the algorithm could easily be adapted to cope with the cycle-bounded reachability between two global states (*i.e.*, control state and content of the channel): it suffices to first set the initial content of the channel, and to add transitions that would, from any point, run one more cycle of the channel and store its whole content in the location.

3 Robust Model-Checking of coFlat-MTL

In this section, we prove the main results of this paper, namely that the robust model-checking problem can be expressed using CAROTs, and then that the robust model-checking problem of coFlat-MTL is EXPSPACE-Complete. EXPSPACE-Hardness is a consequence of the EXPSPACE-Hardness of the satisfiability problem for Bounded-MTL [7]. EXPSPACE-membership is more involved and will be done in several steps:

1. We first construct a family of networks of timed systems that captures the enlarged semantics of timed automata (Subsection 3.1).
2. We then transform this family of networks into a CAROT, that will simulate the joint behaviours of those networks of timed systems with an alternating timed automaton representing the property we want to robustly verify (Subsection 3.2).
3. Finally, we use the CAROT to design a decidability algorithm for the robust model-checking problem, first for Bounded-MTL, and then for coFlat-MTL (Subsection 3.3).

For the rest of the paper, we fix a timed automaton $\mathcal{A} = (L, \ell_0, X, I, \Sigma, T)$.

3.1 From Robustness to Networks of Timed Systems

In this subsection, we transform the robust model-checking problem into a model-checking problem in an infinite family of timed systems. The correctness of the transformation relies on simulation relations between timed transition systems: given two timed transition systems $\mathcal{T}_i = (S_i, \Sigma, \to_i)$ for $i = 1, 2$, we say that $\mathcal{T}_2$ *simulates* $\mathcal{T}_1$ whenever there exists a relation $\mathfrak{R} \subseteq S_1 \times S_2$ such that $(s_1, s_2) \in \mathfrak{R}$ and $s_1 \xrightarrow{\alpha}_1 s_1'$ with $\alpha \in \Sigma \cup \mathbb{R}_+$ implies $s_2 \xrightarrow{\alpha}_2 s_2'$ for some $s_2' \in S_2$ with $(s_1', s_2') \in \mathfrak{R}$. We then write $\mathcal{T}_1 \sqsubseteq \mathcal{T}_2$.

Let n be an integer; we denote by $\mathcal{B}^n$ the timed network composed of the following components (which are not standard timed automata because of the use of disjunction and fractional parts in the guards):

• for each $0 \leq i < n$, $\mathcal{B}_i$ is the timed automaton depicted on Fig. 1, where x_i is a fresh clock not belonging to X, and $[x_i \leq 1]$ is an invariant forbidding the clock x_i become larger than 1. That way, this automaton is forced to fire its transition when the value of x_i reaches 1. We call such an automaton a *Δ-automaton* in the sequel. In the following, we will take indices of clocks x_i modulo n, and in particular $x_{i+1} = x_0$ whenever $i = n - 1$.

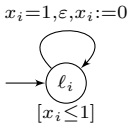

Fig. 1. Automaton $\mathcal{B}_i$

• the timed automaton $\mathcal{B}$, built from $\mathcal{A}$ by modifying the guards and invariants as follows: each constraint of the form $x \leq k$ is replaced with

$$(x < k+1) \ \wedge \Big(x > k \ \Rightarrow \bigvee_{0 \leq i < n} \{x\} \leq x_{i+1} < x_{i-1}\Big)$$

and each constraint of the form $x \geq k$ is replaced with

$$(x > k-1) \ \wedge \Big(x < k \ \Rightarrow \bigvee_{0 \leq i < n} \{x\} \geq x_{i-1} > x_{i+1}\Big)$$

We need to explain when a valuation v satisfies these "extended" guards. Boolean operators are handled in the natural way, and $\{x\}$ is intended to denote the fractional part of the value of clock x.

It naturally defines a timed transition system $[\![\mathcal{B}^n]\!]$, as the synchronised product (synchronised because of the time) of all components. Denoting by $X_n = X \cup \{x_i \mid 0 \leq i < n\}$ the set of clocks of $\mathcal{B}^n$, a configuration of $[\![\mathcal{B}^n]\!]$ can be described by a pair (ℓ, v) where $\ell \in L$ and $v \in \mathbb{R}_+^{X_n}$ (each Δ-automaton has a single location). We write u_n for the valuation over X_n assigning 0 to clocks in X and $\frac{i}{n}$ to every clock x_i for $0 \leq i < n$. The initial configuration of this timed network is the pair (ℓ_0, u_n). Delay and action transitions are defined naturally in the synchronised products of all the components. However in the following we will hide ε-moves due to the components $\mathcal{B}_i$. Thus, in $[\![\mathcal{B}^n]\!]$, we write

$(\ell, v) \stackrel{t}{\Rightarrow} (\ell, v')$ for an interleaving of delay transitions (which sum up to t) and of ε-moves in the $\mathcal{B}_i$'s. For uniformity, we write $\stackrel{\sigma}{\Rightarrow}$ for σ-moves in $[\![\mathcal{B}^n]\!]$. In the sequel, simulation relations assume the relation $\Rightarrow$ in $[\![\mathcal{B}^n]\!]$, and the simple transition relation $\rightarrow_\delta$ in $[\![\mathcal{A}]\!]_\delta$. In the same way, the intended language accepted by $[\![\mathcal{B}^n]\!]$ should ignore ε-transitions. In other words, it should also be defined using the relation $\Rightarrow$ as the transition relation:

$$\mathcal{L}([\![\mathcal{B}^n]\!]) = \{w = (\sigma_i, t_i)_{i\in\mathbb{N}} \mid \exists (\ell_0, u_n) \stackrel{d_0}{\Longrightarrow} (\ell_0, u') \stackrel{\sigma_0}{\Longrightarrow} (\ell_1, u'') \stackrel{d_1}{\Longrightarrow} \cdots \in [\![\mathcal{B}^n]\!] \\ \text{s.t. } \forall i \in \mathbb{N},\ t_i = \sum_{j\le i} d_j\}$$

Lemma 2. *For every $n \geq 3$, $[\![\mathcal{A}]\!]_{\frac{1}{n}} \sqsubseteq [\![\mathcal{B}^n]\!] \sqsubseteq [\![\mathcal{A}]\!]_{\frac{2}{n}}$.*

With the previous definition, the simulation results can be stated in terms of language inclusion (as initial states are preserved by the exhibited simulation relations): for every $n \geq 3$, $\mathcal{L}([\![\mathcal{A}]\!]_{\frac{1}{n}}) \subseteq \mathcal{L}([\![\mathcal{B}^n]\!]) \subseteq \mathcal{L}([\![\mathcal{A}]\!]_{\frac{2}{n}})$.

Theorem 3. *Let $\varphi \in \mathsf{MTL}$. Then, $\mathcal{A} \models \varphi \ \Leftrightarrow\ \exists n \geq 3$ s.t. $[\![\mathcal{B}^n]\!] \models \varphi$.*

3.2 From Networks of Timed Systems to CAROTs

Extending the approach of [7], the CAROT we build is such that it accepts joint executions of the network of timed systems we just built (and not of a single timed automaton as in [7]) and of the alternating timed automaton corresponding to the negation of the coFlat-MTL formula we want to verify. In order to handle arbitrarily many components in the network, and to deal with "extended" guards (*i.e.*, with disjunctions and fractional parts), the construction attached to the network of timed systems needs to be deeply modified. In a first step, we describe a CAROT that only encodes the behaviours of the network of timed systems.

Let M be the maximal constant appearing in the automaton $\mathcal{A}$. Then $M+1$ is larger than or equal to the maximal constant of any $\mathcal{B}^n$. We assume that the clocks of $\mathcal{A}$ (and $\mathcal{B}^n$) take their values in $[0, M+1] \cup \{\bot\}$, where $\bot$ is a special value (intended to represent any value larger than $M+1$). We write Reg for the set $\{0, 1, \ldots, M+1, \bot\}$ and $\Lambda = \wp(L \times X \times \mathsf{Reg})$.

A configuration $(\ell, v) \in L \times \mathbb{R}_+^{X_n}$ of the network of timed systems is encoded as the element $C_{(\ell,v)} = \{(\ell, x, v(x)) \mid x \in X_n\}$, and partitioned into a sequence of disjoint subsets C_0, C_1, ..., C_p, $C_\bot$, obtained using standard region techniques. More precisely, C_0 (resp. $C_\bot$) contains elements whose fractional part is 0 (resp. whose value is $\bot$) and the others C_i gather elements with the same fractional part and are sorted according to the increasing order of fractional parts. We then let $H(C_{(\ell,v)}) = \mathsf{reg}(C_0)\mathsf{reg}(C_1)\mathsf{reg}(C_2)\ldots\mathsf{reg}(C_p)\mathsf{reg}(C_\bot) \in \Lambda^*$, where we write $\mathsf{reg}(C)$ for $\{(\ell, x, \mathsf{reg}(v)) \mid (\ell, x, v) \in C\}$, with $\mathsf{reg}(v)$ the largest element of Reg smaller than or equal to v.

Using the abstraction function H, it is possible to define a discrete transition system $\mathcal{T}_H^n$ which abstracts away precise timing information, but which simulates

the behaviours of $\mathcal{B}^n$. The abstraction function also provides an equivalence relation $\equiv$ on configurations: $C \equiv C'$ iff $H(C) = H(C')$. Extending straightforwardly a result of [21] (regions are compatible with our extended guards), we have:

Lemma 4. *The equivalence relation $\equiv$ is a time-abstract[5] bisimulation.*

The CAROT we will build is based on the above discrete transition system. More precisely, the intended encoding of a configuration in the CAROT is the following: the integral values of the clocks of $\mathcal{A}$ are stored in the discrete state of the CAROT, as well as the set C_0 and the current location ℓ. The other C_i's are stored on the channel, from C_1 (recently written on the channel) to C_p (the next item to be read from the channel).

In order to use the same CAROT to simulate the network of timed system with any number of Δ-automata, we abstract the name of clocks x_i in our encoding, representing them on the channel by a new symbol Δ. Since, at any time, at most one of the x_i's can have integral value, and since we only need to know the order of the x_i's in order to evaluate guards of $\mathcal{B}^n$, the amount of information to be stored in the location of the CAROT does not depend on the number of Δ-automata in the network.

Example 5. Consider for instance a configuration C encoded by the word

$$H(C) = \{(\ell, x, 3), (\ell, x_2, 0)\} \cdot \{(\ell, x_0, 0)\} \cdot \{(\ell, y, 1), (\ell, x_1, 0)\} \cdot \{(\ell, z, \perp)\}.$$

We assume that the maximal constant M is 3. The encoding of the delay-successor of C is obtained by cycling around the letters (except the last one) of the word (and increasing the values of the regions accordingly). Thus the first delay successor of $H(C)$ is

$$\varnothing \cdot \{(\ell, x, 3), (\ell, x_2, 0)\} \cdot \{(\ell, x_0, 0)\} \cdot \{(\ell, y, 1), (\ell, x_1, 0)\} \cdot \{(\ell, z, \perp)\}.$$

The next delay successor is

$$\{(\ell, y, 2), (\ell, x_1, 0)\} \cdot \{(\ell, x, 3), (\ell, x_2, 0)\} \cdot \{(\ell, x_0, 0)\} \cdot \{(\ell, z, \perp)\}.$$

Note that, in that second delay transition, we have reset clock x_1 when it has reached 1, and the integral part of y has increased to 2. The configuration $H(C)$ would be encoded as depicted on Fig. 2 (where we write to the left and read from the right of the channel). With respect to the channel, the first delay transition is performed through the write operation '$\langle x\Delta\rangle!$'. As in [7], it is worth noticing that a cycle of the CAROT corresponds to one time unit elapsing.

Furthermore, to simulate the extended guards used in $\mathcal{B}^n$, we need some additional information about the position of clocks of $\mathcal{A}$ w.r.t. symbols Δ. As we have already seen, a clock x verifies a constraint $\{x\} \leq x_{i+1} < x_{i-1}$ iff its fractional part is smaller than one of the two smallest clocks x_j. In our simulation, this corresponds as being "before" the second symbol Δ on the channel. And

[5] This means that precise delays of time-elapsing transitions are abstracted away.

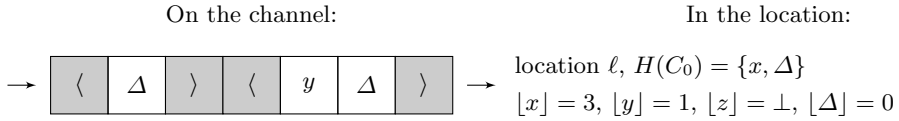

Fig. 2. Encoding of a configuration in a CAROT

symmetrically for constraint $\{x\} \geq x_{i-1} > x_{i+1}$. We thus have to store in the control part of the CAROT which clocks are "before" (resp. "after") the two first (resp. last) symbols Δ. Whereas this can easily be done for the clocks that have been recently written on the channel, this is not possible for the clocks lying at the tail of the channel (this would require to store the position of each clock w.r.t. each symbol Δ). Instead, we use non-determinism to allow the CAROT make predictions about the content of the "tail" of the channel, and then we verify when reading clocks from the channel that these predictions were correct.

For lack of space, we cannot present the formal construction of the CAROT, but report to the Appendix C. We write $\mathcal{C}_{\mathcal{A}}$ for the resulting CAROT, $\mathcal{T}^n_{\mathcal{C}_{\mathcal{A}}}$ for the transition system associated with $\mathcal{C}_{\mathcal{A}}$ and restricted to configurations with correct predictions and n occurrences of Δ on the channel (or in the location), and $\approx$ for the relation that describes which configuration (d, c) of $\mathcal{T}^n_{\mathcal{C}_{\mathcal{A}}}$ (a control state together with a channel content) corresponds to a configuration of $\mathcal{T}^n_H$. The correctness of the construction relies on the following lemmas, whose proofs and many other technical intermediary results are given in Appendix D:

Lemma 6. *For any $n \geq 3$, the transition systems $\mathcal{T}^n_H$ and $\mathcal{T}^n_{\mathcal{C}_{\mathcal{A}}}$ are bisimilar.*

Lemma 7. *Let ϱ be a time-divergent execution in $[\![\mathcal{B}^n]\!]$. Then any computation in $\mathcal{C}_{\mathcal{A}}$ simulating ϱ has correct predictions.*[6]

Let (d^n, c^n) be the configuration of $\mathcal{C}_{\mathcal{A}}$ encoding the initial configuration of $\mathcal{B}^n$, *i.e.*, such that $(d^n, c^n) \approx H(\ell_0, u_n)$. Lemmas 4, 6 and 7 then yield:

Theorem 8. *Let $n \geq 3$. $\mathcal{C}_{\mathcal{A}}$ has a time-divergent*[7] *computation starting in (d^n, c^n) iff $\mathcal{L}([\![\mathcal{B}^n]\!]) \neq \varnothing$.*

The second part of the construction of the CAROT for encoding the robust model-checking problem consists in adding the part related to the temporal formula φ (in MTL). This part will be handled in a very similar way as in [7], we thus simply sketch the construction. First, we build the one-clock alternating timed automaton $\mathcal{A}_{\neg\varphi}$ corresponding to $\neg\varphi$. Then, we build the product of the CAROT $\mathcal{C}_{\mathcal{A}}$ with a CAROT simulating the behaviour of $\mathcal{A}_{\neg\varphi}$. The resulting CAROT, say $\mathcal{C}_{\mathcal{A},\neg\varphi}$, running from the initial configuration $(d^{\varphi,n}, c^{\varphi,n})$ corresponding via $\approx$ to the initial configuration of $\mathcal{B}^n \times \mathcal{A}_{\neg\varphi}$, simulates joint behaviours of $\mathcal{B}^n$ and $\mathcal{A}_{\neg\varphi}$. The accepting condition for $\mathcal{C}_{\mathcal{A},\neg\varphi}$ is the Büchi condition given by 'flattening' $\mathcal{A}_{\neg\varphi}$. The results of [7] combined with our above results yield:

[6] Intuitively, this is because delay transitions force predictions checking.
[7] That is with infinitely many delay transitions.

Theorem 9. *Let $n \geq 3$ and $\varphi \in \mathsf{MTL}$. $\mathcal{C}_{\mathcal{A},\neg\varphi}$ has a time-divergent accepting computation starting in $(d^{\varphi,n}, c^{\varphi,n})$ iff $[\![\mathcal{B}^n]\!] \not\models \varphi$.*

Remark. A rough bound on the size of the CAROT $\mathcal{C}_{\mathcal{A},\neg\varphi}$ is $O\big(|\mathcal{A}|^3 \cdot 2^{O(M\cdot|\varphi|+|X|)}\big)$, where $|\varphi|$ is the number of subformulae of φ. The size of the alphabet of $\mathcal{C}_{\mathcal{A},\neg\varphi}$ is in $O(|X|)$. As proved in [7], if $\mathcal{C}_{\mathcal{A},\neg\varphi}$ has an h-cycle-bounded accepting execution, then there is a bound N_0, which depends on the size of $\mathcal{C}_{\mathcal{A},\neg\varphi}$ and h, such that there exists an h-cycle-bounded accepting execution of length no more than N_0. We do not give the precise value of this bound (see Appendix A instead), but it is exponential in h, which is itself exponential in the size of the input.

3.3 From CAROTs to Robust Model Checking

We first solve the robust model-checking problem for Bounded-MTL, and then turn to the more involved logic coFlat-MTL. Both rely on the previously proved equivalences:

$$\mathcal{A} \not\models \varphi \quad \text{iff} \quad \forall n \geq 3,\ \begin{cases} \mathcal{C}_{\mathcal{A},\neg\varphi} \text{ has a time-divergent accepting} \\ \text{computation starting in } (d^{\varphi,n}, c^{\varphi,n}) \end{cases}$$

Robust model-checking for Bounded-MTL. The algorithm to decide the robust model-checking problem for Bounded-MTL formula relies on the fact that the truth value of a Bounded-MTL formula φ along a run ϱ only depends on the first h time units of ϱ, where h is the sum of the constants appearing in φ [7]. In $\mathcal{A}_{\neg\varphi}$, after having read a prefix of duration h time units, we thus always end up in a sink state, that we report as accepting in the CAROT.

Moreover, the non-blocking assumption made on $\mathcal{A}$ implies the following property:

Lemma 10. *Let $\mathcal{A}$ be a timed automaton, and $\delta > 0$. Given (ℓ, v) a configuration of $\mathcal{A}$ such that $v \models_\delta I(\ell)$, we have that $\mathcal{L}([\![\mathcal{A}]\!]_\delta^{(\ell,v)}) \neq \varnothing$.*

Hence, robustly model-checking $\mathcal{A}$ against a Bounded-MTL property φ will be reduced to searching, for every $n \geq 3$, for a time-bounded accepting prefix in $\mathcal{T}^n_{\mathcal{C}_{\mathcal{A},\neg\varphi}}$, and verifying that the reachable configuration is with correct predictions. Indeed, applying the previous lemma, we already know that we will be able to extend this finite prefix into a time-diverging run witnessing $\neg\varphi$ as soon as the prefix is correctly chosen (meaning it ends up in a accepting state of the CAROT).

We define the following property, for any integer n:

$$\mathcal{P}(n)\text{: } \text{“}\mathcal{C}_{\mathcal{A},\neg\varphi} \text{ has an } h\text{-cycle-bounded accepting computation with correct predictions starting in } (d^{\varphi,n}, c^{\varphi,n})\text{”}$$

Then our problem somehow amounts to checking that for every $n \geq 3$, property $\mathcal{P}(n)$ holds. This is some kind of “universality” checking of the CAROT, where we universally quantify on the initial number of Δ's on the channel. This is achieved using the following two lemmas:

Lemma 11. *Let $n, n' \in \mathbb{N}$ be such that $n' \geq 2n \geq 6$. Then $\mathcal{P}(n') \Rightarrow \mathcal{P}(n)$.*

Proof. We have seen that

$$[\![\mathcal{B}^{n'}]\!] \underset{\text{(Lemma 2)}}{\sqsupseteq} [\![\mathcal{A}_{\frac{2}{n'}}]\!] \underset{(\frac{2}{n'} \leq \frac{1}{n})}{\sqsupseteq} [\![\mathcal{A}_{\frac{1}{n}}]\!] \underset{\text{(Lemma 2)}}{\sqsupseteq} [\![\mathcal{B}^{n}]\!].$$

Also, for $m \geq 3$, the CAROT $\mathcal{C}_{\mathcal{A},\neg\varphi}$, when restricted to configurations with correct predictions, is time-abstract bisimilar to the product of $\mathcal{B}^m$ with $\mathcal{A}_{\neg\varphi}$. Finally, the respective initial configurations are in the relation $\approx$. □

Lemma 12. *Let $N \geq 2 \cdot N_0$. Then $\mathcal{P}(N) \Rightarrow \forall n \in \mathbb{N}, \exists n' \geq n$ s.t. $\mathcal{P}(n')$.*

Proof (Sketch). Using the notion of computation table introduced in [7] and presented in Appendix A, we prove a pumping lemma for CAROTs. Indeed, the height of the computation table is bounded by h; the number of "sliding windows" of such tables is thus bounded. Hence, once the table is large enough, it is possible to duplicate one of its fragments, building new computation tables encoding computations over larger inputs (corresponding to configurations $(d^{\varphi,n'}, c^{\varphi,n'})$ for integers n' arbitrarily larger than n). □

Thanks to those lemmas, it suffices to only look for an h-cycle-bounded execution starting in one of the configurations $(d^{\varphi,N}, c^{\varphi,N})$ (for any $N \geq 2 \cdot N_0$) in order to ensure the existence of an accepting execution for any number of Δ's:

Corollary 13. *Let $N \geq 2 \cdot N_0$. Then $\mathcal{P}(N) \Leftrightarrow \forall n \geq 3,\ \mathcal{P}(n)$.*

Theorem 14. *The model-checking problem for* Bounded-MTL *is* EXPSPACE-*Complete (and* PSPACE-*Complete if constants of the formula are given in unary).*

Proof. The hardness parts follow from the same hardness results for Bounded-MTL satisfiability [7].

The upper bound follows from the previous study. However, since the size of the CAROT is doubly exponential, the non-deterministic algorithm of Theorem 1 has to be applied on-the-fly, without explicitly building the CAROT. Since the number N_0 of different sliding windows of height h is also doubly-exponential in the size of the input, our non-deterministic algorithm will also have a counter, and will stop as soon as the counter reaches N_0. This all can be achieved within exponential space.

If the constants of the formula are unary-encoded, then h is linear in the size of the input formula, and N_0 is simply exponential in the size of the input. The same algorithm then uses only polynomial space. □

Robust model-checking for coFlat-MTL. The case of coFlat-MTL is more involved than that of Bounded-MTL. The reason is that, unlike Bounded-MTL, the truth value of a coFlat-MTL formula φ along a run ϱ does not only depend on a prefix of ϱ of bounded duration. Instead, we have the following decomposition lemma, which follows from [7, Theorem 12]:

Lemma 15. *Let ϖ be an accepting run of $[\![\mathcal{A}]\!]_\delta \times [\![\mathcal{A}_{\neg\varphi}]\!]$. Then it can be decomposed as $\varpi_1 \cdot \varpi_2 \cdot \varpi_3 \cdots \varpi_{2m}$ where there is a finite automaton $\mathcal{F}_{\neg\varphi}$ [8] such that:*

- *(i) the duration of ϖ_{2i-1} (for $1 \leq i \leq m$) is bounded by $h = (2M + 3 + W \cdot 2^{|\varphi|}) \cdot (|\varphi| \cdot 2^{|\varphi|})$,*
- *(ii) the duration of ϖ_{2i} (for $1 \leq i \leq m$) is at least $2^{|\varphi|} \cdot (2W + 1)$, and along that portion, the behaviour of $\mathcal{A}_{\neg\varphi}$ is that of $\mathcal{F}_{\neg\varphi}$,*
- *(iii) the Büchi condition of $\mathcal{F}_{\neg\varphi}$ is satisfied along ϖ_{2m}, and*
- *(iv) $2m \leq |\varphi| \cdot 2^{|\varphi|}$,*

where W is the number of states of the region automaton of $\mathcal{A} \times \mathcal{F}_{\neg\varphi}$. Odd-numbered segments are called the active parts*, while even-numbered ones are said* inactive*.*

This decomposition lemma inspires a decidability algorithm, where we modularly check the existence of runs not satisfying φ by distinguishing between active and inactive parts of the runs. Indeed, given a sequence $(\varrho^\delta)_{\delta>0}$, using combinatorics arguments, it is possible to twist them so that, for $\delta > 0$ small enough (say $\delta \leq \delta_0$), all ϱ^δ look very similar (that is, roughly the junction points between active and inactive parts are close to each other and belong to the same region). Conversely, if we are given a witnessing run ϱ^{δ_0}, and if we consider the junction points of that run, for each inactive (resp. active) part, it is possible for every $\delta > 0$ to build an inactive (resp. active) portion of a run joining the two junction points. The construction of an inactive portion of a run partly relies on approximation results proved in [14, 8], and the construction of an active portion of a run relies on a result similar to Corollary 13. The complete proof is rather technical and gathers in an original way many results of [14, 8, 7], we report it in Appendix E.

An informal version of the decidability algorithm is the following, and can be schematised as on Figure 3:

- Guess the junction points of the active and inactive parts
- For each active part, check that the two guessed junction points are reachable in $\mathcal{C}_{\mathcal{A},\neg\varphi}$ in a cycle-bounded manner[9] (here, we need to prove a result similar to Corollary 13)
- For each inactive and bounded active part, check that the two junction points are "robustly" reachable (using results of [14, 8])
- For the last unbounded active part, check that the automaton $\mathcal{A} \times \mathcal{F}_{\neg\varphi}$, from last junction point, does not robustly satisfy the acceptance condition of $\mathcal{F}_{\neg\varphi}$ interpreted as a co-Büchi condition (using the algorithm of [9]).

We can conclude with the main result of the paper:

Theorem 16. *The robust model-checking problem for* coFlat-MTL *is* EXPSPACE-*Complete.*

[8] Intuitively, $\mathcal{F}_{\neg\varphi}$ is obtained as the flattening of the untimed part of $\mathcal{A}_{\neg\varphi}$, see Appendix E.

[9] The bound on the number of cycles is that of (i) in the above decomposition lemma.

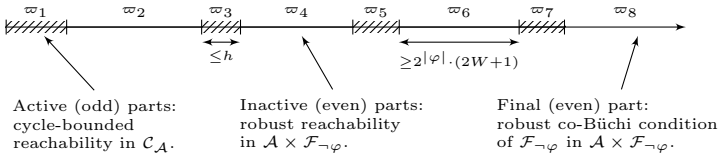

Fig. 3. Global view of our algorithm

4 Conclusion

In this paper, we have proposed a new approach to robust model-checking of timed systems based on channel machines: we construct a family of networks of timed systems such that robustly verifying a formula in a timed automaton reduces to the verification of the formula in one of the members of the family; Then we encode the behaviour of this family of timed systems using channel machines. We have applied this approach to coFlat-MTL, a rather expressive fragment of MTL, and prove that it can be decided in EXPSPACE, which is moreover optimal. The logic coFlat-MTL subsumes LTL, thus it is the more general specification language for which robust model-checking has been proved decidable.

Our correctness proofs heavily rely on technical lemmas proved in [14, 8] and is unfortunately not fully CAROT-based. As future works, we plan to study robust reachability directly on the CAROT encoding the extended semantics, in order to develop a fully CAROT-based algorithm for coFlat-MTL.

References

1. K. Altisen and S. Tripakis. Implementation of timed automata: An issue of semantics or modeling? In *Proc. 3rd Intl Conf. Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, LNCS 3829, pages 273–288. Springer, 2005.
2. R. Alur and D. Dill. A theory of timed automata. *Theor. Comp. Sci.*, 126(2):183–235, 1994.
3. R. Alur, T. Feder, and Th. A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.
4. R. Alur, S. La Torre, and P. Madhusudan. Perturbed timed automata. In *Proc. 8th Intl Workshop Hybrid Systems: Computation & Control (HSCC'05)*, LNCS 3414, pages 70–85. Springer, 2005.
5. C. Baier, N. Bertrand, P. Bouyer, Th. Brihaye, and M. Größer. Almost-sure model checking of infinite paths in one-clock timed automata. Research Report LSV-07-29, ENS Cachan, France, 2007.
6. C. Baier, N. Bertrand, P. Bouyer, Th. Brihaye, and M. Größer. Probabilistic and topological semantics for timed automata. In *Proc. 27th Conf. Found. Softw. Tech. & Theor. Comp. Sci. (FSTTCS'07)*, LNCS. Springer, 2007. To appear.
7. P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. The cost of punctuality. In *Proc. 22nd Ann. Symp. Logic in Computer Science (LICS'07)*. IEEE Comp. Soc. Press, 2007. 109–118.
8. P. Bouyer, N. Markey, and P.-A. Reynier. Robust model-checking of timed automata. Research Report LSV-05-06, ENS Cachan, France, 2005.

9. P. Bouyer, N. Markey, and P.-A. Reynier. Robust model-checking of timed automata. In *Proc. 7th Latin American Symp. Theoretical Informatics (LATIN'06)*, LNCS 3887, pages 238–249. Springer, 2006.
10. M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. Kronos: a model-checking tool for real-time systems. In *Proc. 10th Intl Conf. Computer Aided Verification (CAV'98)*, LNCS 1427, pages 546–550. Springer, 1998.
11. F. Cassez, Th. A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *Proc. 5th Intl Workshop Hybrid Systems: Computation & Control (HSCC'02)*, LNCS 2289, pages 134–148. Springer, 2002.
12. C. Daws and P. Kordy. Symbolic robustness analysis of timed automata. In *Proc. 4th Intl Conf. Formal Modeling and Analysis of Timed Systems (FORMATS'06)*, LNCS 4202, pages 143–155. Springer, 2006.
13. M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. In *Proc. Joint Conf. Formal Modelling and Analysis of Timed Systems & Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, LNCS 3253, pages 118–133. Springer, 2004.
14. M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. Tech. Report 2004.30, Centre Fédéré en Vérification, Belgium, Dec. 2005. Revised version.
15. M. De Wulf, L. Doyen, and J. Raskin. Almost ASAP semantics: From timed models to timed implementations. In *Proc. 7th Intl Workshop Hybrid Systems: Computation & Control (HSCC'04)*, LNCS 2993, pages 296–310. Springer, 2004.
16. C. Dima. Dynamical properties of timed automata revisited. In *Proc. 5th Intl Conf, Formal Modeling and Analysis of Timed Systems (FORMATS'07)*, LNCS 4763, pages 130–146. Springer, 2007.
17. T. French, J. C. McCabe-Dansted, and M. Reynolds. A temporal logic of robustness. In *Proc. 6th Intl Workshop Frontiers of Combining Systems (FroCoS'07)*, LNAI 4720, pages 193–205. Springer, 2007.
18. V. Gupta, Th. A. Henzinger, and R. Jagadeesan. Robust timed automata. In *Proc. Intl Workshop Hybrid and Real-Time Systems (HART'97)*, LNCS 1201, pages 331–345. Springer, 1997.
19. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.
20. K. G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *J. Software Tools for Technology Transfer*, 1(1–2):134–152, 1997.
21. J. Ouaknine and J. Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Comp. Sci.*, 2007. To appear.
22. J. Ouaknine and J. B. Worrell. Revisiting digitization, robustness and decidability for timed automata. In *Proc. 18th Ann. Symp. Logic in Computer Science (LICS'03)*. IEEE Comp. Soc. Press, 2003.
23. J. Ouaknine and J. B. Worrell. On the decidability of metric temporal logic. In *Proc. 19th Ann. Symp. Logic in Computer Science (LICS'05)*, pages 188–197. IEEE Comp. Soc. Press, 2005.
24. A. Pnueli. The temporal logic of programs. In *Proc. 18th Ann. Symp. Foundations of Computer Science (FOCS'77)*, pages 46–57. IEEE Comp. Soc. Press, 1977.
25. A. Puri. Dynamical properties of timed automata. In *Proc. 5th Intl Symp. Formal techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'98)*, LNCS 1486, pages 210–227. Springer, 1998.
26. M. Swaminathan and M. Fränzle. A symbolic decision procedure for robust safety of timed systems. In *Proc. 14th Intl Symp. Temporal Representation and Reasoning (TIME'07)*, page 192. IEEE Comp. Soc. Press, 2007.

A Complements for Theorem 1

In this section, we give some explanations on techniques that can be used to analyse CAROTs, and take as an example the CAROT depicted on Figure 4 [28].

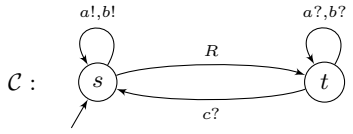

Fig. 4. Example of a CAROT

Figure 5 represents a computation of $\mathcal{C}$ in tabular form. Each line of the table represents a cycle of the channel (*i.e.*, the last operation is a $\triangleright!$), and, reading left-to-right, it records the sequence of transitions during that cycle. The most important property of the table is that the spacing is arranged so that an operation that reads a message is placed directly below the operation that originally wrote the message, necessarily in the previous cycle of the channel. Since we only deal with reachability at this point, we consider w.l.o.g. that the computation tables have finite width. In Figure 5 matching pairs of reads and writes are indicated by rectangular boxes. Because of global renaming the corresponding read and write events may not refer to the same element of Σ. For instance, in Figure 5, a write-event $b!$ is sometimes aligned with a read-event $c?$.

s	$b!$	s	$b!$	s	R	t	$\triangleright?$	$t_\triangleright$			$\triangleright!$	t												
	$b?$	t	$c?$	s	$a!$	s	$b!$	s	R	t	$\triangleright?$	$t_\triangleright$	$\triangleright!$	t										
					$a?$	t	$c?$	s	$b!$	s	R	t	$\triangleright?$	$t_\triangleright$	$\triangleright!$	t								
									$c?$	s	R	t			$\triangleright?$	$t_\triangleright$	$\triangleright!$	$t_\triangleright$	$\triangleright!$	t				

Fig. 5. Computation table

s	$b!$	s	$b!$	s	R	t	$\triangleright?$	$t_\triangleright$	$-$	$t_\triangleright$	$\triangleright!$	t	$-$	t	$-$	t	$-$	t						
t	$b?$	t	$c?$	s	$a!$	s	$b!$	s	R	t	$\triangleright?$	$t_\triangleright$	$\triangleright!$	t	$-$	t	$-$	t						
t	$-$	t	$-$	t	$a?$	t	$c?$	s	$b!$	s	R	t	$\triangleright?$	$t_\triangleright$	$\triangleright!$	t	$-$	t						
t	$-$	t	$-$	t	$-$	t	$-$	t	$c?$	s	R	t	$-$	t	$\triangleright?$	$t_\triangleright$	$\triangleright!$	t						

Fig. 6. Computation table with two sliding windows

We now shortly describe the non-deterministic procedure presented in [29] to check the existence of an accepting cycle-bounded computation using only polynomial space in the *value* of the cycle bound and in the size of the CAROT.

The first step is to fill in the blank spaces in the table by repeating the immediately preceding control state; for example, starting from Figure 5 we

obtain the table in Figure 6. Now, imagine a sliding window of dimension $3 \times h$ (where h is the table height, *i.e.*, the number of cycles of the machine), as depicted on Figure 6. The procedure consists in guessing the sequence of $3 \times h$-windows from left to right, verifying on-the-fly that each new window is *consistent* with the transitions of the CAROT. For instance, in Figure 6, after guessing the leftmost window, it is checked that $(s, b!, s)$ and $(t, b?, t)$ are valid transitions in $\mathcal{C}$. There are several other consistency checks to be done, in particular concerning the renaming relations and zero-testing operations, but we omit the details here, and refer to [29] for more details. The complete procedure only has to store the current sliding window (of size $3 \times h$) and two sequences of relations, each of size $h \times |\Sigma|^2$. Also, it is rather clear that if there exists an accepting computation, then there exists one that never visits twice the same global state (*i.e.*, the same sliding window and the same sequences of relations) between two letters of the input word w. Thus the nondeterministic procedure can be stopped after an exponential number of steps (more precisely after $|w| \cdot (|\delta| + |S|)^h \cdot (2^{|\Sigma|^2})^{2h}$ steps). This algorithm only requires pseudo-polynomial space to solve the cycle-bounded reachability problem for CAROTs, thus proving Theorem 1.

B Complements for Subsection 3.1

Lemma 2. *For every $n \geq 3$, $[\![\mathcal{A}]\!]_{\frac{1}{n}} \sqsubseteq [\![\mathcal{B}^n]\!] \sqsubseteq [\![\mathcal{A}]\!]_{\frac{2}{n}}$.*

Proof (of $[\![\mathcal{B}^n]\!] \sqsubseteq [\![\mathcal{A}]\!]_{\frac{2}{n}}$). We prove that the relation defined by

$$(\ell, v) \prec (\ell, v_{|X})$$

is the required simulation (where $\ell \in L$ and $v \in \mathbb{R}_+^{X_n}$.

We assume that $(\ell, v) \stackrel{\sigma}{\Rightarrow} (\ell', v')$, due to the transition $\ell \xrightarrow{g,\sigma,Y} \ell'$ from $\mathcal{A}$. If $x \leq k$ is a constraint of g, then we have $v(x) < k + 1$. If $v(x) \leq k$, then $v \models x \leq k$, hence $v \models_{\frac{2}{n}} x \leq k$. Otherwise, $v(x) > k$, so that we also have $\{v(x)\} \leq v(x_{i+1}) < v(x_{i-1})$ for some i. Since clocks x_{i+1} and x_{i-1} are reset when they reach 1, we have $v(x_{i+1}) - v(x_{i-1}) = \frac{2}{n} \mod 1$ (this is inherited from the initial valuation v_n), and $-1 \leq v(x_{i+1}) - v(x_{i-1}) \leq 1$. Thus $v(x_{i+1}) - v(x_{i-1}) = \frac{2}{n}$ or $v(x_{i+1}) - v(x_{i-1}) = \frac{2}{n} - 1$. The former equality is impossible since the guard requires $v(x_{i+1}) < v(x_{i-1})$. Thus, $\{v(x)\} \leq v(x_{i+1}) \leq \frac{2}{n}$, $v \models_{\frac{2}{n}} x \leq k$.

We similarly analyse the case of a constraint $x \geq k$ and easily come to the same conclusion: $v \models_{\frac{2}{n}} x \geq k$. Thus, we get that $v_{|X} \models_{\frac{2}{n}} g$. Hence $(\ell, v_{|X}) \xrightarrow{\sigma} (\ell', v'_{|X})$ and $(\ell', v') \prec (\ell', v'_{|X})$.

We next consider the case of $(\ell, v) \stackrel{t}{\Rightarrow} (\ell, v')$ for some $t \in \mathbb{R}^+$. Since the automata $\mathcal{B}_i$ cannot reset clocks in X, we have $v'_{|X} = v_{|X} + t$, and thus $(\ell, v_{|X}) \xrightarrow{t}_{\frac{2}{n}} (\ell, v'_{|X})$ (invariants are treated in a way similar to the guards above). □

Proof (of $[\![\mathcal{A}]\!]_{\frac{1}{n}} \sqsubseteq [\![\mathcal{B}^n]\!]$). We prove that the relation defined as

$$(\ell, v_{|X}) \prec (\ell, v)$$

for any valuation v s.t. $v(x_{i+1}) - v(x_i) = 1/n$ for each $0 \leq i < n$, is a strong simulation.

Now, let v and v' be two valuations s.t. $(\ell, v_{|X}) \xrightarrow{\sigma} (\ell', v'_{|X})$. Then there is a transition $\ell \xrightarrow{g,\sigma,r} \ell'$ in $\mathcal{A}$ s.t. $v_{|X} \models_{\frac{1}{n}} g$. Assume g contains a constraint $x \leq k$. If $v(x) \leq k$, then we are done. Otherwise, we have $k < v(x) \leq k + \frac{1}{n}$, so that $\{v(x)\} \leq \frac{1}{n}$. Choosing i such that $v(x_i)$ is minimal (amongst $\{v(x_j) \mid 0 \leq j < n\}$) leads to the expected constraint, namely $\{v(x)\} \leq v(x_{i+1}) < v(x_{i-1})$. Thus, $(\ell, v) \stackrel{\sigma}{\Rightarrow} (\ell', v')$ and $(\ell', v'_{|X}) \prec (\ell', v')$.

The argument is similar for constraints of the form $x \geq k$.

The case of delay transitions is immediate (as before, invariants are treated in a way similar to guards). □

Theorem 3. *Let $\varphi \in \mathsf{MTL}$. Then,*

$$\mathcal{A} \models\!\models \varphi \;\Leftrightarrow\; \exists n \geq 3 \text{ s.t. } [\![\mathcal{B}^n]\!] \models \varphi$$

Proof. Assume that $\mathcal{A} \models\!\models \varphi$. Since $\mathcal{L}([\![\mathcal{A}]\!]_{\delta'}) \subseteq \mathcal{L}([\![\mathcal{A}]\!]_{\delta})$ as soon as $\delta' \leq \delta$, there exists some $n \geq 3$ s.t. $[\![\mathcal{A}]\!]_{\frac{2}{n}} \models \varphi$. As $\mathcal{L}([\![\mathcal{B}^n]\!]) \subseteq \mathcal{L}([\![\mathcal{A}]\!]_{\frac{2}{n}})$, we get that $[\![\mathcal{B}^n]\!] \models \varphi$. Conversely, assume that $n \geq 3$ is s.t. $[\![\mathcal{B}^n]\!] \models \varphi$. As $\mathcal{L}([\![\mathcal{A}]\!]_{\frac{1}{n}}) \subseteq \mathcal{L}([\![\mathcal{B}^n]\!])$, we get that $[\![\mathcal{A}]\!]_{\frac{1}{n}} \models \varphi$, which entails $\mathcal{A} \models\!\models \varphi$. □

C Construction of the CAROT $\mathcal{C}_{\mathcal{A}}$

Let n be an integer. We first give a more detailed presentation of $\mathcal{T}_H^n$. Let M be the maximal constant appearing in the automaton $\mathcal{A}$. Then $M+1$ is larger than or equal to the maximal constant of $\mathcal{B}$. We assume that the clocks of $\mathcal{A}$ (and $\mathcal{B}$) take their values in $[0, M+1] \cup \{\bot\}$, where $\bot$ is a special value (intended to represent any value larger than $M+1$) satisfying the following requirements:

$$\bot + t = \bot \quad \text{for any } t \in \mathbb{R}^+ \qquad\qquad \bot > t \quad \text{for any } t \leq M$$
$$t + t' = \bot \quad \text{if } t + t' > M$$

We write Val for the set $[0, M+1] \cup \{\bot\}$, and Reg for the set $\{0, 1, \ldots, M+1, \bot\}$. If $\gamma \in \mathbb{R}^+$ and $\gamma \leq M$, we write $\mathsf{reg}(\gamma)$ for the largest integer in Reg which is smaller than or equal to γ. We write $\mathsf{reg}(\bot) = \bot$. We also define the set $R = L \times X_n \times \mathsf{Reg}$ and it set of subsets $\Lambda = \wp(R)$.

A configuration of the network of timed systems is a couple (ℓ, v) where ℓ is a location of $\mathcal{B}$ and v is a valuation of the clocks of $X_n = X \cup \{x_i \mid 0 \leq i < n\}$, and such that $v_{|X}$ satisfies the set of invariants of ℓ and $v(x_i) \leq 1$ for all $i < n$. Such a configuration is encoded as the element $C_{(\ell,v)} = \{(\ell, x, v(x)) \mid x \in X_n\}$, which is itself partitioned into a sequence of disjoint subsets C_0, C_1, ..., C_P, $C_\bot$, such that $C_\bot = \{(\ell, x, v) \in C \mid v = \bot\}$, $\bigcup_{i=0}^{P} C_i = C_{(\ell,v)} \setminus C_\bot$, and if $i, j \neq \bot$, for all $(\ell, x, v) \in C_i$ and $(\ell', x', v') \in C_j$, $\{v\} \leqslant \{v'\}$ iff $i \leqslant j$ (so that (ℓ, x, v) and (ℓ, x', v') are in the same block C_i iff v and v' have the same fractional part). We assume in addition that the fractional part of

elements in C_0 is 0 (even if it means that $C_0 = \varnothing$). We then let $H(C_{(\ell,v)}) = \mathsf{reg}(C_0)\mathsf{reg}(C_1)\mathsf{reg}(C_2)\ldots\mathsf{reg}(C_P)\mathsf{reg}(C_\perp) \in \Lambda^*$, where we abusively write $\mathsf{reg}(C)$ for $\{(\ell, x, \mathsf{reg}(v)) \mid (\ell, x, v) \in C\}$.

An execution of the network of timed systems is then composed of transitions $C \xrightarrow{\sigma} C'$ for $\sigma \in \Sigma$ and $C \xrightarrow{t} C'$ for $t \in \mathbb{R}^+$ in the usual way. Using the abstraction function H, it is possible to define a discrete transition system $\mathcal{T}_H$ which abstracts away precise timing information, but which simulates the behaviours of $\mathcal{A}$. The abstraction function also provides an equivalence relation $\equiv$ on configurations: $C \equiv C'$ iff $H(C) = H(C')$. As claimed in Subsection 3.2, since regions are compatible with extended guards, one can prove, extending a result of [35], that we have:

Lemma 4. *The equivalence relation $\equiv$ is a time-abstract bisimulation.*

We now define the CAROT $\mathcal{C}_\mathcal{A}$ meant to encode the executions of $\mathcal{B}^n$, for any integer n. Before giving details of the construction, we introduce some notations. We consider an extra symbol Δ meant to represent any clock x_i. Note that the parameter n will be passed to $\mathcal{C}_\mathcal{A}$ by this symbol: to simulate executions of $\mathcal{B}^n$, we will consider an initial content of the channel equal to $\langle\Delta\rangle^{n-1}$ (the n-th symbol Δ will initially lie in the control state). We also consider a special symbol ϵ_x for each clock $x \in X$ and denote by X^ϵ the set $\{\epsilon_x \mid x \in X\}$. We define the set Γ as $X \cup X^\epsilon \cup \{\Delta, \epsilon\}$. Given a clock $x \in X$, there exists a unique element $\epsilon_x \in X^\epsilon$ associated with it. We denote by *Lab* this mapping. We can now define formally the CAROT $\mathcal{C}_\mathcal{A}$ as follows:

- the channel alphabet contains the symbols $\langle$ and $\rangle$ that are used as delimiters between the descriptions of each C_i on the channel. It also contains the set of symbols Γ defined above.
- a location q is a tuple $\langle \ell, r, c_0, c_\Delta, s_1^+, s_2^+, s_1^-, s_2^-, s_1^\epsilon, s_2^\epsilon\rangle$ where:
  - ℓ ranges over the set L of locations of $\mathcal{A}$;
  - $r \in \mathsf{Reg}^X$ is the set of integral parts of the clocks of $\mathcal{A}$ (the integral part of the other clocks is always 0);
  - c_0 is an element of $\wp(X)$ corresponding to $H(C_0)$;
  - $c_\Delta \in \{\perp, 0, 1\}$ indicates if one[10] of the clocks x_i, with $0 \leq i \leq n-1$, is present or not in the control part ($\perp$ if not), and, in the former case, if its integral value is 0 or 1.
  - s_1^+, s_2^+, s_1^-, s_2^-, s_1^ϵ and s_2^ϵ are six elements intended to describe the content of the channel on his extremities. The four first are elements of $\wp(X)$ representing the sets of clocks very recently and recently read and expected very soon and soon, resp. The two last are elements of $\wp(X)$ that describe which symbols ϵ_x are expected to be at the right extremity of the channel.
- the initial state is the state $\langle \ell_0, \{0\}^X, X, 0, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing\rangle$;

[10] Recall that due to the special structure of the $\mathcal{B}_i$'s, at most one clock x_i can have a null fractional part.

– we now describe the set of transitions of the CAROT. There are two kinds of transitions: delay transitions and actions transitions. We first describe *delay transitions* and distinguish three cases, depending on the value of c_Δ:
  - **First case:** $c_\Delta = \bot$**.** In this case, no clock of any of the $\mathcal{B}_i$'s has an integral value. We then distinguish between two cases, depending on whether some clock of $\mathcal{A}$ has an integral value or not:
    (*i*) $c_0 \neq \varnothing$: the transition consists in transferring the set c_0 from the location to the channel; observe that the truth of the invariants will be preserved by such a transition, since if some clocks starts satisfying $x > k$, then it also satisfies $\{x\} \leq x_{i+1} < x_{i-1}$ for some i. We also keep track of the fact that those clocks have been seen "recently". This is achieved through the following rules[11]:

$$\{\langle \ell, r, c_0, \bot, s_1^+, s_2^+, s_1^-, s_2^-, s_1^\epsilon, s_2^\epsilon\rangle \xrightarrow{\langle c_0\rangle !} \langle \ell, r, \varnothing, \bot, s_1^+ \cup c_0, s_2^+, s_1^-, s_2^-, s_1^\epsilon, s_2^\epsilon\rangle \mid c_0 \neq \varnothing\}$$

    (*ii*) $c_0 = \varnothing$: in this case, we read the last cell of the form $\langle c\rangle$ of the channel, with $c \in \wp(\Gamma)$. If any x or ϵ_x in c is present in s_1^- or s_1^ϵ, then it is removed from that set. Also, the clocks in $c \cap X$ are written in c_0 in the target state, and their integral values are incremented. Note that at this step, elements ϵ are only read, just erasing them from the channel. Last, if $\Delta \in c$, then we end up in a state with $c_\Delta = 1$. Again, it can be observed that the invariants are preserved by such a transition: the only difficult case may occur if a clock $x \in c$ was needed to enforce $\{x\} \geq x_{i-1} > x_{i+1}$, due to some constraint $x < k$. But after the transition, the condition $x < k$ will not be fulfilled anymore.
    The corresponding rules are the following ones:

$$\{\langle \ell, r, \varnothing, \bot, s_1^+, s_2^+, s_1^-, s_2^-, s_1^\epsilon, s_2^\epsilon\rangle \xrightarrow{\langle c\rangle ?} \langle \ell, r', c \cap X, c_\Delta, s_1^+, s_2^+, s'^-_1, s_2^-, s'^\epsilon_1, s_2^\epsilon\rangle \mid r'(x) = r(x) + \chi_c,\ c_\Delta = \bot \text{ if } \Delta \notin c, 1 \text{ otherwise}, s'^-_1 = s_1^- \setminus (c \cap X) \text{ and } s'^\epsilon_1 = s_1^\epsilon \setminus Lab^{-1}(c \cap X^\epsilon)\}$$

  - **Second case:** $c_\Delta = 1$**.** In this case, no delay transition is possible. Indeed, one of the clocks x_i has reached value 1 and the corresponding $\mathcal{B}_i$ must fire a silent transition before time can elapse anew.
  - **Third case:** $c_\Delta = 0$**.** Similarly to the case $c_\Delta = \bot$, we simply write on the channel the clocks that leave their integral value, together with the

[11] For the sake of readability, we write e.g. $\xrightarrow{\langle c\rangle !}$ for the successive transitions $\xrightarrow{\langle !}$ followed by transitions writing the content of c (in no particular order) and finally $\xrightarrow{\rangle !}$. Similarly for transitions $\xrightarrow{\langle c\rangle ?}$.

symbol Δ since $c_\Delta = 0$; we also add the clocks of c_0 to the set s_1^+. Again, the truth of the invariants is preserved by that transition.

$$\{\langle \ell, r, c_0, 0, \varnothing, s_2^+, s_1^-, s_2^-, s_1^\epsilon, s_2^\epsilon\rangle \xrightarrow{\langle c_0 \cup \Delta\rangle!} \langle \ell, r, \varnothing, \bot, c_0, s_2^+, s_1^-, s_2^-, s_1^\epsilon, s_2^\epsilon\rangle\}$$

We now turn to the *action transitions*. There are again different kinds of such transitions, namely transitions of some $\mathcal{B}_i$, and transitions of $\mathcal{B}$.

- The former ones are simpler, since there is only one kind of silent transitions, resetting clock x_i as it equals 1. This is encoded in the control part of our channel machine through the value of c_Δ. We have to enforce that the invariant is preserved: this would not be the case if (and only if) the invariant of ℓ in $\mathcal{A}$ contains "$x \leq k$" as a conjunct, with $r(x) = k$ and $x \in u_2^+$.
  Note that when this clock x_i is reset from 1 to 0, this changes the global ordering of the set of clocks $(x_i)_{i\leq n}$, and thus changes the constraints on fractional parts of clocks of $\mathcal{A}$. Therefore, we have to update all the sets s_1^+, s_2^+, s_1^-, s_2^-, s_1^ϵ and s_2^ϵ. First, concerning the sets s_1^+ and s_2^+, it is easy to verify that they must be shifted: the new value of s_2^+ is the old one of s_1^+ and the new one of s_1^+ is the set of clocks whose fractional part is zero, *i.e.*, c_0. Regarding sets s_2^- and s_1^-, we first have to check that s_1^- is empty for allowing this transition (this checks that our "predictions" were correct). We then shift s_2^- to s_1^-, and set s_2^- to $\varnothing$. We act similarly for s_1^ϵ and s_2^ϵ.
  We obtain the following set of rules:

$$\{\langle \ell, r, c_0, 1, s_1^+, s_2^+, s_1^-, s_2^-, s_1^\epsilon, s_2^\epsilon\rangle \to \langle \ell, r, c_0, 0, \varnothing, s_1^+, s_2^-, \varnothing, s_2^\epsilon, \varnothing\rangle \mid s_1^- = s_1^\epsilon = \varnothing,\ (r(x) = k \wedge x \in u_2^+) \Rightarrow \text{"}x \leq k\text{"} \notin I(\ell)\}$$

- A transition $\ell \xrightarrow{g,\sigma,z} \ell'$ of $\mathcal{A}$ gives rise to transitions from $\langle \ell, r, c_0, s_1^+, s_2^+, s_1^-, s_2^-, s_1^\epsilon, s_2^\epsilon\rangle$ to $\langle \ell', r', c_0', s'^+_1, s'^+_2, s'^-_1, s'^-_2, s'^\epsilon_1, s'^\epsilon_2\rangle$ with the following conditions:
  - the clocks in z, which are being reset when the transition is fired, are set to 0 in r'. On $X \setminus z$, r and r' coincide.
  - the clocks in z are also added in c_0', so that $c_0' = c_0 \cup z$. Those clocks must also be removed from the channel, which is achieved by renaming each clock $x \in z$ into ϵ_x.
  - the guard g must be satisfied within Δ. Comparing the values of the clocks with integers is easy, since we always know the integral part of each clock and whether its fractional part is zero or not. Evaluating guards of the form $\{x\} \leq x_{i+1} < x_{i-1}$ (resp. $x_{i+1} < x_{i-1} \leq \{x\}$) is achieved by checking that $x \in s_1^+ \cup s_2^+$ (resp. by enforcing that $x \in s_1^- \cup s_2^-$ or $\epsilon_x \in s_1^\epsilon \cup s_2^\epsilon$). Note that while the sets s_1^+ and s_2^+ are exactly the sets of clocks having "small" fractional parts, the sets s_1^- and s_2^- (and s_1^ϵ and s_2^ϵ) correspond to *predictions* of clocks having "large" fractional parts (in other terms, clocks that

will be read *soon* from the channel). Thus, in order to verify a guard of the second form, it may be necessary to add some extra clocks in the sets s_1^- and s_2^-. This is what will be done next, through the sets t_1 and t_2. Formally, we define the satisfaction relation $\langle r, c_0, s_1^+, s_2^+, s_1^-, s_2^- \rangle \models g$ inductively as follows:

- $\langle r, c_0, s_1^+, s_2^+, s_1^-, s_2^- \rangle \models g_1 \wedge g_2$ iff $\langle r, c_0, s_1^+, s_2^+, s_1^-, s_2^- \rangle \models g_1$ and $\langle r, c_0, s_1^+, s_2^+, s_1^-, s_2^- \rangle \models g_2$;
- $\langle r, c_0, s_1^+, s_2^+, s_1^-, s_2^- \rangle \models x \leq k$ iff either $r(x) < k$, or $r(x) = k$ and $x \in c_0 \cup s_1^+ \cup s_2^+$.
- $\langle r, c_0, s_1^+, s_2^+, s_1^-, s_2^- \rangle \models x \geq k$ iff either $r(x) \geq k$, or $r(x) = k-1$ and $x \in s_1^- \cup s_2^-$.

$*$ the invariant in ℓ' can be computed in the very same way.

We end up with the following set of transitions:

$$\left\{ \begin{array}{l} \langle \ell, r, c_0, c_\Delta, s_1^+, s_2^+, s_1^-, s_2^-, s_1^\epsilon, s_2^\epsilon \rangle \xrightarrow{R} \langle \ell', r', c_0', c_\Delta, s'^+_1, s'^+_2, s'^-_2, s'^-_1, s_1'^\epsilon, s_2'^\epsilon \rangle \mid \\ \quad \exists \ell \xrightarrow{g,\sigma,z} \ell' \in \delta \text{ and } \exists t_1, t_2 \in \wp(X) \text{ s.t. the following holds:} \\ \quad\quad * \ r'(x) = 0 \text{ if } x \in z,\ r'(x) = r(x) \text{ if } x \notin z, \\ \quad\quad * \ c_0' = c_0 \cup z, \\ \quad\quad * \ s'^+_1 = s_1^+ \setminus z, \text{ and } s'^+_2 = s_2^+ \setminus z, \\ \quad\quad * \ t_1 \cup t_2 \subseteq X \setminus (c_0 \cup s_1^+ \cup s_2^+ \cup s_1^- \cup s_2^-), \\ \quad\quad * \ \langle r, c_0, s_1^+, s_2^+, s_1^- \cup t_1, s_2^- \cup t_2 \rangle \models g, \\ \quad\quad * \ \langle r', c_0', s'^+_1, s'^+_2, s'^-_1, s'^-_2 \rangle \models I(\ell), \\ \quad\quad * \ R \text{ is defined on } z \text{ by } R(x) = \epsilon_x \text{ if } x \in z_1 \cup z_2 \\ \quad\quad\quad \text{and } R(x) = \epsilon \text{ otherwise.} \\ \quad\quad * \ \text{letting } z_1 = z \cap (s_1^- \cup t_1) \text{ and } z_2 = z \cap (s_2^- \cup t_2), \\ \quad\quad\quad \text{we have } s'^-_1 = s_1^- \cup t_1 \setminus z_1,\ s'^-_2 = s_2^- \cup t_2 \setminus z_2, \\ \quad\quad\quad s_1'^\epsilon = s_1^\epsilon \cup z_1, \text{ and } s_2'^\epsilon = s_2^\epsilon \cup z_2. \end{array} \right\}$$

D Correctness of $\mathcal{C}_\mathcal{A}$

In order to prove the correctness of the construction of the CAROT $\mathcal{C}_\mathcal{A}$, we will first show, as stated by Lemma 6, that the transition system associated with the CAROT $\mathcal{C}_\mathcal{A}$ is in bisimulation with the transition system $\mathcal{T}_H^n$ obtained for the equivalence $\equiv$ and for $\mathcal{B}^n$:

Lemma 6. *For any $n \geq 3$, the transition systems $\mathcal{T}_H^n$ and $\mathcal{T}_{\mathcal{C}_\mathcal{A}}^n$ are bisimilar.*

We first introduce some definitions to reason on sequences of the CAROT. Let us pick a configuration $(d, c) \in Q \times \Sigma^*$ composed of a location d and the content of the channel c. Following our construction, we have $d = \langle \ell, r, c_0, c_\Delta, s_1^+, s_2^+, s_1^-, s_2^-, s_1^\epsilon, s_2^\epsilon \rangle$ and $c = \langle c_1 \rangle \langle c_2 \rangle \dots \langle c_p \rangle$ with $c_i \in \wp(\Gamma)$ for every i in $\{1, \dots, p\}$. Intuitively, two clocks appear in the same cell c_i if and only if they have an equal fractional part, and clocks in c_i have a smaller fractional part than these in c_j if and only if $i < j$. Using this description, it is possible to define the set u_1^+ of clocks of X that are on the channel before the first occurrence of Δ (including eventually the occurrence of Δ in the discrete location in the case $c_\Delta = 0$), which

corresponds to the smallest clock x_i. More precisely, suppose for example that c_Δ equals $\perp$ and let $i_1^+ = \min\{1 \leq j \leq p \mid \Delta \in c_j\}$; then we have:

$$u_1^+ = \bigcup_{1 \leq j \leq i_1^+} c_j \cap X.$$

More generally, let us define the following positions:

$$\begin{cases} i_2^+ & = \min\{i_1^+ < j \leq p \mid \Delta \in c_j\} \\ i_1^- & = \max\{1 \leq j \leq p \mid \Delta \in c_j\} \\ i_2^- & = \max\{1 \leq j < i_1^- \mid \Delta \in c_j\} \end{cases}$$

and consider the function f defined by $f(i,j,Y) = \bigcup_{i \leq k \leq j} c_k \cap Y$, for two indices i and j and a set $Y \subseteq \Gamma$. We define the followings sets:

	$c_\Delta = \perp$	$c_\Delta = 0$	$c_\Delta = 1$
u_1^+	$f(1, i_1^+, X)$	$\varnothing$	$f(1, i_1^+, X)$
u_2^+	$f(i_1^+ + 1, i_2^+, X)$	$f(1, i_1^+, X)$	$f(i_1^+ + 1, i_2^+, X)$
u_1^-	$f(i_1^-, p, X)$	$f(i_1^-, p, X)$	$\varnothing$
u_2^-	$f(i_2^-, i_1^- - 1, X)$	$f(i_2^-, i_1^- - 1, X)$	$f(i_1^-, p, X)$
$Lab(u_1^\epsilon)$	$f(i_1^-, p, X^\epsilon)$	$f(i_1^-, p, X^\epsilon)$	$\varnothing$
$Lab(u_2^\epsilon)$	$f(i_2^-, i_1^- - 1, X^\epsilon)$	$f(i_2^-, i_1^- - 1, X^\epsilon)$	$f(i_1^-, p, X^\epsilon)$

Intuitively, the set u_2^+ represents the clocks of X present on the channel before the second occurrence of Δ and strictly after the first occurrence of Δ. Similarly, the sets u_1^- and u_2^- describe the content of the right-extremity of the channel, and the sets u_1^ϵ and u_2^ϵ the same extremity but for elements of X^ϵ.

Then, we say that a configuration (d, c) is *without prediction* if $s_1^- = s_2^- = s_1^\epsilon = s_2^\epsilon = \varnothing$. We say that a configuration (d, c) is *with correct predictions* if it satisfies the following inequalities:

$$\begin{cases} s_1^- \subseteq u_1^-, \ s_2^- \subseteq u_2^-, \\ s_1^\epsilon \subseteq u_1^\epsilon, \ s_2^\epsilon \subseteq u_2^\epsilon. \end{cases}$$

We can now define what will be the bisimulation relation $\approx$. First, note that given a configuration (d, c) of the CAROT, and an index $i \in \{0, \ldots, n-1\}$, there exists a unique word $w_i(d, c) = H_0 H_1 \ldots H_p H_\perp \in \Lambda^*$ associated with it in our encoding and such that i is the index of the smallest clock x_j, for $j \in \{0, \ldots, n-1\}$. The other words $w_j(d, c)$ are obtained from $w_i(d, c)$ by a circular permutation of the indexes of x_k's. This degree of freedom on indexes is due to the abstraction of these clocks in (d, c) where they are all represented by the symbol Δ. In addition, observe that the word $w_i(d, c)$ only depends on the 4-tuple $\langle \ell, r, c_0, c_\Delta \rangle$ and on c. We define the relation $\approx$ between the elements of Λ^* and the set of configurations of $\mathcal{C}_\mathcal{A}$ *with correct predictions* as follows:

$$w \approx (d, c) \Leftrightarrow \exists i.\ w_i(d, c) = w.$$

Note that, for any integer $n \in \mathbb{N}$, we have that the configuration (d^n, c^n) associated via $\approx$ with the configuration $H(\ell_0, \{0\}^X)$ in $[\![\mathcal{B}^n]\!]$ verifies:

$$\begin{cases} d^n = \langle \ell_0, \{0\}^X, X, 0, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing \rangle \\ c^n = \langle \Delta^{n-1} \rangle \end{cases}$$

Before proving that this relation is a bisimulation relation, we prove some technical lemmas.

Lemma 17. *Let (d, c) be a reachable configuration of the CAROT $\mathcal{C}_\mathcal{A}$ from initial configuration (d^n, c^n) with $n \geq 3$.*[12] *Then $s_1^+ = u_1^+$ and $s_2^+ = u_2^+$.*

Proof. The proof is by induction on the length of an execution of the CAROT leading to that configuration. The result is obvious in the initial configuration of the CAROT, where all the sets are empty.

Assume the result holds in a given configuration (d, c), and pick a successor configuration (d', c'). We first consider the case where the transition from (d, c) to (d', c') corresponds to a delay transition. We split this case into several subcases:

- if $c_\Delta = \bot$ and $c_0 \neq \varnothing$ in (d, c): then c_0 is written on the channel, thus added to u_1^+, and also added to s_1^+, so that their equality is preserved. The sets s_2^+ and u_2^+ are unchanged.
- if $c_\Delta = \bot$ and $c_0 = \varnothing$, then u_1^+, u_2^+ and s_1^+, s_2^+ are unchanged;
- if $c_\Delta = 0$, then in configuration (d', c'), we have $i_1^+ = 1$ (since Δ has just been written on the channel), so that u_1^+ exactly contains c_0 and the set u_2^+ remains unchanged along this transition. This exactly corresponds to s_1^+ and s_2^+, resp.

We now consider the two kinds of action transitions:

- If the transition corresponds to a transition in some $\mathcal{B}_i$ (the first kind of action transition), then the content of s_1^+ is shifted into s_2^+ and s_1^+ is emptied. This precisely corresponds to the way we compute u_1^+ and u_2^+.
- If the transition corresponds to a transition of $\mathcal{A}$, then the clocks that are being reset along that transition are removed from s_1^+ and s_2^+, and they are simultaneously rewritten into their corresponding ϵ_x-symbol (or into the symbol ϵ). Our equalities are thus preserved. □

Lemma 18. *Let ϱ be an execution of the CAROT $\mathcal{C}_\mathcal{A}$ starting in (d^n, c^n) with $n \geq 3$. We suppose that every configuration reached along ϱ has correct predictions. Then, the following properties hold:*

(i) for every configuration (d, c) reached along ϱ:

$$\begin{array}{ll} s_2^\epsilon \subseteq c_0 \cup s_1^+ & \text{and } s_1^\epsilon \cup s_2^\epsilon \subseteq c_0 \cup s_1^+ \cup s_2^+ \\ u_2^\epsilon \subseteq c_0 \cup u_1^+ & \text{and } u_1^\epsilon \cup u_2^\epsilon \subseteq c_0 \cup u_1^+ \cup u_2^+ \end{array}$$

[12] This is for indices i_1^+, i_2^+, i_1^- and i_2^- to always be correctly defined.

(*ii*) *for every configuration* (d, c) *reached along* ϱ*, every occurrence of a symbol* ϵ_x *appears in the right-extremity of the channel, in a cell covered by* $u_1^\epsilon \cup u_2^\epsilon$*. Formally, this means that if* c_Δ *equals bot or* 0 *(resp.* 1*), then* $j < i_2^-$ *(resp.* i_1^-*) implies* $c_j \cap X^\varepsilon = \varnothing$*.*

(*iii*) *if a transition* $(d, c) \xrightarrow{e} (d', c')$ *with* $e = \ell \xrightarrow{g,\sigma,z} \ell'$ *is fired along* ϱ*, then the sets* t_1 *and* t_2 *of the corresponding rule of* $\mathcal{C}_\mathcal{A}$ *satisfy:*

$$t_1 \subseteq u_1^- \text{ and } t_2 \subseteq u_2^-$$

Proof. The proof proceeds again by induction on the length of ϱ, and considers successively each of the three items.

The result holds trivially for the initial configuration. Now assume that it holds for some configuration (d, c), and pick a successor configuration (d', c'). Along delay transitions, the sets s_2^ϵ and u_2^ϵ remain unchanged while s_1^ϵ and u_1^ϵ can only become smaller: clocks reaching an integral value are removed from the channel. On the other hand, s_2^+ is never changed and s_1^+ can only get larger, and similarly for u_1^+ and u_2^+ by Lemma 17. The inclusions are thus preserved. Along an action transition corresponding to a transition of some $\mathcal{B}_i$, the set s_2^ϵ is emptied (and thus satisfies its inclusion); its content is transfered into s_1^ϵ, while the content of s_1^+ is transfered into s_2^+, so that the second inclusion also holds. Similarly, for the sets u_2^ϵ and u_1^ϵ, we can observe that changing c_Δ from 1 to 0 implies that the content of u_2^ϵ is transfered into u_1^ϵ. Finally, using property (*ii*), we can argue that the new value of u_2^ϵ is $\varnothing$. Indeed, the new interval concerned by the set u_2^ϵ is "shifted" on the left, and thus the new value of this set is the empty set. Last, along action transitions derived from transitions of $\mathcal{A}$, any item that is added to s_1^ϵ or s_2^ϵ is also added into c_0. Similarly for u_1^ϵ and u_2^ϵ.

We now prove the item (*ii*). By contradiction, suppose that this property is not satisfied. By a simple examination of the rules of $\mathcal{C}_\mathcal{A}$, one can notice that the only rules writing new elements of X^ϵ correspond to transitions of $\mathcal{A}$. Let us consider such a transition $t = \ell \xrightarrow{g,\sigma,z} \ell'$. Note that these rewriting operations only occur for elements of $z_i = (s_i^- \cup t_i) \cap z$, for $i \in \{1, 2\}$. Using the third part of this lemma (item (*iii*)) and the fact that configurations are with correct predictions, we obtain $z_i \subseteq u_i^-$ for $i \in \{1, 2\}$. This concludes the proof of this item.

Finally, we come to the proof of the third item. The proof is by contradiction. Suppose first that $t_1 \not\subseteq u_1^-$, and let $x \in X$ such that $x \in t_1 \setminus u_1^-$. We denote by prime elements the sets corresponding to the target configuration (d', c').

- *First case:* $x \notin z$. Considering s'^-_1, we will show that (d', c') has wrong predictions, which yields a contradiction. Indeed, we have $x \in s'^-_1 = s_1^- \cup t_1$ and $x \notin u'^-_1 = u_1^- \setminus z$ (by hypothesis, $x \notin u_1^-$). We obtain $s'^-_1 \not\subseteq u'^-_1$, as desired.
- *Second case:* $x \in z$. Then we have $x \in z_1 = (s_1^- \cup t_1) \cap z$. As a consequence, $x \notin s'^-_1 = s_1^- \cup t_1 \setminus z_1$ and $x \in s'^\epsilon_1 = s_1^\epsilon \cup z_1$. As above, we will show that this implies that (d', c') does not have correct predictions. More precisely, we show that $x \notin u'^\epsilon_1$, and thus that $s'^\epsilon_1 \not\subseteq u'^\epsilon_1$. Let us compute the set u'^ϵ_1. We

have ${u'}_1^\epsilon = u_1^\epsilon \cup (u_1^- \cap z)$. First, by hypothesis, we have $x \notin u_1^-$. By definition of the rules of $\mathcal{C}_\mathcal{A}$, we have $t_1 \cap (c_0 \cup s_1^+ \cup s_2^+) = \varnothing$. By Lemma 17, we obtain $t_1 \cap (c_0 \cup u_1^+ \cup u_2^+) = \varnothing$. Finally, the property stated by item (i) of this lemma yields $t_1 \cap u_1^\epsilon = \varnothing$, and thus $x \notin u_1^\epsilon$. As a consequence, $x \notin {u'}_1^\epsilon$, as desired.

The reasoning for the case $t_2 \not\subseteq u_2^-$ is similar. □

We introduce some additional definitions. Given a valuation v of the clocks in X_n, we define the following four sets of clocks:

$$S_1^+(v) = \{x \in X \mid \exists i.\ 0 < \{v(x)\} \leq v(x_i) < v(x_{i+1}) < v(x_{i-1})\}$$
$$S_2^+(v) = \{x \in X \mid \exists i.\ 0 \leq v(x_i) < \{v(x)\} \leq v(x_{i+1}) < v(x_{i-1})\}$$
$$S_1^-(v) = \{x \in X \mid \exists i.\ v(x_{i+1}) < v(x_{i-1}) < v(x_i) \leq \{v(x)\} < 1\}$$
$$S_2^-(v) = \{x \in X \mid \exists i.\ v(x_{i+1}) < v(x_{i-1}) \leq \{v(x)\} < v(x_i) \leq 1\}.$$

It is easily observed that those sets are equal as soon as valuations v and v' are region-equivalent (*i.e.*, (ℓ, v) and (ℓ, v') correspond to the same configuration w). As a consequence, given a configuration w, we will write w instead of v as a parameter of these function, *i.e.* $S_1^+(w)$ instead of $S_1^+(v)$ for the first set for example.

We now claim a first lemma, which can be easily proved by examining definitions:

Lemma 19. *Let $n \geq 3$. Given a configuration $w \in \Lambda^*$ of $\mathcal{T}_H^n$, and a configuration (d, c) with correct predictions of the CAROT $\mathcal{C}_\mathcal{A}$ such that $w \approx (d, c)$. We have:*

$$\forall i \in \{1, 2\}, \forall \diamond \in \{+, -\}, S_i^\diamond(w) = u_i^\diamond(d, c)$$

We can now prove that the relation $\approx$ is a bisimulation relation. We begin with a first implication.

Lemma 20. *Let $n \geq 3$, and (d, c) be a reachable configuration from initial configuration (d^n, c^n), with correct predictions, and $w \approx (d, c)$. Consider a transition $(d, c) \to (d', c')$, where (d', c') also has correct predictions. Then, there exists $w' \in \Lambda^*$ such that $w \to w'$ and $w' \approx (d', c')$.*

Proof. The proof depends on the kind of transition being applied between (d, c) and (d', c'). We begin with delay transitions:

- if $c_\Delta = \bot$ and $c_0 \neq \varnothing$ in d, then the corresponding configuration $w \in \Lambda^*$ is of the form $w = H_0 H_1 \dots H_p H_\bot$ with $H_0 \neq \varnothing$ and there is no clock x_i, for $i \in \{0, \dots, n-1\}$, such that $x_i \in H_0$. Then the immediate delay successor of w is $w' = \varnothing H_0 H_1 \dots H_p H_\bot$, which naturally verifies $w \to w'$. Finally, it is routine to verify that if $w = w_i(d, c)$, then we have $w' = w_i(d', c')$.
- if $c_\Delta = \bot$ and $c_0 = \varnothing$, then $w = \varnothing H_1 \dots H_p H_\bot$. The immediate delay successor w' of w is $w' = H_p H_1 \dots H_{p-1} H_\bot$. It is again clear that it verifies $w' = w_i(d', c')$ as soon as $w = w_i(d, c)$.

– if $c_\Delta = 0$, then we have $w = H_0 H_1 \dots H_p H_\perp$ with $x_i \in H_0$ for some index i. In particular, we have $w = w_i(d, c)$. Then the immediate delay successor of w is $w' = \varnothing H_0 H_1 \dots H_p H_\perp$ and verifies both $w \to w'$ and $w' = w_i(d', c')$.

We now turn to the action transitions:

– if the transition from (d, c) to (d', c') corresponds to the action transition of some $\mathcal{B}_i$, then we have $w = w_{i+1}(d, c)$, which can be written $w = H_0 H_1 \dots H_p H_\perp$ with $x_i \in H_0$. More precisely, in order to fire this transition, we must have the tuple $(\ell, x_i, 1)$ element of H_0. Then a successor of w is the element $w' = H_0' H_1 \dots H_p H_\perp$ where H_0' is obtained from H_0 by replacing the element $(\ell, x_i, 1)$ by $(\ell, x_i, 0)$. This transition is allowed as we have enforced the invariant to be preserved. We obtain $w' = w_i(d', c')$ and $w \to w'$.
– last, if the transition corresponds to a transition $\ell \xrightarrow{g,\sigma,z} \ell'$ of $\mathcal{A}$, then let t_1 and t_2 be the sets of clocks corresponding to the corresponding move in the CAROT (they correspond to "predictions" of clocks having large fractional part). It is clear enough that the configuration w' corresponds to the state obtained from w by applying this transition (the location is updated, clocks in z are put in c_0 and renamed on the channel). We now prove that the condition $\langle r, c_0, s_1^+, s_2^+, s_1^- \cup t_1, s_2^- \cup t_2 \rangle \models g$, which is required to hold in (d, c) for the transition to exist, entails that $w \models g$. This is achieved inductively:
  - this is clear when $g = g_1 \wedge g_2$;
  - if g is of the form $x \leq k$, then the corresponding guard in $\mathcal{B}$ is $x < k+1 \wedge (x > k \Rightarrow \bigvee_{0 \leq i < n} \{x\} \leq x_{i+1} \leq x_{i-1})$. By Lemmas 17, 19, we have the equality between $s_1^+(d, c)$ (resp. $s_2^+(d, c)$) and $S_1^+(w)$ (resp. $S_2^+(w)$). As a consequence, it is easily seen that the satisfaction of $\langle r, c_0, s_1^+, s_2^+, s_1^- \cup t_1, s_2^- \cup t_2 \rangle \models g$ implies the satisfaction of the guard in $\mathcal{B}$.
  - if g is of the form $x \geq k$, then the corresponding guard in $\mathcal{B}$ is $x > k-1 \wedge (x < k \Rightarrow \bigvee_{0 \leq i < n} \{x\} \geq x_{i-1} \geq x_{i+1})$. By Lemma 18.(*iii*) together with Lemma 19, we obtain $s_1^-(d, c) \cup t_1 \subseteq S_1^-(w)$ (respectively $s_2^-(d, c) \cup t_2 \subseteq S_2^-(w)$). By definition of $\langle r, c_0, s_1^+, s_2^+, s_1^- \cup t_1, s_2^- \cup t_2 \rangle \models g$, this immediately implies that the guard of $\mathcal{B}$ is satisfied in w.

We prove in a similar way that the invariant in ℓ' is fulfilled. □

And conversely, for the second implication,

Lemma 21. *Let $w \in \Lambda^*$ and (d, c) with correct predictions such that $w \approx (d, c)$. Consider a transition $w \to w'$. Then there exists a configuration (d', c') with correct predictions s.t. $w' \approx (d', c')$ and $(d, c) \to (d', c')$.*

Proof. We again study each of the five possible transitions from w to w'. Regarding *delay* transitions, we only consider transitions going from one region to its immediate successor, the general case being a succession of such atomic delay transitions.

– for delay transitions in which some clocks *leave* an integral value, we have $w = H_0 H_1 \dots H_p H_\perp$ with $H_0 \neq \varnothing$ and $w' = \varnothing H_0 H_1 \dots H_p H_\perp$. We distinguish

two cases, depending on whether some clock x_i appears in H_0. If this is the case, *i.e.* there exists some index i with x_i appearing in H_0, then we must have $(\ell, x_i, 0)$ element of H_0. Indeed, due to the invariant on x_i, delay transition is only possible from the integral value 0. As a consequence, we must have $c_\Delta = 0$ in the configuration (d, c). Then, defining (d', c') as the delay successor of (d, c), we obtain the desired configuration. In the latter case, *i.e.* if there is no clock x_i in H_0, then we must have $c_\Delta = \bot$ in (d, c) and we can apply the delay transition to this configuration. The target configuration, say (d', c'), verifies both conditions.

– for delay transitions in which some clocks *reach* an integral value, we have $w = \varnothing H_1 \dots H_p H_\bot$ and $w' = H_p H_1 \dots H_{p-1} H_\bot$. This implies that in the configuration (d, c), we have both $c_0 = \varnothing$ and $c_\Delta = \bot$. As a consequence, there exists a unique delay transition from (d, c), leading to the configuration (d', c'), which satisfies the desired constraints.

Considering now *action* transitions, we distinguish between action transitions of $\mathcal{B}_i$'s and those of $\mathcal{A}$. In the former case, we must have $(\ell, x_i, 1)$ element of H_0, with $w = H_0 H_1 \dots H_p H_\bot$ for some index i. This implies that (d, c) verifies $c_\Delta = 1$, and thus we can apply the $\mathcal{B}_i$'s transition to the configuration (d, c). Also, the invariant must be preserved by that transition of $\mathcal{B}_i$, which ensures that the corresponding transition exists in $\mathcal{C}_\mathcal{A}$. It is routine to verify that the target configuration (d', c') verifies $w' \approx (d', c')$. The latter case, for action transitions of $\mathcal{A}$, is more involved. Indeed, we must verify that firing a transition in $\mathcal{T}_H$ implies that the same transition is also firable in $\mathcal{C}_\mathcal{A}$, staying in the set of configurations with correct predictions. To that purpose, we notice that the firability of a transition from the configuration w only depends on the integral values of clocks (which is preserved in the corresponding configuration (d, c)) and on the sets $S_1^+(w)$, $S_2^+(w)$, $S_1^-(w)$ and $S_2^-(w)$. Due to Lemma 17 together with Lemma 19, we have the equalities $s_1^+(d, c) = S_1^+(w)$ and $s_2^+(d, c) = S_2^+(w)$. This implies the correctness for guards of the form $x \leq k$. For guards of the form $x \geq k$, we only have the inclusions $s_1^-(d, c) \subseteq S_1^-(w)$ and $s_2^-(d, c) \subseteq S_2^-(w)$. Then, the transition may not be firable from (d, c) in the sense that $\langle r, c_0, s_1^+(d, c), s_2^+(d, c), s_1^-(d, c), s_2^-(d, c)\rangle \not\models x \geq k$. To allow firability, we can use the sets t_1 and t_2 to enlarge the sets $s_1^-(d, c)$ and $s_2^-(d, c)$. Let us define $t_i = S_i^-(w) \setminus s_i^-(d, c)$ for $i \in \{1, 2\}$. We of course have $\langle r, c_0, s_1^+(d, c), s_2^+(d, c), s_1^-(d, c) \cup t_1, s_2^-(d, c) \cup t_2\rangle \models x \geq k$. It remains to verify the sets t_1 and t_2 meet the requirements imposed in the definition of the rules of $\mathcal{C}_\mathcal{A}$, *i.e.*, $t_1 \cup t_2 \subseteq X \setminus (c_0 \cup s_1^+(d, c) \cup s_2^+(d, c) \cup s_1^-(d, c) \cup s_2^-(d, c))$. By definition, we have $(t_1 \cup t_2) \cap (s_1^-(d, c) \cup s_2^-(d, c)) = \varnothing$. Moreover, Lemma 19 implies that $t_1 \cup t_2 \subseteq u_1^-(d, c) \cup u_2^-(d, c)$. It is easy to verify that every clock appears exactly once on the channel or in the set c_0. Then, the intersection $(u_1^-(d, c) \cup u_2^-(d, c)) \cap (c_0 \cup u_1^+(d, c) \cup u_2^+(d, c))$ is empty. By Lemma 17, this allows to conclude that the guard is fulfilled. The case of invariants is similar (and easier). □

The two lemmas 20 and 21 together prove Lemma 6. A last point remains to be detailed. Once we have this bisimulation relation, how can we study the

CAROT $\mathcal{C}_\mathcal{A}$ only on its configurations *with correct predictions*? We prove the following lemma, from which immediately follows Lemma 7, stated in Section 3.

Lemma 22. *Let ϱ be an execution in $\mathcal{C}_\mathcal{A}$ with infinitely many delay transitions. Then every configuration reached along ϱ is with correct predictions.*

Proof. The proof proceeds by contradiction. Suppose that at some time the execution ϱ goes through a configuration (d, c) with false predictions. This means that the configuration contains in its discrete part d a clock in one of the predictive sets s_1^-, s_2^-, s_1^ϵ and s_2^ϵ which does not appear in the corresponding descriptive set in u_1^-, u_2^-, u_1^ϵ and u_2^ϵ. A careful analysis of the rules of $\mathcal{C}_\mathcal{A}$ shows that only clocks which are present on the channel, *i.e.* in one of the descriptive sets, can be removed from the predictive sets. As a consequence, one of these sets will not be empty when trying to let some time elapse, which contradicts the firability of ϱ. Note that this time elapsing transition will necessarily occur because ϱ has infinitely many delay transitions. □

The following lemma describes another favourable case, which will be used later (proof of Lemma 12):

Lemma 23. *Let $n \geq 3$, and ϱ be an execution of the CAROT $\mathcal{C}_\mathcal{A}$ starting in the initial configuration d^n, c^n and ending with a configuration (d_f, c_f) with correct predictions. Then all the configurations reached along ϱ are with correct predictions.*

Proof. Let (d, c) be a configuration with false predictions reached along ϱ. We will see that all its successors, by any transition, also have false predictions. The result immediately follows.

We consider the different kind of possible transitions, and begin with delay transitions:

- First, if we have a delay transition from $c_\Delta = \bot$ and $c_0 \neq \varnothing$, then the sets of predictions are unchanged, and so are the sets $u_1^-, u_2^-, u_1^\epsilon$ and u_2^ϵ. Then, the new configuration still has wrong predictions.
- Second, if we have $c_\Delta = \bot$ and $c_0 = \varnothing$, then it is easy to see that the same clocks are removed from the sets s_1^- (resp. s_1^ϵ) and u_1^- (resp. u_1^ϵ). Then, the new configuration still has wrong predictions.
- Third, we consider a delay transition from a configuration verifying $c_\Delta = 0$. Then, the sets concerning predictions are unchanged, and so are the sets $u_1^-, u_2^-, u_1^\epsilon$ and u_2^ϵ, what implies that the new configuration still has wrong predictions.

We turn now to action transitions. For actions in some $\mathcal{B}_i$, note that the configuration (d, c) must satisfy $s_1^- = s_1^\epsilon = \varnothing$. As a consequence, these two sets can not be concerned by wrong predictions. Thus, wrong predictions of configuration (d, c) must be located in s_2^- or in s_2^ϵ. Note that in the new configuration, new sets concerning predictions are obtained via shifting theses: sets of index 2 are emptied, and sets of index 1 are defined as previous sets of index 2. Concerning

the sets $u_1^-, u_2^-, u_1^\epsilon$ and u_2^ϵ, it is easily observed that the new sets are obtained using the same operation. As a consequence, the new configuration still has wrong predictions.

Finally, we consider action transitions corresponding to transitions of $\mathcal{A}$, say some transition $(\ell, g, \sigma, z, \ell')$. Let $i \in \{1, 2\}$ and suppose first that $s_i^- \not\subseteq u_i^-$. Let $x \in s_i^- \setminus u_i^-$. Two cases may arise, depending on whether $x \in z$ or not. We denote by prime elements the elements corresponding to the reached configuration.

- First case: $x \notin z$. After the firing of this transition, the content of the cells concerned by u_i^- are only affected by the rewriting operation. As a consequence, we have $u_i'^- \subseteq u_i^-$ and $x \in s'^-_i$, what implies that the new configuration has wrong predictions.
- Second case: $x \in z$. Suppose that $x \in u_i^\epsilon$. Applying a reasonning similar to this of the proof of Lemma 18, one can show that if $x \in u_i^\epsilon$, then $x \in c_0 \cup u_1^+ \cup u_2^+$. By Lemma 17, this implies a contradiction since x can not have been added to s_i^-, whereas x is an element of $x \in c_0 \cup s_1^+ \cup s_2^+$ (see the rules of the CAROT). As a consequence, we have $x \notin u_i^\epsilon$, what implies $x \notin u'^\epsilon_i$. Since we have $x \in s_i'^\epsilon$, this shows that the new configuration has wrong predictions.

We do not detail the case of $s_i^\epsilon \not\subseteq u_i^\epsilon(d, c)$, which can be handled in a similar way. □

E Complements for Subsection 3.3

E.1 Robust model-checking fot Bounded-MTL

Lemma 10. *Let $\mathcal{A}$ be a timed automaton, and $\delta > 0$. Given (ℓ, v) a configuration of $\mathcal{A}$ such that $v \models_\delta I(\ell)$, we have that $\mathcal{L}(\llbracket\mathcal{A}\rrbracket_\delta^{(\ell,v)}) \neq \varnothing$.*

Proof. We have supposed that $\mathcal{A}$ is non-blocking. Hence, for every configuration (ℓ', v') of $\mathcal{A}$ such that $v' \models_0 I(\ell')$, there is an infinite run in $\llbracket\mathcal{A}\rrbracket$ from (ℓ', v'). We fix $\widetilde{v}$ such that $d(v, \widetilde{v}) \leq \delta$ and $\widetilde{v} \models_0 I(\ell)$ (this is possible as $v \models_\delta I(\ell)$).

We will first prove that $\mathcal{L}(\llbracket\mathcal{A}\rrbracket_\delta^{(\ell,\widetilde{v})}) \neq \varnothing$ by exhibiting a time-diverging run in $\llbracket\mathcal{A}\rrbracket_\delta$ from $(\ell, \widetilde{v})$. First, as $\mathcal{A}$ is non-blocking, we know that there exists an infinite run ϱ in $\llbracket\mathcal{A}\rrbracket$ from $(\ell, \widetilde{v})$. If that run is time-diverging, then we are done. Otherwise, ϱ is Zeno run (*i.e.*, a run whose global duration of the run is bounded). This run visits infinitely often twice a state of the region automaton of $\mathcal{A}$. By hypothesis, those cycles are progress cycles. We can then apply [33, Theorem 19 and Lemma 30] and iterate each of these cycles while enforcing time to elapse. That way, we transform ϱ into a time-diverging run ϱ' in $\llbracket\mathcal{A}\rrbracket_\delta$. Moreover, we can assume that we have let unchanged a prefix of ϱ which visits twice a state of the region automaton of $\mathcal{A}$. This prefix contains a cycle. By assumption, this cycle resets at least once every clock. Thus, from (ℓ, v), we can mimic the run ϱ', which will give a time-diverging run from (ℓ, v) in $\llbracket\mathcal{A}\rrbracket_\delta$. This is possible to mimic ϱ' because there is a prefix long enough which is read in $\llbracket\mathcal{A}\rrbracket$ (hence the corresponding prefix from (ℓ, v) will be in $\llbracket\mathcal{A}\rrbracket_\delta$). And after that prefix, we arrive

precisely at the same position (clocks have been reset exactly at the same time in the two runs), thus the rest of ϱ' can be read as well. □

Lemma 12. *Let $N \geq 2 \cdot N_0$.*[13] *Then $\mathcal{P}(N) \Rightarrow \forall n \in \mathbb{N}, \exists n' \geq n$ s.t. $\mathcal{P}(n')$.*

Proof. We consider an h-cycle-bounded accepting computation for $\mathcal{C}_{\mathcal{A},\neg\varphi}$ starting in $(d^{\varphi,n}, c^{\varphi,n})$. This computation is with correct predictions iff the last configuration of the computation is with correct predictions (see Lemma 23). Predictions can thus be checked adding, at the end of the computation, instructions reading the content of the channel up to the second Δ (and checking that the guesses were all correct). This sequence of instructions can be made by an independent cycle-free module where each branch has length at most $5 + 2|X|$ (two read actions $\langle?$, two read actions $\rangle?$, two read actions $\Delta?$, one read actions $x?$ and one read action $\epsilon_x?$ per clock $x \in X$, and one renaming operation). These extra instructions may add one line to the computation table, but not more (we only read and we do not write on the channel).

Consider now the computation table (as in Appendix A) corresponding to this extended computation. The height of this table is at most $h + 1$. We distinguish between two cases:

- either nothing written on the channel during the computation is read (which means that the channel does not cycle). This implies that we can add as many Δ as we want at the tail of the channel and the same sequence of actions will be accepting for these largest numbers of Δ's initially on the channel. In that case, for every $n' \geq n$, $\mathcal{C}_{\mathcal{A},\neg\varphi}$ has an $(h+1)$-cycle-bounded accepting computation from state $(d^{\varphi,n'}, c^{\varphi,n'})$. Moreover, the test for correct predictions is ensured by the last instructions we have added.
- or the whole channel has cycled at least once. This means that on the first line of the computation table, there are $n - 1$ read actions $\Delta?$. Thus, there are n sliding windows whose first line is of the form $s\,\Delta?\,t$ (for some states s and t). We forget sliding windows whose last line contains a read action for the last check. There remains thus at least $n - 2 \cdot |X| - 5$ read actions $\Delta?$ on the first line. Assuming $n \geq N_0$, two such sliding windows are identical. We can thus iterate the part of the sliding window which is inbetween. This part contains at least one $\Delta?$-action. Thus, by iterating this part, we can construct accepting computation tables for initial configurations $(d^{\varphi,n'}, c^{\varphi,n'})$ where n' is arbitrarily large, which means that there is an increasing sequence of indices $(n_i)_{i\geq 1}$ such that $\mathcal{C}_{\mathcal{A},\neg\varphi}$ has an h-cycle-bounded accepting computation from $(d^{\varphi,n_i}, c^{\varphi,n_i})$ for every $i \geq 1$. The last line of the computation table ensures that the computation has correct predictions (no part of the last check module can have been iterated as there is no cycle in this module). □

Corollary 13. *Let $N \geq 2 \cdot N_0$. $\mathcal{C}_{\mathcal{A},\neg\varphi}$ has an h-cycle-bounded accepting computation with correct predictions starting in $(d^{\varphi,N}, c^{\varphi,N})$ if, and only if, for*

[13] A more tighten lower bound should be $N_0 + 5 + 2 \cdot |X|$, but such a bound would be mysterious in the core of the paper. As N_0 is quite large, we of course have that $2 \cdot N_0 \geq N_0 + 5 + 2 \cdot |X|$.

every $n \geq 3$, $\mathcal{C}_{\mathcal{A},\neg\varphi}$ has an h-cycle-bounded accepting computation with correct predictions starting in $(d^{\varphi,n}, c^{\varphi,n})$.

Proof. This is a consequence of Lemmas 11 and 12. Indeed, for every $n \geq 3$, there is some $n' \geq \max(2n, N)$ such that $\mathcal{C}_{\mathcal{A},\neg\varphi}$ has an h-cycle-bounded accepting computation with correct predictions starting in $(d^{\varphi,n'}, c^{\varphi,n'})$ (Lemma 12). Applying Lemma 11, we get that $\mathcal{C}_{\mathcal{A},\neg\varphi}$ has an h-cycle-bounded accepting computation with correct predictions starting in $(d^{\varphi,n}, c^{\varphi,n})$. □

E.2 Robust model-checking fot coFlat-MTL

In the core of the paper, we have only given very brief explanations about our algorithm for solving the robust model-checking problem for coFlat-MTL. We give here full details on this algorithm.

We first define useful objects and fix some notations:

- The alternating timed automaton $\mathcal{A}_{\neg\varphi}$ (see [36] for its construction) accepts all infinite timed words that do not satisfy φ. Along an execution of $\mathcal{A}_{\neg\varphi}$, some parts only verify untimed properties (cf. *inactive parts* in Lemma 25), and those then correspond to a behaviour of a finite automaton. The states of $\mathcal{A}_{\neg\varphi}$ are subformulae of $\neg\varphi$, and these inactive parts correspond to move in $\mathcal{A}_{\neg\varphi}$ where the current slice of the computation only contains *inactive* (or LTL) formulae (inactive formulae are formulae in which the outermost modality is unbounded, but the current value of the clock is $\bot$, *i.e.* above the maximal constant). Moreover, along those inactive parts, the channel contains no formula information (because no timing information is relevant for the formulae), but only timing informations for the automaton $\mathcal{A}$. Everything concerning formulae is stored in the discrete state. Thus, we construct a finite automaton $\mathcal{F}_{\neg\varphi}$ which will "play the role" of $\mathcal{A}_{\neg\varphi}$ along the inactive parts. The construction is made in two steps:
  - A state is a set Ψ such that

    $$\Psi \subseteq \{\psi \text{ LTL subformula of } \neg\varphi\} \cup \{\psi \text{ subformula of } \neg\varphi \text{ with an unbounded outermost modality}\}.$$

    There is a transition $\Psi_1 \xrightarrow{\sigma} \Psi_2$ whenever there is a σ-move allowed between configuration Ψ_1 and Ψ_2 in $\mathcal{A}_{\neg\varphi}$.
  - Then, we apply the Miyano-Hayashi construction [34, 37] to handle the accepting condition of $\mathcal{A}_{\neg\varphi}$ (we know that a computation of $\mathcal{A}_{\neg\varphi}$ will eventually end up by an inactive part).

  The resulting automaton has size at most exponential in the size of $\mathcal{A}_{\neg\varphi}$, hence exponential in $|\varphi|$, and has a Büchi acceptance condition.
- Based on ideas of [32], we proposed in [30, 31] an extended region automaton which is correct for robustly checking reachability, safety and LTL properties. We assume the reader is familiar with the classical region automaton

construction [27], and if $\mathcal{A}$ is a timed automaton, we write $\mathcal{R}(\mathcal{A})$ for its classical region automaton. The extended region automaton $\mathcal{R}^*(\mathcal{A})$ has the same transitions as $\mathcal{R}(\mathcal{A})$ and has extra transitions of the form $(\ell, r) \xrightarrow{\gamma} (\ell, r')$ whenever (ℓ, r') is in an SCC of $\mathcal{R}(\mathcal{A})$ and $\overline{r} \cap \overline{r'} \neq \varnothing$ (where $\overline{\cdot}$ represents the topological closure).
- In the following, we use the ∞-norm to measure the distance between valuations: given two valuations $u, v \in \mathbb{R}^X$, we define $d(u, v) = \max_{x \in X} |u(x) - v(x)|$, and if V is a subset of $\mathbb{R}^X$, $d(u, V) = \sup_{v \in V} d(u, v)$. We extend this definition in a straightforward way to configurations of a timed automaton.
- Let $\mathcal{A}$ be a timed automaton. For every $0 < \delta < 1$, for every $k \in \mathbb{N}$, we write $f_k(\delta) = \sqrt{18(|X|+1)^{k+2}\delta}$. This constant has been introduced in [33, Theorem 38] for measuring the distance between paths in $[\![\mathcal{A}]\!]_\delta$ of length less than k and real paths in $[\![\mathcal{A}]\!]$. Indeed, assuming that $f_k(\delta) < 1$, for every run ϱ in $[\![\mathcal{A}]\!]_\delta$ of length less than k, there exists a run ϱ' in $[\![\mathcal{A}]\!]$ such that $d(\varrho, \varrho') \leq f_k(\delta)$.[14]

Our algorithm for robustly model-checking coFlat-MTL is presented as Algorithm 1 and illustrated on Figure 7.

Theorem 24. *Given $\mathcal{A}$ a timed automaton and φ a* coFlat-MTL *formula, Algorithm 1 decides whether $\mathcal{A} \models \varphi$.*

Proof. We assume that $\mathcal{A} \not\models \varphi$. For every $\delta > 0$, there exists a run ϱ^δ in $[\![\mathcal{A}]\!]_\delta$ such that $\varrho^\delta \not\models \varphi$. W.l.o.g. we assume that ϱ^δ already corresponds to a run of the product $[\![\mathcal{A}]\!]_\delta \times [\![\mathcal{A}_{\neg\varphi}]\!]$ (concretising the fact that $\varrho^\delta \not\models \varphi$). First notice that

$$\forall \delta' \leq \delta,\ \varrho^{\delta'} \in [\![\mathcal{A}]\!]_{\delta'} \times [\![\mathcal{A}_{\neg\varphi}]\!] \subseteq [\![\mathcal{A}]\!]_\delta \times [\![\mathcal{A}_{\neg\varphi}]\!]. \tag{1}$$

Twisting a little bit the rather involved decomposition theorem [29, Theorem 12], we can split executions into "active" and "inactive" parts as follows.

Lemma 25. *Let ϖ be a run of $[\![\mathcal{A}]\!]_\delta \times [\![\mathcal{A}_{\neg\varphi}]\!]$ not satisfying φ. Then it can be written as $\varpi_1 \cdot \varpi_2 \cdot \varpi_3 \cdots \varpi_{2m}$ where:*

(i) the duration of ϖ_{2i-1} (for $1 \leq i \leq m$) is bounded by $(2M + 3 + W \cdot 2^{|\varphi|}) \cdot (|\varphi| \cdot 2^{|\varphi|})$,
(ii) the duration of ϖ_{2i} (for $1 \leq i \leq m$) is at least $2^{|\varphi|} \cdot (2W + 1)$, and along that portion, the behaviour of $\mathcal{A}_{\neg\varphi}$ is that of $\mathcal{F}_{\neg\varphi}$,
(iii) the Büchi condition of $\mathcal{F}_{\neg\varphi}$ is satisfied along ϖ_{2m}, and
(iv) $2m \leq |\varphi| \cdot 2^{|\varphi|}$.

Odd-numbered segments are called the active parts*, while even-numbered ones are said to be* inactive*.*

We split each of the runs ϱ^δ into active and inactive parts as in Lemma 25. Since we have infinitely many runs and a finite bound on the maximum number of

[14] Meaning that the two paths have the same length, and for any corresponding two states α and α' along ϱ and ϱ', $d(\alpha, \alpha') \leq f_k(\delta)$.

Algorithm 1: Non-deterministic algorithm for coFlat-MTL robust model-checking

Input: A timed automaton $\mathcal{A}$ and a coFlat-MTL formula φ

$W =$ number of states of $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$;
$N_1 = N_0 + (|X| + 1) \cdot f_{2W+1}\left(\frac{2}{N_0+1}\right)$;
// where N_0 is the iteration constant of Lemma 12
guess $m \leq |\varphi| \cdot 2^{|\varphi|-1}$; // the number of inactive parts
set (s_0, S_0) with $s_0 = S_0$ as the initial state of $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$;
for *i from* 1 *to* $m-1$ **do**
  begin
    guess a tuple $(r_{2i}, R_{2i}, s_{2i}, S_{2i}, T_{2i})$ of states of $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$;
    check that R_{2i} belongs to an SCC of $\mathcal{R}(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$;
    check that T_{2i} belongs to an SCC of $\mathcal{R}(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$;
    check that T_{2i} is reachable from R_{2i} in $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$;
    check that S_{2i} is reachable from T_{2i} in $\mathcal{R}(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$ through a path ν of length no more than W;
    check using PDBMs that the part of s_{2i} close to S_{2i} is reachable from T_{2i} along ν;
    guess a $(h + M \cdot W)$-cycle bounded computation table[15] for the CAROT $\mathcal{C}_{\mathcal{A},\neg\varphi}$ for N_0 symbols Δ from s_{2i-2} (close to S_{2i-2}) to r_{2i} (close to R_{2i}), and with correct predictions;
      // Close meaning at distance no more than $f_{2W+1}\left(\frac{2}{N_1+1}\right)$
  end
end
guess a tuple (r_{2m}, R_{2m}) of states of $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$;
check that R_{2m} belongs to an SCC of $\mathcal{R}(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$;
guess a $(h + M \cdot W)$-cycle bounded computation table with correct predictions for the CAROT $\mathcal{C}_{\mathcal{A},\neg\varphi}$ for N_0 symbols Δ from s_{2m-2} (close to S_{2m-2}) to r_{2m} (close to R_{2m});
check that there exists a path in $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$, starting in R_{2m}, and satisfying the Büchi condition of $\mathcal{F}_{\neg\varphi}$;
if *all checks correct* **then**
  return *no*; // φ does not hold robustly
end

[15] With $h = (2M + 3 + W \cdot 2^{|\varphi|}) \cdot (|\varphi| \cdot 2^{|\varphi|})$, following Lemma 25.

active and inactive parts, for some integer k, there are infinitely many executions having that number of active and inactive parts. From (1), we can assume that all the runs have exactly that number of parts.

We write ϱ_i^δ for the i-th fragment of ϱ^δ. As ϱ_{2i}^δ is inactive, the portion of $\mathcal{A}_{\neg\varphi}$ used along that fragment corresponds to the behaviour of the finite automaton $\mathcal{F}_{\neg\varphi}$, and ϱ_{2i}^δ can then be viewed as a run in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_\delta$. Applying the construction of [30, Theorem 4], for each i, ϱ_{2i}^δ can be approximated by a

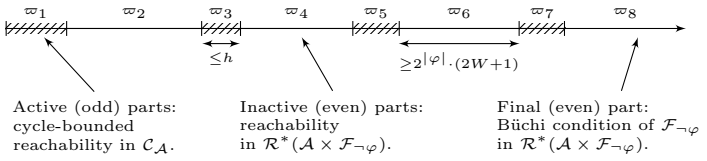

Fig. 7. Global view of our algorithm

path ν^δ_{2i} in $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$, whose underlying trace, when removing γ-transitions, corresponds to the trace underlying the run ϱ^δ_{2i}. Moreover, the run ϱ^δ_{2i} remains at distance at most $f_{2W+1}(\delta)$ from ν^δ_{2i}, where W is the number of states of $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$. Looking more carefully at the construction, we also notice that we can find a prefix of ν^δ_{2i}, say μ^δ_{2i}, of length bounded by W which is entirely a path of $\mathcal{R}(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$ (*i.e.*, without any γ-transition), and which ends up in an SCC of $\mathcal{R}(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$. This part is rather small, and we assume it is part of the previous active part. It may increase the previous active part by at most $M \cdot W$ time units. If the first portion of ϱ^δ was inactive (*i.e.*, $\varrho^\delta_0 = \varnothing$), then it may create an initial small "active" part. The decomposition is illustrated on Figure 8.

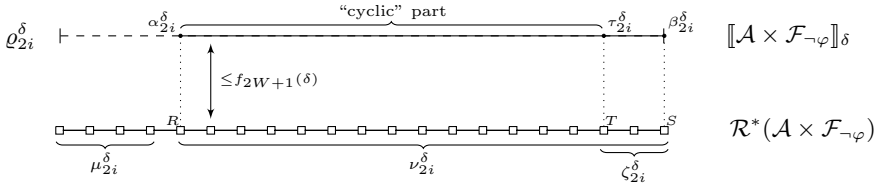

Fig. 8. Decomposition of an inactive part

So, we now assume that ν^δ_{2i} starts in an SCC. Assume $i < m$. We write R^δ_{2i} (resp. S^δ_{2i}) the first (resp. last) region of ν^δ_{2i}. We also note α^δ_{2i} (resp. β^δ_{2i}) the first (resp. last) point of ϱ^δ_{2i}. We write $r^\delta_{2i} = [\alpha^\delta_{2i}]$ and $s^\delta_{2i} = [\beta^\delta_{2i}]$. We have that $d(\alpha^\delta_{2i}, R^\delta_{2i}) \leq f_{2W+1}(\delta)$ and $d(\beta^\delta_{2i}, S^\delta_{2i}) \leq f_{2W+1}(\delta)$. Looking once more more carefully to the construction of [30, Theorem 4], there is a region T^δ_{2i} along ν^δ_{2i} which is in an SCC of $\mathcal{R}(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$ and such that the suffix of ν^δ_{2i} starting in T^δ_{2i} is acyclic and γ-free (in particular, it has length less than W). We write this suffix ζ^δ_{2i}. We write τ^δ_{2i} the point of ϱ^δ_{2i} corresponding to that region. We also have that $d(\tau^\delta_{2i}, T^\delta_{2i}) \leq f_{2W+1}(\delta)$.

For $i = m$, we write R^δ_{2m} the first region of ν^δ_{2m} and α^δ_{2m} the first point of ϱ^δ_{2m}. We write $r^\delta_{2m} = [\alpha^\delta_{2m}]$. We have that $d(\alpha^\delta_{2m}, R^\delta_{2m}) \leq f_{2W+1}(\delta)$.

We can find $(r_{2i}, s_{2i}, R_{2i}, S_{2i}, T_{2i}, \zeta_{2i})_{1 \leq i < m}$ and (r_m, R_m) characterising ϱ^δ for a sequence of δ's converging to 0. From (1), we can assume that all the runs ϱ^δ are characterised by these tuples. By construction, there is a path in $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$

from R_{2i} to S_{2i}, and for every $\delta > 0$, $\alpha_{2i}^{\delta} \in r_{2i}$, $\beta_{2i}^{\delta} \in s_{2i}$, $d(\alpha_{2i}^{\delta}, R_{2i}) \leq f_{2W+1}(\delta)$ and $d(\beta_{2i}^{\delta}, S_{2i}) \leq f_{2W+1}(\delta)$. Furthermore, T_{2i} satisfies the expected requirements (as there is a run in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_{\delta}$ from τ_{2i}^{δ} to β_{2i}^{δ} following the trace of ζ_{2i}, it means that there exists $\tau \in T_{2i}$ such that there is a run in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_{\delta+f_{2W+1}(\delta)}$ from τ to β_{2i}^{δ}).

In addition, the active parts, which correspond to the cycle-bounded computations of the CAROT $\mathcal{C}_{\mathcal{A},\neg\varphi}$, can be guessed as claimed. This is due to results obtained in the previous sections, namely the simulation of $[\![\mathcal{A}]\!]_{\frac{1}{N}}$ by $\mathcal{B}^n$ (Lemma 2) and the time-abstract bisimulation between $\mathcal{T}^n_{\mathcal{C}_{\mathcal{A}}}$ and $\mathcal{T}^n_H$ (Lemma 6). Together, these results indeed show that the existence of an execution of duration bounded by $h + M \cdot W$ in $[\![\mathcal{A}]\!]_{\delta}$ for arbitrarily small δ from $\beta_{2i-1}^{\delta} \in s_{2i-1}$ to $\alpha_{2i}^{\delta} \in r_{2i}$, with previous conditions of closeness on these points, imply the existence of an $(h + M \cdot W)$-cycle-bounded computation, with correct predictions, with N_1 symbols Δ, between the two configurations of the CAROT corresponding to these points. This new bound N_1 is required, so that if, on the channel, two clocks are close (*i.e.*, separated by no more than $f_{2W+1}\left(\frac{2}{N_1+1}\right)$ Δ's, then we can choose where we iterate the computation table and we can choose it in such a way that, by iterating, those clocks remain close, and their relative distance converges to 0. That way, closeness to regions S_{2i-1} and R_{2i} can be preserved.

Finally, we consider the last unbounded even part. We must show that there exists a path in the automaton $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$, starting in R_{2m} and verifying the Büchi condition of $\mathcal{F}_{\neg\varphi}$. For every $\delta > 0$, there is a path ϱ_{2m}^{δ} in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_{\delta}$ satisfying the Büchi condition of $\mathcal{F}_{\neg\varphi}$. Denoting by α_{2m}^{δ} the first point of ϱ_{2m}^{δ}, we have that α_{2m}^{δ} is close from region R_{2m}. Since R_{2m} belongs to an SCC of $\mathcal{R}(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$, we can build, from any point $\alpha_{2m} \in R_{2m}$, and for every $\delta > 0$, a path in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_{\delta}$ starting in α_{2m} and verifying the Büchi condition of $\mathcal{F}_{\neg\varphi}$. This means that region R_{2m} does not robustly satisfy the acceptance condition of $\mathcal{F}_{\neg\varphi}$ interpreted as a co-Büchi condition in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]$. This last property is equivalent, by the results of [31], to the existence, in $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$, of a path starting in R_{2m} and verifying the Büchi condition of $\mathcal{F}_{\neg\varphi}$.

We will now prove the converse, that is, if the algorithm returns "no", then indeed $\mathcal{A} \not\models \varphi$. First we state some useful lemmas.

Lemma 26 ([30, Lemma 7]). *Assume that R is a state of $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$ belonging to an SCC of $\mathcal{R}(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$, and that S is a state of $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$ that can be reached from R in $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$. Then, for every $\delta > 0$, for every $\alpha \in R$, for every $\beta \in S$, there is a run in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_{\delta}$ from α to β.*

Moreover, we can get the following result:

Lemma 27. *For every $\delta, \delta' > 0$, for every α, α' such that $d(\alpha, \alpha') \leq \delta'$, for every β, if there is a run in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_{\delta}$ from α to β which resets every clock at least once, then there is a run in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_{\delta+\delta'}$ from α' to β.*

Proof. It is sufficient to mimic the run from α to β (same delays, same transitions). Because of the larger enlargement of the guards, we are never blocked. Moreover, as every clock is reset at least once, after a while the two runs will be the same. □

Lemma 28. *Assume that R is a state of $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$, and that S is a state of $\mathcal{R}(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$ that can be reached from R in $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$ along a path ν of length at most W. Fix $0 < \delta < \frac{1}{2(1+|X|)^{W+2}}$. Assume that there is $\alpha \in R$ and β such that $d(\beta, S) \leq \delta$ and there is a run ϱ in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_\delta$ from α to β along the trace of ν. Then, for every $0 < \delta' < \frac{1}{2(1+|X|)^{W+2}}$, for every $\alpha' \in R$, for every $\beta' \in \overline{[\beta]}$ such that $d(\beta', S) \leq \delta'$, there is a run ϱ' in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_{\delta'}$ from α' to β' along the trace of ν.*

Proof. In the case of guards enlargement only, PDBMs can be used to do exact computations (see [33, Lemmas 39-42]). Hence, we get that for every $0 < \delta' < \frac{1}{2(1+n)^{W+2}}$, for every $\beta' \in \overline{[\beta]}$ such that $d(\beta', S) \leq \delta'$), for every $\alpha' \in R$, there is a run from α' to β' in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_{\delta'}$. □

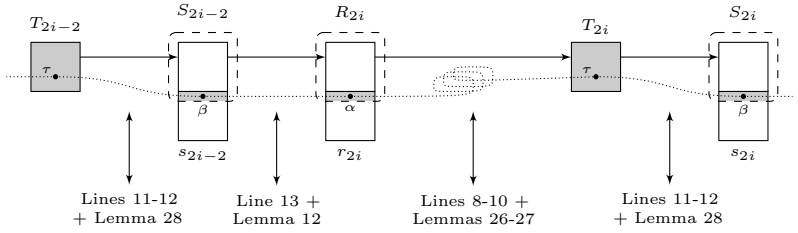

Fig. 9. Building a witnessing path ϱ^δ from Algorithm 1

Choose m and tuples $(r_{2i}, R_{2i}, s_{2i}, S_{2i}, T_{2i})_{1 \leq i < m}$ and (r_m, R_m) satisfying the requirement of the algorithm. We write ζ_{2i} for the path in $\mathcal{R}(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$ from T_{2i} to S_{2i} mentioned in the algorithm (it has length no more than W). For every $1 \leq i \leq m$, guess computation tables with N_0 symbols Δ such that the initial state is in s_{2i-2} and close to S_{2i-2} and the final state is in r_{2i} and close to R_{2i} (*i.e.*, at distance at most $f_{2W+1}\left(\frac{2}{N_0+1}\right)$). By iterating in the computation table (see Lemma 12), for every $\varepsilon > 0$, there exists N_ε, such that for every $N \geq N_\varepsilon$, putting N symbols Δ, there is a state $\beta_{2i-2} \in s_{2i-2}$ such that $d(\beta_{2i-2}, S_{2i-2}) \leq \varepsilon$ and a state $\alpha_{2i} \in r_{2i}$ such that $d(\alpha_{2i}, R_{2i}) \leq \varepsilon$, and there is an $(h + M \cdot W)$-cycle bounded computation of the CAROT from β_{2i-2} to α_{2i}.

Applying Lemma 26, for every $\alpha'_{2i} \in R_{2i}$, for every $\tau_{2i} \in T_{2i}$, there is a run in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_\varepsilon$ from α'_{2i} to τ_{2i}. Applying Lemma 27, there is a run in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_{2\varepsilon}$ from α_{2i} to τ_{2i}. Applying Lemma 28, there is a run in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_\varepsilon$ from τ_{2i} to β_{2i}. Hence, there is a run in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_{2\varepsilon}$ from α_{2i} to β_{2i}. The construction is illustrated on Figure 9.

Gluing all these paths together, we obtain, for every $\delta > 0$, a run in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_\delta$ reaching state $\alpha_{2m} \in r_{2m}$, close to R_{2m}. There remains to build an infinite time-divergent tail satisfying the Büchi condition of $\mathcal{F}_{\neg\varphi}$. Since there exists a path in $\mathcal{R}^*(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$ starting in R_{2m} and verifying the Büchi acceptance condition

of $\mathcal{F}_{\neg\varphi}$, there exists, for every $\delta > 0$, a path in $[\![\mathcal{A} \times \mathcal{F}_{\neg\varphi}]\!]_\delta$ starting in R_{2m} and verifying the Büchi condition of $\mathcal{F}_{\neg\varphi}$. Since R_{2m} belongs to an SCC of $\mathcal{R}(\mathcal{A} \times \mathcal{F}_{\neg\varphi})$, it is possible to glue the previous path, reaching state $\alpha_{2m} \in r_{2m}$, close to R_{2m}, to this new path (because of the enlarged semantics). However, the path verifying the Büchi condition may not be time-divergent. Using the same arguments as in the proof of Lemma 10, one can show that it is possible to build from this path another one which also satisfies the Büchi condition (because we only eventually duplicate some cycles), and which is time-divergent. Finally, for every $\delta > 0$, we can construct a run ϱ^δ not satisfying φ. Hence, $\mathcal{A} \not\models \varphi$. □

References

27. R. Alur and D. Dill. A theory of timed automata. *Theor. Comp. Sci.*, 126(2):183–235, 1994.
28. P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. The cost of punctuality. Research Report LSV-07-24, ENS Cachan, France, 2007.
29. P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. The cost of punctuality. In *Proc. 22nd Ann. Symp. Logic in Computer Science (LICS'07)*. IEEE Comp. Soc. Press, 2007. 109–118.
30. P. Bouyer, N. Markey, and P.-A. Reynier. Robust model-checking of timed automata. Research Report LSV-05-06, ENS Cachan, France, 2005.
31. P. Bouyer, N. Markey, and P.-A. Reynier. Robust model-checking of timed automata. In *Proc. 7th Latin American Symp. Theoretical Informatics (LATIN'06)*, LNCS 3887, pages 238–249. Springer, 2006.
32. M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. In *Proc. Joint Conf. Formal Modelling and Analysis of Timed Systems & Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, LNCS 3253, pages 118–133. Springer, 2004.
33. M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. Tech. Report 2004.30, Centre Fédéré en Vérification, Belgium, Dec. 2005. Revised version.
34. S. Miyano and T. Hayashi. Alternating finite automata on omega-words. *Theoretical Computer Science*, 32:321–330, 1984.
35. J. Ouaknine and J. Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Comp. Sci.*, 2007. To appear.
36. J. Ouaknine and J. B. Worrell. On the decidability of metric temporal logic. In *Proc. 19th Ann. Symp. Logic in Computer Science (LICS'05)*, pages 188–197. IEEE Comp. Soc. Press, 2005.
37. M. Y. Vardi. An automata-theoretic approach to Linear Temporal Logic. In *Proc. Logics for Concurr.: Structure versus Automata*, LNCS 1043, pages 238–266. Springer, 1996.