On alternating timed automata for MITL

Thomas Brihaye* Morgane Estiévenart† Gilles Geeraerts‡

June 2014

Abstract

Motivated by the MITL model-checking problem, we define a new semantics for *one clock alternating timed automata* (OCATA). In this paper, we expose this new semantics and show how it is used to build Büchi timed automata equivalent to OCATA emanating from MITL formulas. As a byproduct of our work, we identified an easily complementable class of OCATA with Büchi acceptance conditions.

1 Introduction

Automata-based model-checking [6, 15] is nowadays a prominent technique for establishing the correctness of computer systems. In this framework, the system to analyse is modeled by means of a *Büchi automaton* (BA) $\mathcal{B}$ and the *property* to prove is usually expressed using a *linear temporal logic (LTL) formula* φ. Establishing correctness of the system amounts to showing that the language $L(\mathcal{B})$ of the system is included in the language of a BA $\mathcal{A}_\varphi$ recognizing the language $[\![\varphi]\!]$ of the formula. One of the most efficient techniques to perform LTL model-checking is to first construct from the LTL formula φ an *alternating Büchi automaton* (ABA) recognizing $[\![\varphi]\!]$ which is then turned to a BA accepting the same language. This translation is possible thanks to the well-known subset construction due to Miyano and Hayashi [12].
While those techniques are now routinely used, the model of BA and the LTL logic are sometimes not *expressive* enough because they can model the *possible sequences of events* (for instance, the LTL formula $\Box(p \Rightarrow \Diamond q)$ says that every p-event should eventually be followed by a q-event), but cannot express *quantitative properties about the (real) time elapsing between successive events.* To overcome these weaknesses, Alur and Dill [1] have proposed the model of *Büchi timed automata* (BTA), that extends BA with a finite set of (real valued) clocks. A real-time extension of LTL called *Metric Temporal Logic* (MTL) was proposed by Koymans [9]. Unfortunately, its model-checking problem is undecidable so that some of MTL fragments were studied. In this paper we focus on a decidable syntactic fragment of MTL, called *Metric Interval Temporal Logic* (MITL), proposed by Henzinger *et al.* in [3]. With this logic, we can express real-time property such as $\Box(p \Rightarrow \Diamond_{[1,2]} q)$, which means 'all p-event must be followed by a q-event that occurs *between* 1 *and* 2 *time units later*'.
The techniques developed on BA and ABA to perform LTL model-checking rely on nice properties which cannot be generalized to the timed framework. For example, in general, BTA are not determinizable [1] (the problem of the determinizability is even undecidable [7]) and not complementable [1] ; the universality and hence the emptiness language problem are undecidable for *alternating Büchi timed automata* (ABTA) [1].

*Département de Mathématiques, Université de Mons (UMONS), Belgium.
†Département de Mathématiques, Université de Mons (UMONS), Belgium, this author has been supported by a FRIA scholarship.
‡Département d'Informatique, Université Libre de Bruxelles (U.L.B.), Belgium.

In the case of an MTL formula φ, Ouaknine and Worrell [13] showed how to construct a *one-clock alternating timed automaton* (OCATA) $\mathcal{A}_\varphi$ recognizing the language $[\![\varphi]\!]$ of φ on finite words: we extend this result to the scope of infinite words [5]. Unfortunately, in general, the translation from OCATA to Büchi timed automata (BTA) is impossible. Indeed, a run of an alternating timed automaton can be understood as several copies of the same automaton running in parallel on the same word, but whose clock values are not always synchronized. So, the number of different clock values tracked along a run cannot be bounded and hence, the Miyano Hayashi construction [12] cannot be directly applied.
In this paper, we present a novel semantics for OCATA [4, 5], that we call the *interval semantics*, where clock valuations are not punctual values but intervals with real endpoints. One of the features of this semantics is that several clock values can be grouped into intervals, thanks to a so-called *(bounded) approximation function*. This semantics enables us to bound the number of clock copies used by OCATA using *bounded approximation functions*: the price to pay is that in general the resulting language is only an *under-approximation* of the starting one.
Nevertheless, we identify a class of OCATA $\mathcal{A}$ for which there exist a family of bounded approximations functions f such that $L^\omega_f(\mathcal{A}) = L^\omega(\mathcal{A})$. It is the class of OCATA A_φ, obtained from MITL formulas φ. Using the Miyano Hayashi construction, we can then translate these OCATA into BTA accepting the same language.
As a byproduct of our work, we identify the easily complementable class of *tree-like OCATA* (TOCATA), a subclass of OCATA that exhibits some structure akin to a tree (in the same spirit as the Weak and Very Weak Alternating Automata [10, 8]) and which subsumes the class of OCATA $\mathcal{A}_\varphi$ constructed as in [14], for every MTL formula φ.

2 Preliminaries

Basic notions. Let $\mathbb{R}$, $\mathbb{R}^+$ and $\mathbb{N}$ denote the sets of real, non-negative real and natural numbers respectively. We call ***interval*** a convex subset of $\mathbb{R}$. We rely on the classical notation $\langle a, b\rangle$ for intervals, where $\langle$ is (or [, $\rangle$ is) or], $a \in \mathbb{R}$ and $b \in \mathbb{R} \cup \{+\infty\}$. For an interval $I = \langle a, b\rangle$, we let $\inf(I) = a$ be the *infimum* of I, $\sup(I) = b$ be its *supremum* (a and b are called the *endpoints* of I) and $|I| = \sup(I) - \inf(I)$ be its *length*. We note $\mathcal{I}(\mathbb{R})$ the set of all intervals. We note $\mathcal{I}(\mathbb{R}^+)$ (resp. $\mathcal{I}(\mathbb{N}^{+\infty})$) the set of all intervals whose endpoints are in $\mathbb{R}^+$ (resp. in $\mathbb{N} \cup \{+\infty\}$). Let $I \in \mathcal{I}(\mathbb{R})$ and $t \in \mathbb{R}$, we note $I + t$ for $\{i + t \in \mathbb{R} \mid i \in I\}$.

Let Σ be a finite alphabet. An *infinite word* on a set S is an infinite sequence $s = s_1 s_2 s_3 \ldots$ of elements in S. An infinite time sequence $\bar{\tau} = \tau_1 \tau_2 \tau_3 \ldots$ is an infinite word on $\mathbb{R}^+$ s.t. $\forall i \in \mathbb{N}, \tau_i \leq \tau_{i+1}$. An *infinite timed word* over Σ is a pair $\theta = (\bar{\sigma}, \bar{\tau})$ where $\bar{\sigma}$ is an infinite word over Σ, $\bar{\tau}$ an infinite time sequence. We also note θ as $(\sigma_1, \tau_1)(\sigma_2, \tau_2)(\sigma_3, \tau_3)\ldots$. We note $T\Sigma^\omega$ the set of all infinite timed words. A *timed language* is a (possibly infinite) set of infinite timed words.

Alternating timed automata. Let $\Gamma(L)$ be a set of formulas of the form $\top$, or $\bot$, or $\gamma_1 \vee \gamma_2$ or $\gamma_1 \wedge \gamma_2$ or ℓ or $x \bowtie c$ or $x.\gamma$, with $c \in \mathbb{N}$, $\bowtie \in \{<, \leq, >, \geq\}$, $\ell \in L$. We call $x \bowtie c$ a *clock constraint*. Then, a *one-clock alternating timed automaton* (OCATA) [14] is a tuple $\mathcal{A} = (\Sigma, L, \ell_0, F, \delta)$ where Σ is a finite alphabet, L is a finite set of locations, ℓ_0 is the initial location, $F \subseteq L$ is a set of accepting locations, $\delta : L \times \Sigma \to \Gamma(L)$ is the transition function. Intuitively, disjunctions in $\delta(\ell)$ model non-determinism, conjunctions model the creation of several automaton copies running in parallel (that must all accept for the word to be accepted) and $x.\gamma$ means that the clock x is reset when taking the transition.

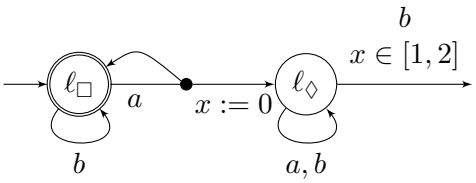

Figure 1: OCATA $\mathcal{A}_\varphi$ with $\varphi = \Box(a \Rightarrow \Diamond_{[1,2]} b)$.

3 The intervals semantics for OCATA on infinite words

The *standard* semantics for OCATA [13, 11] is defined as an infinite transition system whose *configurations* are finite sets of pairs (ℓ, v), where ℓ is a location and v is the valuation of the (unique) clock. Intuitively, each configuration thus represents the current state of all the copies (of the unique clock) that run in parallel in the OCATA. In this section, we introduce a *novel* semantics for OCATA, in which *configurations* are sets of *states* (ℓ, I), where ℓ is a location of the OCATA and I is an *interval*, instead of a single point in $\mathbb{R}^+$. Intuitively, a state (ℓ, I) is an abstraction of all the states (ℓ, v) with $v \in I$, in the standard semantics. We further introduce the notion of *approximation function*. Roughly speaking, an approximation function associates with each configuration C (in the interval semantics), a set of configurations that *approximates* C (in a sense that will be made precise later), *and contains less states* than C.

Formally, a *state* of an OCATA $\mathcal{A} = (\Sigma, L, \ell_0, F, \delta)$ is a pair (ℓ, I) where $\ell \in L$ and $I \in \mathcal{I}(\mathbb{R}^+)$. We note $S = L \times \mathcal{I}(\mathbb{R}^+)$ the state space of $\mathcal{A}$. When $I = [v, v]$, we shorten (ℓ, I) by (ℓ, v). A *configuration* of an OCATA $\mathcal{A}$ is a (possibly empty) finite set of states of $\mathcal{A}$ in which all intervals associated with the same location are disjoint. In the rest of the paper, we sometimes see a configuration C as a function from L to $2^{\mathcal{I}(\mathbb{R}^+)}$ s.t. for all $\ell \in L$: $C(\ell) = \{I \mid (\ell, I) \in C\}$. We note $\mathrm{Config}(\mathcal{A})$ the set of all configurations of $\mathcal{A}$. The *initial configuration* of $\mathcal{A}$ is $\{(\ell_0, 0)\}$. For a configuration C and a delay $t \in \mathbb{R}^+$, we note $C + t$ the configuration $\{(\ell, I + t) \mid (\ell, I) \in C\}$. Let E be a finite set of intervals from $\mathcal{I}(\mathbb{R}^+)$. We let $\|E\| = |\{[a, a] \in E\}| + 2 \times |\{I \in E \mid \inf(I) \neq \sup(I)\}|$ denote the number of *individual clocks* we need to encode all the information present in E, using one clock to track singular intervals, and two clocks to retain $\inf(I)$ and $\sup(I)$ respectively for non-singular intervals I. For a configuration C, we let $\|C\| = \sum_{\ell \in L} \|C(\ell)\|$.

Interval semantics. Let $M \in \mathrm{Config}(\mathcal{A})$ be a configuration of an OCATA $\mathcal{A}$, and $I \in \mathcal{I}(\mathbb{R}^+)$. We define the satisfaction relation "$\models_I$" on $\Gamma(L)$ as:

$M \models_I \top$		$M \models_I \ell$	iff $(\ell, I) \in M$
$M \models_I \gamma_1 \wedge \gamma_2$	iff $M \models_I \gamma_1$ and $M \models_I \gamma_2$	$M \models_I x \bowtie c$	iff $\forall x \in I, x \bowtie c$
$M \models_I \gamma_1 \vee \gamma_2$	iff $M \models_I \gamma_1$ or $M \models_I \gamma_2$	$M \models_I x.\gamma$	iff $M \models_{[0,0]} \gamma$

A configuration M is a *minimal model* of the formula $\gamma \in \Gamma(L)$ wrt $I \in \mathcal{I}(\mathbb{R}^+)$ iff $M \models_I \gamma$ and there is no $M' \subsetneq M$ s.t. $M' \models_I \gamma$. Intuitively, for $\ell \in L, \sigma \in \Sigma$ and $I \in \mathcal{I}(\mathbb{R}^+)$, a minimal model of $\delta(\ell, \sigma)$ wrt I represents a (minimal) configuration the automaton can reach from state (ℓ, I) by reading σ. Observe that the definition of $M \models_I x \bowtie c$ only allows to take a transition $\delta(\ell, \sigma)$ from state (ℓ, I) if all the values in I satisfy the clock constraint $x \bowtie c$ of $\delta(\ell, \sigma)$. We denote $\mathsf{Succ}((\ell, I), \sigma) = \{M \mid M \text{ is a minimal model of } \delta(\ell, \sigma) \text{ wrt } I\}$. We lift the definition of Succ to configurations C as follows: $\mathsf{Succ}(C, \sigma)$ is the set of all configurations C' of the form $\cup_{s \in C} M_s$, where for all $s \in C$: $M_s \in \mathsf{Succ}(s, \sigma)$. That is, each $C' \in \mathsf{Succ}(C, \sigma)$ is obtained by choosing one minimal model M_s in $\mathsf{Succ}(s, \sigma)$ for each $s \in C$, and taking the union of all those M_s.

Example: Let us consider the OCATA of Fig. 1, and let us compute the minimal models of $\delta(\ell_\Diamond, b) = \ell_\Diamond \vee (x \geq 1 \wedge x \leq 2)$ wrt to $[1.5, 2]$. A minimal model of $\ell_\Diamond$ wrt $[1.5, 2]$ is $M_1 = \{(\ell_\Diamond, [1.5, 2])\}$. A minimal model of $(x \geq 1 \wedge x \leq 2)$ is $M_2 = \emptyset$ since all values in $[1.5, 2]$

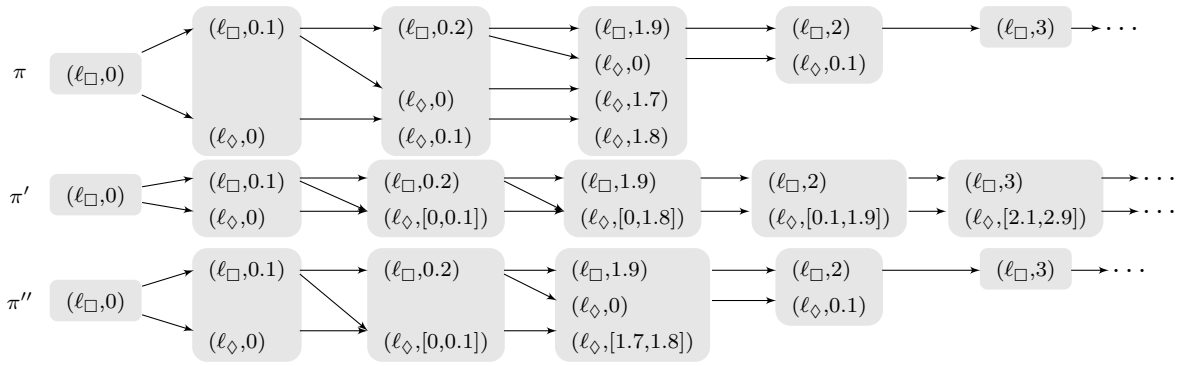

Figure 2: Several OCATA run prefixes

satisfy $(x \geq 1 \wedge x \leq 2)$. As $M_2 \subseteq M_1$, M_2 is the unique minimal model of $\delta(\ell_\diamond, b)$ wrt $[1.5, 2]$: $\mathsf{Succ}((\ell_\diamond, [1.5, 2]), b) = \{M_2\}$.

Approximation functions For an OCATA $\mathcal{A}$, an *approximation function* is a function $f : \mathrm{Config}(\mathcal{A}) \mapsto 2^{\mathrm{Config}(\mathcal{A})}$ s.t. for all configurations C, for all $C' \in f(C)$, for all locations $\ell \in L$: (i) (i) $\|C'(\ell)\| \leq \|C(\ell)\|$; (ii) for all $I \in C(\ell)$, there exists $J \in C'(\ell)$ s.t. $I \subseteq J$; and (iii) for all $J \in C'(\ell)$, there are $I_1, I_2 \in C(\ell)$ s.t. $\inf(J) = \inf(I_1)$ and $\sup(J) = \sup(I_2)$. We note $APP_\mathcal{A}$ the set of approximation functions for $\mathcal{A}$. We lift all approximation functions f to sets $\mathcal{C}$ of configurations in the usual way: $f(\mathcal{C}) = \cup_{C \in \mathcal{C}} f(C)$. In the rest of the paper we will rely mainly on approximation functions that enable to *bound* the number of clock copies in all configurations along all runs of an OCATA $\mathcal{A}$. Let $k \in \mathbb{N}$, we say that $f \in APP_\mathcal{A}$ is a *k-bounded approximation function* iff for all $C \in \mathrm{Config}(\mathcal{A})$, for all $C' \in f(C)$: $\|C'\| \leq k$.

f-Runs of OCATA We can now define formally the notion of *run* of an OCATA in the interval semantics. This notion will be parametrised by an approximation function f, that will be used to reduce the number of states present in all configurations along the run. Each new configuration in the run is thus obtained in three steps: letting time elapse, performing a discrete step, and applying the approximation function. Formally, let $\mathcal{A}$ be an OCATA of state space S, $f \in APP_\mathcal{A}$ be an approximation function and $\theta = (\sigma_1, \tau_1)(\sigma_2, \tau_2) \ldots (\sigma_i, \tau_i) \ldots$ be an infinite timed word. Let us note $t_i = \tau_i - \tau_{i-1}$ for all $i \geq 1$, assuming $\tau_0 = 0$. An *f-run of $\mathcal{A}$ on θ* is an infinite sequence $C_0, C_1, \ldots, C_i, \ldots$ of configurations s.t. $C_0 = \{(\ell_0, 0)\}$ and for all $i \geq 1$: $C_i \in f(\mathsf{Succ}(C_{i-1} + t_i, \sigma_i))$. Observe that for all pairs of configurations C, C' s.t. $C' \in f(\mathsf{Succ}(C + t, \sigma))$ for some f, t and σ, each $s \in C$ can be associated with a unique set $\mathsf{dest}(C, C', s) \subseteq C'$ containing all the 'successors' of s in C' and obtained as follows. Let $\overline{C} \in \mathsf{Succ}(C + t, \sigma)$ be s.t. $C' \in f(\overline{C})$. Thus, by definition, $\overline{C} = \cup_{\overline{s} \in C} M_{\overline{s}}$, where each $M_{\overline{s}} \in \mathsf{Succ}(\overline{s}, \sigma)$ is the minimal model that has been chosen for $\overline{s}$ when computing $\mathsf{Succ}(C, \sigma)$. Then, $\mathsf{dest}(C, C', s) = \{(\ell', J) \in C' \mid (\ell', I) \in M_s \text{ and } I \subseteq J\}$. Remark that $\mathsf{dest}(C, C', s)$ is well-defined because intervals are assumed to be disjoint in configurations. The function dest allows to define a DAG representation of runs, as it is usual with alternating automata. We regard a run $\pi = C_0, C_1, \ldots, C_i, \ldots$ as a rooted DAG $G_\pi = (V, \rightarrow)$, whose vertices V correspond to the states of the OCATA (vertices at depth i correspond to C_i), and whose set of edges $\rightarrow$ expresses the OCATA transitions. Formally, $V = \cup_{i \geq 0} V_i$, where for all $i \geq 0$: $V_i = \{(s, i) \mid s \in C_i\}$ is the set of all vertices of depth i. The root of G_π is $((\ell_0, 0), 0)$ and $(s_1, i_1) \rightarrow (s_2, i_2)$ iff $i_2 = i_1 + 1$ and $s_2 \in \mathsf{dest}(C_{i-1}, C_i, s_1)$. From now on, we will mainly use the DAG characterisation of f-runs.

Example: Fig. 2 displays three DAG representation of run prefixes of $\mathcal{A}_\varphi$ (Fig. 1), on the word $(a, 0.1)(a, 0.2)(a, 1.9)(b, 2)(b, 3) \ldots$ (grey boxes highlight the successive configurations). π only is an Id-run and shows that the number of clock copies cannot be bounded in general: if $\mathcal{A}_\varphi$ reads n a's between instants 0 and 1, n copies of the clock are created in location $\ell_\diamond$.

f-language of OCATA We can now define the accepted language of an OCATA, parametrised by an approximation function f. A *branch* of an f-run G is a (finite or) infinite path in G_π. We note $Bran^\omega(G)$ the set of all *infinite* branches of G_π and, for a branch β, we note $Infty(\beta)$ the set of locations occurring infinitely often along β. An f-run is *accepting* iff $\forall \beta \in Bran^\omega(G)$, $Infty(\beta) \cap F \neq \emptyset$ (i.e. we consider Büchi acceptance condition). We say that an infinite timed word θ is f-accepted by $\mathcal{A}$ iff there exists an accepting f-run of $\mathcal{A}$ on θ. We note $L^\omega_f(\mathcal{A})$ the language of all infinite timed words f-accepted by $\mathcal{A}$. We close the section by observing that a standard semantics for OCATA (where clock valuations are punctual values instead of intervals) is a particular case of the interval semantics, obtained by using the approximation function Id s.t. $Id(C) = \{C\}$ for all C. We denote by $L^\omega(\mathcal{A})$ the language $L^\omega_{Id}(\mathcal{A})$. Then, the following proposition shows the impact of approximation functions on the accepted language of the OCATA: they can only lead to *under-approximations* of $L^\omega(\mathcal{A})$.

Proposition 1. *For all OCATA $\mathcal{A}$, for all $f \in APP_\mathcal{A}$: $L^\omega_f(\mathcal{A}) \subseteq L^\omega(\mathcal{A})$.*

Idea. In Id-runs, all clock values are punctual, while in f-runs, clock values can be non-punctual intervals. Consider a set $(\ell, v_1), \ldots, (\ell, v_n)$ of states in location ℓ and with punctual values $v_1 \leq \ldots \leq v_n$, and consider its approximation $s = (\ell[v_1, v_n])$. Then, if a σ-labeled transition is firable from s, it is also firable from all (ℓ, v_i). The converse is not true: there might be a set of σ-labeled transitions that are firable from each (ℓ, v_i), but no σ-labeled transition firable from s, because *all clock values* in I must satisfy the transition guard. □

4 TOCATA: an easily complementable class of OCATA

In this section, we introduce the class of *tree-like* OCATA (TOCATA for short) whose acceptance condition can be made simpler than in the general case. This specific property enables to easily complement the TOCATA.

Tree-like OCATA An OCATA $\mathcal{A} = (\Sigma, L, \ell_0, F, \delta)$ is a TOCATA iff there exists a partition $L_1, L_2, \ldots, L_m$ of L and a partial order $\preccurlyeq$ on the sets $L_1, L_2, \ldots, L_m$ s.t.: (i) each L_i contains either only accepting states or no accepting states: $\forall 1 \leq i \leq m$ either $L_i \subseteq F$ or $L_i \cap F = \emptyset$; and (ii) the partial order $\preccurlyeq$ is compatible with the transition relation and yields the 'tree-like' structure of the automaton in the following sense: $\preccurlyeq$ is s.t. $L_j \preccurlyeq L_i$ iff $\exists \sigma \in \Sigma$, $\ell \in L_i$ and $\ell' \in L_j$ such that ℓ' is present in $\delta(\ell, \sigma)$.

Properties of TOCATA The first peculiar property of TOCATA is concerned with the acceptance condition. In the general case, a run of an OCATA is accepting iff all its branches visit accepting states infinitely often. Thanks to the partition of locations characterising TOCATA, this condition can be made simpler: a run is now accepting iff each branch eventually visits accepting states *only*, because it reaches a partition of the locations that are all accepting.

Proposition 2. *An Id-run G_π of a TOCATA $\mathcal{A}$ with set of accepting locations F is* accepting *iff $\forall \beta = \beta_0\beta_1 \ldots \beta_i \ldots \in Bran^\omega(G_\pi)$, $\exists n_\beta \in \mathbb{N}$ s.t. $\forall i > n_\beta$: $\beta_i = ((\ell, v), i)$ implies $\ell \in F$.*

The second property of interest is that TOCATA can be easily complemented swaping accepting and non-accepting locations, and 'dualising' the transition relation (as in the case of OCATA on finite words [14]). Formally, the dual of a formula $\varphi \in \Gamma(L)$ is the formula $\overline{\varphi}$ defined inductively as follows. $\forall \ell \in L$, $\overline{\ell} = \ell$; $\overline{false} = true$ and $\overline{true} = false$; $\overline{\varphi_1 \vee \varphi_2} = \overline{\varphi_1} \wedge \overline{\varphi_2}$; $\overline{\varphi_1 \wedge \varphi_2} = \overline{\varphi_1} \vee \overline{\varphi_2}$; $\overline{x.\varphi} = x.\overline{\varphi}$; the dual of a clock constraint is its negation (for example: $\overline{x \leq c} = x > c$). Then, for all TOCATA $\mathcal{A} = (\Sigma, L, \ell_0, F, \delta)$, we let $\mathcal{A}^C = (\Sigma, L, \ell_0, L \setminus F, \overline{\delta})$ where $\overline{\delta}(\ell, \sigma) = \overline{\delta(\ell, \sigma)}$. Thanks to Proposition 2, we can prove that $\mathcal{A}^C$ accepts the complement of $\mathcal{A}$'s language:

Proposition 3. *For all TOCATA* $\mathcal{A}$, $L^\omega(\mathcal{A}^C) = T\Sigma^\omega \setminus L^\omega(\mathcal{A})$.

5 Application: MITL model-checking

We present here our technique to perform MITL model-checking. To prove our approach is correct, we mainly rely on the interval semantics previously defined and the properties of TOCATA.

Metric Interval Temporal Logic. Given a finite alphabet Σ, the formulas of MITL are defined by the following grammar, where $\sigma \in \Sigma$ and $I \in \mathcal{I}(\mathbb{N}^{+\infty})$ is non-singular:

$$\varphi := \top \mid \sigma \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \varphi_1 U_I \varphi_2.$$

We adopt the following shortcuts: $\Diamond_I \varphi$ ("eventually φ in I") stands for $\top U_I \varphi$, $\Box_I \varphi$ ("always φ on I") for $\neg\Diamond_I\neg\varphi$. Given an infinite timed word $\theta = (\bar{\sigma}, \bar{\tau})$ over Σ, a position $i \in \mathbb{N}_0$ and an MITL formula φ, we write $(\theta, i) \models \varphi$ when θ *satisfies* φ from position i. This satisfaction relation is defined in the usual way, let us only recall the semantics of the until operator 'U': $(\theta, i) \models \varphi_1 U_I \varphi_2$ iff $\exists j \geq i$: $(\theta, i) \models \varphi_2$, $\tau_j - \tau_i \in I \wedge \forall i \leq k < j$: $(\theta, k) \models \varphi_1$. We say that θ *satisfies* φ, written $\theta \models \varphi$, iff $(\theta, 1) \models \varphi$. We note $[\![\varphi]\!]$ the timed language $\{\theta \mid \theta \models \varphi\}$.

When applying, to an MITL formula φ, the construction defined by Ouaknine and Worrell [14] in the setting of MTL interpreted on *finite words*, one obtains a TOCATA $\mathcal{A}_\varphi$ that accepts the *infinite words* language of ϕ (we rely on the specific properties of TOCATA given by Propositions 2 and 3 to prove this result). For example, the OCATA $\mathcal{A}_\varphi$ in Fig. 1 accepts exactly $[\![\Box(a \Rightarrow \Diamond_{[1,2]} b)]\!]$. Moreover, for every MITL formula φ, there exists an $M(\varphi)$-bounded approximation function $f^\star_\varphi$ s.t. $L^\omega_{f^\star_\varphi}(\mathcal{A}_\varphi) = [\![\varphi]\!]$.

To perform MITL model checking on a system BTA $\mathcal{B}$ and a formula φ, use would like to have a BTA $\mathcal{B}_{\neg\varphi}$ recognizing $[\![\neg\varphi]\!]$ so that we could verify that $L(\mathcal{B}) \subseteq [\![\varphi]\!]$ verifying that $L(\mathcal{B}) \times L(\neg\varphi) = \emptyset$. In general, it is not possible to turn OCATA into BTA [1] because of the unbounded number of clock copies they use (see Example 1). But thanks to the bound $M(\neg\varphi)$ on the number of clock copies and the approximation function $f^\star_{\neg\varphi}$ it is now easy to build a BTA $\mathcal{B}_{\neg\varphi}$ accepting $[\![\neg\varphi]\!]$ for every MITL formula φ, using a subset construction *à la Miyano Hayashi* [12]. Equipped with these theoretical results, we elaborated an algorithm to solve the *model-checking* problem of MITL. We define region-based and zone-based [2] versions of this algorithm. They work *on-the-fly* in the sense that they work directly on the structure of the OCATA $\mathcal{A}_{\neg\varphi}$ (whose size is linear in the size of φ), avoiding building $\mathcal{B}_{\neg\varphi}$ beforehand (which is, in the worst case, exponential in the size of φ). We developed a prototype tool implementing these algorithms.

6 Conclusion

In this paper, we considered the class of timed languages defined by MITL formulas. We proved that the class of OCATA used by Ouaknine and Worrell [13] to represent these languages form a first subclass of OCATA for which there exists a family of bounded approximation functions $f^\star_\varphi$, s.t., for every MITL formula φ, $L^\omega_{f^\star_\varphi}(\mathcal{A}_\varphi) = L^\omega(\mathcal{A}_\varphi)$. We think that an interesting future work would be to characterize the class of OCATA's $\mathcal{A}$ for which there exist a bounded approximation function f such that $L^\omega_f(\mathcal{A}) = L^\omega(\mathcal{A})$. In another context, we hope the interval semantics could help us to perform MITL synthesis on finite words.

References

[1] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[2] P. A. Abdulla, J. Deneux, J. Ouaknine, K. Quaas and J. Worrell. Universality Analysis for One-Clock Timed Automata. *Fundam. Inform.*, 89(4):419–450, 2008.

[3] R. Alur, T. Feder and T. A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.

[4] T. Brihaye, M. Estiévenart and G. Geeraerts. On MITL and Alternating Timed Automata. In *FORMATS*, volume 8053 of *LNCS*. Springer, 2013.

[5] T. Brihaye, M. Estiévenart and G. Geeraerts. On MITL and Alternating Timed Automata over infinite words. Technical report, `http://math.umons.ac.be/maef/full.pdf`, 2014.

[6] E. M. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT Press, 2001.

[7] O. Finkel. Undecidable problems about timed automata. In *FORMATS*, volume 4202 of *LNCS*. Springer, 2006.

[8] P. Gastin and D. Oddoux. Fast LTL to Büchi Automata Translation. In *CAV*, volume 2102 of *LNCS*. Springer, 2001.

[9] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[10] O. Kupferman and M. Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. Comput. Log.*, 2(3):408–429, 2001.

[11] S. Lasota and I. Walukiewicz. Alternating timed automata. *ACM Trans. Comput. Log.*, 9(2), 2008.

[12] S. Miyano and T. Hayashi. Alternating Finite Automata on omega-Words. *Theor. Comput. Sci.*, 32:321–330, 1984.

[13] J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *LICS'05*. IEEE, 188-197, 2005.

[14] J. Ouaknine and J. Worrell. On the decidability and complexity of metric temporal logic over finite words. *Logical Methods in Computer Science*, 3(1), 2007.

[15] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *LICS'86*. IEEE, 1986.