MightyL: A Compositional Translation from MITL to Timed Automata[⋆]

Thomas Brihaye[1], Gilles Geeraerts[2], Hsi-Ming Ho[1], Benjamin Monmege[3]

[1] Université de Mons, Belgium, thomas.brihaye,hsi-ming.ho@umons.ac.be
[2] Université libre de Bruxelles, Belgium, gigeerae@ulb.ac.be
[3] Aix Marseille Univ, CNRS, LIF, France, benjamin.monmege@lif.univ-mrs.fr

Abstract. Metric Interval Temporal Logic (MITL) was first proposed in the early 1990s as a specification formalism for real-time systems. Apart from its appealing intuitive syntax, there are also theoretical evidences that make MITL a prime real-time counterpart of Linear Temporal Logic (LTL). Unfortunately, the tool support for MITL verification is still lacking to this day. In this paper, we propose a new construction from MITL to timed automata via very-weak one-clock alternating timed automata. Our construction subsumes the well-known construction from LTL to Büchi automata by Gastin and Oddoux and yet has the additional benefits of being compositional and integrating easily with existing tools. We implement the construction in our new tool MightyL and report on experiments using Uppaal and LTSmin as back-ends.

1 Introduction

The design of critical software that respect real-time specifications is a notoriously difficult problem. In this context, verification of programs against formal specifications is crucial, in order to handle the thin timing behaviours. In the untimed setting, a logic widely used both in academia and industry is *Linear Temporal Logic* (LTL) [31]. A crucial ingredient of its success is the possibility to translate LTL formulae into (Büchi) automata. In the real-time counterpart, *Metric Interval Temporal Logic* (MITL) [3] has been introduced twenty years ago where it was established that it can be translated into (Büchi) *timed automata* (TA). Beyond verification of real-time software, there are numerous interests in MITL from other domains, e.g. automated planning and scheduling [36], control engineering [18] and systems biology [6]. The translation from MITL to TAs is complicated and has led to some simplified constructions, e.g. [17, 27]. However, despite these efforts, the tool support for MITL is still lacking to this day. To the best of our knowledge, the only implementation of an automata-based construction is described in [10, 11], but is not publicly available. Since existing verification tools based on timed automata have been around for quite some time

⋆ This work has been supported by the FRS/F.N.R.S. PDR grant SyVeRLo, and (partially) funded by the DeLTA project (ANR-16-CE40-0007) and the SensAS project (INS2I JCJC'17).

and have been successful (e.g., UPPAAL [26] first appeared in 1995), it would be preferable if such translation can be used with these tools.

In the present paper, we attempt to amend the situation by proposing a more practical construction from MITL to (Büchi) timed automata. Compared to [10, 11], our construction has the following advantages:

1. While we also use *one-clock alternating timed automata* (OCATA) [29] as an intermediate formalism, our construction exploits the 'very-weakness' of the structure of OCATAs obtained from MITL formulae to reduce state space. In particular, our construction subsumes LTL2BA [19] in the case of LTL.
2. The number of clocks in the resulting TA is reduced by a factor of up to two. This is achieved via a more fine-grained analysis of the possible clock values (see Section 5).
3. The construction is *compositional*: for each location of the OCATA $\mathcal{A}$ obtained from the input MITL formula, we construct a 'component' TA and establish a connection between the runs of $\mathcal{A}$ and the runs of the synchronous product of these components. Thanks to this connection, we can give the output TA in terms of components; this greatly simplifies the implementation, and speeds up its execution.
4. The construction is compatible with off-the-shelf model-checkers: our tool MIGHTYL generates output automata in the UPPAAL XML format which, besides UPPAAL [26] itself, can also be analysed by LTSMIN [23] with OPAAL front-end, TIAMO [9], ITS-TOOLS [33], DIVINE [5], etc.

Related work. There is already a number of MITL-to-TA constructions in the literature [3, 17, 27]. However, most of them interpret MITL over *signals* (i.e. the *continuous* semantics of MITL) and hence generate *signal automata*. This choice unfortunately hinders the possibility to leverage existing tools based on classical timed automata over *timed words* [2] and is probably one of the reasons why the aforementioned constructions have never been implemented.[4] We, following [4, 10, 11, 35] (among others), interpret MITL over timed words (i.e. the *pointwise* semantics of MITL). Note that there have been some implementations that deal with peculiar specification patterns over timed words (e.g. [1]). For MITL, apart from [10, 11] that we mentioned earlier, we are only aware of implementations for rather restricted cases, such as the safety fragment of $\mathsf{MITL}_{0,\infty}$ [12] or MITL over untimed words [36]. Our construction subsumes all of these approaches.

Using alternating automata as an intermediate formalism is a standard approach in LTL model-checking [34]. However, the translation from alternating automata to Büchi automata may incur an exponential blow-up if the output automaton is constructed explicitly [19]. For this reason, an *on-the-fly* approach is proposed in [21], but it requires a specialised model-checking algorithm. Alternatively, [8] gives a *symbolic* encoding of alternating automata which can be used directly with NUSMV [13], but minimality of transitions (which may

[4] Nonetheless, it has been argued that a continuous model of time is preferable from a theoretical point of view; see e.g. [22].

potentially improve the performance of verification algorithms, cf. [21]) is difficult to enforce in this setting (see also [14, 32]). Our construction combines the advantages of these approaches—it can be regarded as a symbolic encoding of OCATAs in TAs, enforcing some minimality criteria on transitions for efficiency (see Section 6)—and provides compatibility with existing tools that construct state spaces on-the-fly. By contrast, [17, 27], not based on OCATAs, also give the resulting automaton in terms of smaller component automata, but they have to use specialised product constructions to synchronise the components.

Apart from automata-theoretic approaches, [7] considers 'bounded model-checking' which encodes the satisfiability problem for MITL (in the continuous semantics) into an SMT problem (*Satisfiability Modulo Theories*) [15]. This approach is complete when very large bounds (numbers of regions of equivalent TA) are used, but such bounds are clearly impractical for current SMT solvers.

Outline. Section 2 starts with preliminary definitions of timed logics and (alternating) timed automata. Sections 3, 4 and 5 then give our new translation from formulae to generalised Büchi (timed) automata for LTL, $\mathsf{MITL}_{0,\infty}$ (a fragment of MITL where only intervals of the form $[0, a]$, $[0, a)$, $[a, +\infty)$, or $(a, +\infty)$ are allowed), and full MITL, respectively. We report on our OCaml implementation MightyL and some promising experiments on several benchmarks in Section 6.

2 Timed logics vs (alternating) timed automata

Timed languages. Let AP be a finite set of atomic propositions, and $\Sigma = 2^{\mathsf{AP}}$. A timed word over Σ is an infinite sequence $\rho = (\sigma_1, \tau_1)(\sigma_2, \tau_2)\ldots$ over $\Sigma \times \mathbb{R}^+$ with $(\tau_i)_{i\geq 1}$ a non-decreasing sequence of non-negative real numbers. We denote by $T\Sigma^\omega$ the set of timed words over Σ. A *timed language* is a subset of $T\Sigma^\omega$.

Timed logics. We consider the satisfiability and model-checking problem of Metric Interval Temporal Logic (MITL), an extension of Linear Temporal Logic (LTL) in which temporal modalities can be labelled with *non-singular* timed intervals (or $[0, 0]$, which is the only singular interval we allow). Formally, MITL formulae over AP are generated by the grammar

$$\varphi := p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathbf{X}_I \varphi \mid \varphi\, \mathbf{U}_I\, \varphi$$

where $p \in \mathsf{AP}$ and I is either a non-singular interval over $\mathbb{R}^+$ with endpoints in $\mathbb{N} \cup \{+\infty\}$ or $[0, 0]$. To simplify our explanations, we will only consider closed non-singular intervals in the sequel, i.e. intervals of the form $[a, b]$ or $[a, +\infty)$, with $0 \leq a < b < +\infty$. We let $|I|$ be the length of the interval I: $|[a, b]| = b - a$ for $0 \leq a < b < +\infty$ and $|[a, +\infty)| = +\infty$.

We consider the *pointwise semantics* and interpret MITL formulae over timed words. The semantics of a formula φ in MITL is defined inductively: given $\rho = (\sigma_1, \tau_1)(\sigma_2, \tau_2) \cdots \in T\Sigma^\omega$, and a position $i \geq 1$, we let

- $(\rho, i) \models p$ if $p \in \sigma_i$;
- $(\rho, i) \models \varphi_1 \wedge \varphi_2$ if $(\rho, i) \models \varphi_1$ and $(\rho, i) \models \varphi_2$;

– $(\rho, i) \models \neg\varphi$ if $(\rho, i) \not\models \varphi$;
– $(\rho, i) \models \mathbf{X}_I\varphi$ if $(\rho, i+1) \models \varphi$ and $\tau_{i+1} - \tau_i \in I$;
– $(\rho, i) \models \varphi_1 \, \mathbf{U}_I \, \varphi_2$ if there exists $j \geq i$, $(\rho, j) \models \varphi_2$, $\tau_j - \tau_i \in I$, and, for all $i \leq k < j$, $(\rho, k) \models \varphi_1$.

use '≡' over ':=' here?

We derive other Boolean operators with the following macros: $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\top \equiv p \vee \neg p$, $\bot \equiv \neg\top$, and $\varphi_1 \Rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$. We also define other temporal operators as usual: the 'eventually' operator $\mathbf{F}_I\varphi \equiv \top \, \mathbf{U}_I \, \varphi$, the 'globally' operator $\mathbf{G}_I\varphi \equiv \neg\mathbf{F}_I\neg\varphi$, the 'release' operator $\varphi_1 \mathbf{R}_I \varphi_2 \equiv \neg((\neg\varphi_1)\,\mathbf{U}_I\,(\neg\varphi_2))$, and the 'dual-next' operator $\overline{\mathbf{X}}_I\varphi \equiv \neg\mathbf{X}_I\neg\varphi$ (contrary to LTL, it is not true that $\neg\mathbf{X}_I\varphi \equiv \mathbf{X}_I\neg\varphi$). With the release and dual-next operators, we can transform every formula φ into *negative normal form*, i.e. formulae using only predicates of AP, their negations, and the operators $\vee$, $\wedge$, $\mathbf{U}_I$, $\mathbf{R}_I$, $\mathbf{X}_I$, and $\overline{\mathbf{X}}_I$. To help the understanding, let us detail the semantics of $\varphi_1 \, \mathbf{R}_I \, \varphi_2$:

– $(\rho, i) \models \varphi_1 \, \mathbf{R}_I \, \varphi_2$ if for all $j \geq i$ such that $\tau_j - \tau_i \in I$, either $(\rho, j) \models \varphi_2$, or there exists $i \leq k < j$ such that $(\rho, k) \models \varphi_1$.

We say that ρ satisfies the formula φ, written $\rho \models \varphi$ if $(\rho, 1) \models \varphi$, and we denote by $[\![\varphi]\!]$ the set of all timed words satisfying φ. When writing formulae, we omit the trivial interval $[0, +\infty)$. LTL is the fragment of MITL where all operators are labelled by $[0, \infty)$; and $\mathsf{MITL}_{0,\infty}$ is the fragment where, in all intervals, either the left endpoint is 0 or the right endpoint is $+\infty$.

Timed automata. Let X be a finite set of real valued variables, called clocks. The set $\mathcal{G}(X)$ of *clock constraints* g over X is defined by $g := \top \mid g \wedge g \mid x \bowtie c$, where $\bowtie \in \{\leq, <, \geq, >\}$, $x \in X$ and $c \in \mathbb{N}$. A *valuation* over X is a mapping $v \colon X \to \mathbb{R}^+$. We denote by $\mathbf{0}$ the valuation that maps every clock to 0, and we write the valuation simply as a value in $\mathbb{R}^+$ when X is a singleton. The satisfaction of a constraint g by a valuation v is defined in the usual way and noted $v \models g$, and we denote by $[\![g]\!]$ the set of valuations v satisfying g. For $t \in \mathbb{R}^+$, we let $v + t$ be the valuation defined by $(v + t)(x) = v(x) + t$ for all $x \in X$. For $R \subseteq X$, we let $v[R \leftarrow 0]$ be the valuation defined by $(v[R \leftarrow 0])(x) = 0$ if $x \in R$, and $(v[R \leftarrow 0])(x) = v(x)$ otherwise.

We introduce the notion of *generalised Büchi timed automaton* (GBTA) as an extension of classical timed automata [2] with a generalised accepting condition (used by [20] in the untimed setting). A GBTA is a tuple $\mathcal{A} = (L, \Sigma, \ell_0, \Delta, \mathcal{F})$ where L is a finite set of locations, Σ is a finite alphabet, $\ell_0 \in L$ is the initial location, $\Delta \subseteq L \times \Sigma \times \mathcal{G}(X) \times 2^X \times L$ is the transition relation, and $\mathcal{F} = \{F_1, \ldots, F_n\}$, with $F_i \subseteq L$ for all $1 \leq i \leq n$, is the set of sets of final locations. A timed automaton (TA), as described in [2], is a special case of GBTA where $\mathcal{F} = \{F\}$ is a singleton (F contains the accepting locations of the TA). A *state* of $\mathcal{A}$ is a pair (ℓ, v) of a location $\ell \in L$ and a valuation v of the clocks in X. A *run* of $\mathcal{A}$ over the timed word $(\sigma_1, \tau_1)(\sigma_2, \tau_2) \cdots \in T\Sigma^\omega$ is a sequence of states $C_0, C_1, \ldots$ where (i) $C_0 = (\ell_0, \mathbf{0})$ and (ii) for each $i \geq 0$ such that $C_i = (\ell, v)$, there is a transition $(\ell, \sigma_{i+1}, g, R, \ell')$ such that $C_{i+1} = (\ell', v')$, $v + (\tau_{i+1} - \tau_i) \models g$ (assuming $\tau_0 = 0$) and $v' = (v + (\tau_{i+1} - \tau_i))[R \leftarrow 0]$. By the generalised Büchi acceptance condition, a run is *accepting* iff the set of locations that it visits infinitely often contains at least one location from each set F_i, for all $1 \leq i \leq n$. We let $[\![\mathcal{A}]\!]$ be the set of timed words on which there exist accepting runs of $\mathcal{A}$.

Synchronisation of timed automata. In the following, we will consider GBTAs described by synchronous products of several components. More precisely, given two GBTAs $\mathcal{A}^1 = (L^1, \Sigma, \ell_0^1, \Delta^1, \mathcal{F}^1)$ and $\mathcal{A}^2 = (L^2, \Sigma, \ell_0^2, \Delta^2, \mathcal{F}^2)$ over disjoint sets of clocks, we define the GBTA $\mathcal{A}^1 \times \mathcal{A}^2 = (L, \Sigma, \ell_0, \Delta, \mathcal{F})$ obtained by synchronising $\mathcal{A}^1$ and $\mathcal{A}^2$. $\mathcal{A}^1 \times \mathcal{A}^2$'s set of locations is $L = L^1 \times L^2$, with $\ell_0 = (\ell_0^1, \ell_0^2)$. The acceptance condition is obtained by mimicking a disjoint union of the generalised Büchi conditions: assuming $\mathcal{F}^1 = \{F_1, \dots, F_n\}$ and $\mathcal{F}^2 = \{G_1, \dots, G_m\}$, we let $\mathcal{F} = \{F_1 \times L^2, \dots, F_n \times L^2, L^1 \times G_1, \dots, L^1 \times G_m\}$. Finally, $((\ell_1^1, \ell_1^2), \sigma, g, R, (\ell_2^1, \ell_2^2)) \in \Delta$ if there exists $(\ell_1^1, \sigma, g^1, R^1, \ell_2^1) \in \Delta^1$ and $(\ell_1^2, \sigma, g^2, R^2, \ell_2^2) \in \Delta^2$ such that $g = g^1 \wedge g^2$ and $R = R^1 \cup R^2$. This definition can be extended for the synchronisation of a set of GBTAs $\{\mathcal{A}^i \mid i \in I\}$: the product is then written as $\prod_{i \in I} \mathcal{A}^i$.

One-clock alternating timed automata. One-clock alternating timed automata (OCATA) [29] extend (non-deterministic) one-clock timed automata by adding *conjunctive transitions*. Intuitively, a conjunctive transition spawns several copies of the automaton that run in parallel from the targets of the transition. A word is accepted if and only if *all* copies accept it. An example is shown in Fig. 1, where the conjunctive transition is the hyperedge starting from ℓ_0.

Formally, we consider a single clock x and, for a set L of locations, let $\Gamma(L)$ be the set of formulae defined by

$$\gamma := \top \mid \bot \mid \gamma \vee \gamma \mid \gamma \wedge \gamma \mid \ell \mid x \bowtie c \mid x.\gamma$$

where $c \in \mathbb{N}$, $\bowtie \in \{\leq, <, \geq, >\}$, and $\ell \in L$. Compared to the clock constraints defined above for TAs, $\Gamma(L)$ allows non-determinism ($\vee$ operator), locations as atoms, and expressions of the form $x.\gamma$ (meaning that x is reset in γ). An OCATA is a tuple $\mathcal{A} = (L, \Sigma, \ell_0, \delta, F)$ where L is a finite set of locations, Σ is a finite alphabet, $\ell_0 \in L$ is the initial location, $\delta \colon L \times \Sigma \to \Gamma(L)$ is the transition function, and $F \subseteq L$ is the set of final locations. A *state* of $\mathcal{A}$ is a pair (ℓ, v) of a location in L and a valuation of the single clock x. Models of the formulae in $\Gamma(L)$, with respect to a clock valuation $v \in \mathbb{R}^+$, are sets of states M:

- $M \models_v \top$; $M \models_v \ell$ if $(\ell, v) \in M$; $M \models_v x \bowtie c$ if $v \bowtie c$; $M \models_v x.\gamma$ if $M \models_0 \gamma$;
- $M \models_v \gamma_1 \wedge \gamma_2$ if $M \models_v \gamma_1$ and $M \models_v \gamma_2$;
- $M \models_v \gamma_1 \vee \gamma_2$ if $M \models_v \gamma_1$ or $M \models_v \gamma_2$.

A set M of states is said to be a *minimal model* of the formula $\gamma \in \Gamma(S)$ with respect to a clock valuation $v \in \mathbb{R}^+$ iff $M \models_v \gamma$ and there is no proper subset $M' \subset M$ with $M' \models_v \gamma$. A run of $\mathcal{A}$ over a timed word $\rho = (\sigma_1, \tau_1)(\sigma_2, \tau_2) \cdots \in T\Sigma^\omega$ is a rooted directed acyclic graph (DAG) $G = (V, \to)$ with vertices of the form $(\ell, v, i) \in L \times \mathbb{R}^+ \times \mathbb{N}$, $(\ell_0, 0, 0)$ as root, and edges as follows: for every vertex (ℓ, v, i), we choose a minimal model M of the formula $\delta(\ell, \sigma_{i+1})$ with respect to $v + (\tau_{i+1} - \tau_i)$ (again, $\tau_0 = 0$), and we have an edge $(\ell, v, i) \to (\ell', v', i+1)$ in G

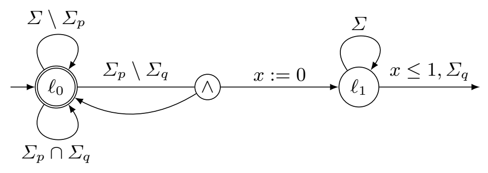

Fig. 1. An OCATA accepting the language of $\mathbf{G}(p \Rightarrow \mathbf{F}_{[0,1]} q)$.

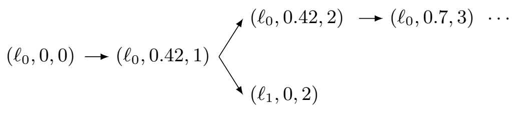

Fig. 2. A run of the OCATA of Fig. 1 over $(\emptyset, 0.42)(\{p\}, 0.42)(\{q\}, 0.7)\ldots$.

for every state (ℓ', v') appearing in model M. Such a run is *accepting* iff there is no infinite path in G that visit final locations only finitely often. We let $[\![\mathcal{A}]\!]$ be the set of timed words on which there exist accepting runs of $\mathcal{A}$.

It is also useful to see a run as a linear sequence of *configurations* (i.e. finite sets of states) which gather all states at a given DAG level. Formally, from a DAG $G = (V, \rightarrow)$ we extract the sequence of configurations $K_0, K_1, \ldots$ where $K_i = \{(\ell, v) \mid (\ell, v, i) \in V\}$ for all $i \geq 0$.[5]

Example 1. Consider the OCATA of Fig. 1 on the alphabet $\Sigma = 2^{\{p,q\}}$. For each proposition $\pi \in \{p, q\}$, we write $\Sigma_\pi = \{\sigma \in \Sigma \mid \pi \in \sigma\}$. A run over the timed word $(\emptyset, 0.42)(\{p\}, 0.42)(\{q\}, 0.7)\ldots$ is depicted in Fig. **??**. It starts with the DAG rooted in $(\ell_0, 0, 0)$ (initially, there is only one copy in ℓ_0 with the clock equal to 0). This root has a single successor $(\ell_0, 0.42, 1)$, which has two successors $(\ell_0, 0.42, 2)$ and $(\ell_1, 0, 2)$ (after firing the conjunctive transition from ℓ_0). Then, $(\ell_1, 0, 2)$ has no successor since the empty model is a minimal model of the next transition (the transition from ℓ_1 points to no location). The associated sequence of configurations starts by: $\{(\ell_0, 0)\}, \{(\ell_0, 0.42)\}, \{(\ell_0, 0.42), (\ell_1, 0)\}\ldots$

Each formula φ of MITL can be translated into an OCATA $\mathcal{A}_\varphi$ that accepts the same language [11,29], and with a number of locations linear in the number of subformulae of φ. We recall the definition of $\mathcal{A}_\varphi$ for the sake of completeness. The set of locations of $\mathcal{A}_\varphi$ contains: (i) a copy φ_{init} of φ; (ii) all the subformulae of φ (that we suppose to be in negative normal form) whose outermost connective is $\mathbf{U}_I$ or $\mathbf{R}_I$; and (iii) copies ψ and ψ^r of all subformulae ψ of φ whose outermost connective is $\mathbf{X}_I$ or $\overline{\mathbf{X}}_I$. Its initial location is φ_{init}, and the accepting locations of F are all the subformulae of the form $\varphi_1 \, \mathbf{R}_I \, \varphi_2$. Finally, δ is defined inductively:

- $\delta(\varphi_{init}, \sigma) = x.\delta(\varphi, \sigma)$, $\delta(\top, \sigma) = \top$, and $\delta(\bot, \sigma) = \bot$;
- $\delta(p, \sigma) = \top$ if $p \in \sigma$, $\delta(p, \sigma) = \bot$ otherwise;

[5] In the current (infinite-word) setting, we cannot define acceptance conditions in terms of configurations as in [29].

- $\delta(\neg p, \sigma) = \top$ if $p \notin \sigma$, $\delta(\neg p, \sigma) = \bot$ otherwise;
- $\delta(\varphi_1 \vee \varphi_2, \sigma) = \delta(\varphi_1, \sigma) \vee \delta(\varphi_2, \sigma)$, and $\delta(\varphi_1 \wedge \varphi_2, \sigma) = \delta(\varphi_1, \sigma) \wedge \delta(\varphi_2, \sigma)$;
- $\delta(\varphi_1 \mathbf{U}_I \varphi_2, \sigma) = (x.\delta(\varphi_2, \sigma) \wedge x \in I) \vee (x.\delta(\varphi_1, \sigma) \wedge \varphi_1 \mathbf{U}_I \varphi_2 \wedge x \leq \sup I)$;
- $\delta(\varphi_1 \mathbf{R}_I \varphi_2, \sigma) = (x.\delta(\varphi_2, \sigma) \vee x \notin I) \wedge (x.\delta(\varphi_1, \sigma) \vee \varphi_1 \mathbf{R}_I \varphi_2 \vee x > \sup I)$;
- $\delta(\mathbf{X}_I \varphi, \sigma) = x.(\mathbf{X}_I \varphi)^r$, and $\delta((\mathbf{X}_I \varphi)^r, \sigma) = x \in I \wedge x.\delta(\varphi, \sigma)$;
- $\delta(\overline{\mathbf{X}}_I \varphi, \sigma) = x.(\overline{\mathbf{X}}_I \varphi)^r$, and $\delta((\overline{\mathbf{X}}_I \varphi)^r, \sigma) = x \notin I \vee x.\delta(\varphi, \sigma)$.

As already noticed in [11], the OCATA $\mathcal{A}_\varphi$ produced from an MITL formula φ is *very-weak* [19, 25, 28], i.e. it comes with a partial order on its locations such that all locations appearing in $\delta(\ell, \sigma)$ are bounded above by ℓ in this order. For an OCATA $\mathcal{A}_\varphi$ obtained from an MITL formula φ, the order is given by the subformula order: φ_{init} is the greatest element in the order, and a location ψ is less than χ if ψ is a subformula of χ. We will also make use of the following properties of δ: (i) if ℓ' appears in $\delta(\ell, \sigma)$ then it is preceded by a clock reset if and only if $\ell' \neq \ell$; and (ii) each ℓ' either has no ancestors or has a unique ancestor, i.e. there is a unique ℓ such that ℓ' appears in $\delta(\ell, \sigma)$ for some σ.

Theorem 2 ([11]). *For all formulae φ of MITL, $[\![\mathcal{A}_\varphi]\!] = [\![\varphi]\!]$.*

Remark 3. To ease the presentation, we use Boolean formulae over atomic propositions as transition labels. For instance, $\Sigma \setminus \Sigma_p$ will be written as $\neg p$.

3 Compositional removal of alternation

The current and next two sections are devoted to explaining the core idea of our construction: simulate the OCATA $\mathcal{A}_\varphi$ obtained from an MITL formula φ by the synchronous product of component Büchi timed automata, one for each temporal subformula. The very-weakness of $\mathcal{A}_\varphi$ is crucial for our construction to work: a run of $\mathcal{A}_\varphi$ is accepting iff $\mathcal{A}_\varphi$ does not get stuck at a non-accepting location in any branch. Therefore, we can keep track of each location with a separate component and simply define a suitable Büchi acceptance condition on each such component.[6] The main advantage of our compositional construction is that it preserves the structure of the formula, and thus we can hope that the model-checking tool (which will in the end be responsible for the composition) takes this into account.[7] At the very least, the model-checking tool can use an on-the-fly approach in composition (as is indeed the case for UPPAAL and LTSMIN), which is often faster in practice: the explicit construction of the whole product can be avoided when there is an accepting run. In what follows, we fix a formula $\varphi \in$ MITL over AP in negative normal form, and let Φ be the set of subformulae ψ of φ whose outermost connective is $\mathbf{U}_I$ or $\mathbf{R}_I$ (for the sake

some additional text here!

[6] This is not possible for general (not very-weak) OCATAs since it might be the case that a branch alternates between several non-accepting location without ever hitting an accepting location.

[7] The same idea underlies the antichain-based algorithms for LTL model-checking [**?**], where the structure can be exploited to define a pre-order on the state space of the resulting automaton.

of simplicity, we forget the operators $\mathbf{X}_I$ and $\overline{\mathbf{X}}_I$ hereafter). Then, we add a fresh atomic proposition p_ψ for each subformula $\psi \in \Phi$ (i.e. for each non-initial location of the OCATA $\mathcal{A}_\varphi$). Let AP_φ denote the set of fresh atomic propositions hence introduced. For each subformula ψ of φ, we denote by $\mathcal{P}_\psi$ the set of atomic propositions $p_\xi \in \mathsf{AP}_\varphi$ such that ξ is a top-level temporal subformula of ψ, i.e. the outermost connective of ξ is $\mathbf{U}_I$ or $\mathbf{R}_I$, yet ξ does not occur under the scope of another $\mathbf{U}_I$ or $\mathbf{R}_I$. For instance, $\mathcal{P}_{p\mathbf{U}_I q \vee r\mathbf{U}_I(s\mathbf{R}t)} = \{p_{p\mathbf{U}_I q}, p_{r\mathbf{U}_I(s\mathbf{R}t)}\}$.

Hintikka sequences and triggers. A *Hintikka sequence* of φ is a timed word ρ' over $\mathsf{AP} \cup \mathsf{AP}_\varphi$. Intuitively, Hintikka sequences can be regarded as an instrumented version of timed words, where the extra atomic propositions from AP_φ are *triggers* that connect timed words to their runs in the OCATA $\mathcal{A}_\varphi$; this is the central notion of our construction which, as we will prove, indeed simulates the runs of $\mathcal{A}_\varphi$. Pulling the trigger p_ψ (i.e. setting p_ψ to true) at some point means that ψ is required to hold at this point. However, the absence of a trigger p_ξ does not mean that subformula ξ must not be satisfied—its satisfaction is simply not required at this point. We denote by $\mathsf{proj}_{\mathsf{AP}}(\rho')$ the timed word obtained by hiding all the atomic propositions in AP_φ from ρ'. We also let $\mathsf{proj}_{\mathsf{AP}}(\mathcal{L}) = \{\mathsf{proj}_{\mathsf{AP}}(\rho') \mid \rho' \in \mathcal{L}\}$ for a timed language $\mathcal{L}$ over $\mathsf{AP} \cup \mathsf{AP}_\varphi$.

Formulae over $\mathsf{AP} \cup \mathsf{AP}_\varphi$. We now introduce some syntactic operations on Boolean combinations of atomic propositions in $\mathsf{AP} \cup \mathsf{AP}_\varphi$, that will be used to construct the component Büchi automata later. Specifically, for a subformula ψ of φ, we define formulae $\overline{\psi}$, $*\psi$, $\sim\psi$, and $\widehat{\psi}$.

The formula $\overline{\psi}$ is obtained from ψ by replacing all top-level temporal subformulae by their corresponding triggers. Formally, $\overline{\psi}$ is defined inductively as follows (where $p \in \mathsf{AP} \cup \mathsf{AP}_\varphi$):

$$\overline{\psi_1 \wedge \psi_2} = \overline{\psi_1} \wedge \overline{\psi_2} \qquad \overline{\psi} = \psi \text{ when } \psi \text{ is } \top \text{ or } \bot \text{ or } p \text{ or } \neg p$$
$$\overline{\psi_1 \vee \psi_2} = \overline{\psi_1} \vee \overline{\psi_2} \qquad \overline{\psi} = p_\psi \text{ when } \psi \text{ is } \psi_1 \mathbf{U}_I \psi_2 \text{ or } \psi_1 \mathbf{R}_I \psi_2 \,.$$

The formula $*\psi$, read as "do not pull the triggers of ψ", will be used to ensure that our component automata only follow the minimal models of the transition function of $\mathcal{A}_\varphi$ (we will see in Section 6 how crucial it is, for performance, to generate only minimal models). It is the conjunction of negations of all the atomic propositions in $\mathcal{P}_\psi$. As a concrete example, $*((\neg p \vee \psi_1 \mathbf{U} \psi_2) \wedge (q \vee \psi_3 \mathbf{R} (\psi_4 \mathbf{U} \psi_5))) = \neg p_{\psi_1 \mathbf{U} \psi_2} \wedge \neg p_{\psi_3 \mathbf{R}(\psi_4 \mathbf{U} \psi_5)}$. The formula $\sim\psi$ asserts that $\overline{\psi}$ is false and none of its triggers is activated: $\sim\psi = \neg\overline{\psi} \wedge *\psi$. Finally, the formula $\widehat{\psi}$ is defined as $\mathsf{mm}(\overline{\psi})$ where $\mathsf{mm}(\alpha)$ is defined inductively as follows:

$$\mathsf{mm}(p) = p \qquad \mathsf{mm}(\neg p) = \neg p \qquad \mathsf{mm}(\top) = \top \qquad \mathsf{mm}(\bot) = \bot$$
$$\mathsf{mm}(\alpha_1 \vee \alpha_2) = \big(\mathsf{mm}(\alpha_1) \wedge \sim\alpha_2\big) \vee \big(\mathsf{mm}(\alpha_2) \wedge \sim\alpha_1\big) \vee \big((\alpha_1 \vee \alpha_2) \wedge *\alpha_1 \wedge *\alpha_2\big)$$
$$\mathsf{mm}(\alpha_1 \wedge \alpha_2) = \mathsf{mm}(\alpha_1) \wedge \mathsf{mm}(\alpha_2)\,.$$

Intuitively, $\mathsf{mm}(\alpha)$ is satisfiable if and only if α is satisfiable, but $\mathsf{mm}(\alpha)$ only permits models of α that are minimal with respect to the triggers it contains:

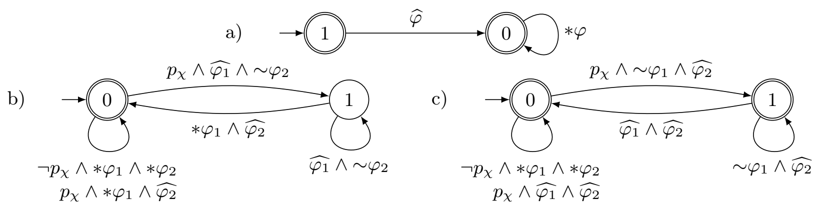
a)
1
$\widehat{\varphi}$
0
$*\varphi$
b)
0
$p_\chi \wedge \widehat{\varphi_1} \wedge \sim\varphi_2$
1
$*\varphi_1 \wedge \widehat{\varphi_2}$
$\neg p_\chi \wedge *\varphi_1 \wedge *\varphi_2$
$p_\chi \wedge *\varphi_1 \wedge \widehat{\varphi_2}$
$\widehat{\varphi_1} \wedge \sim\varphi_2$
c)
0
$p_\chi \wedge \sim\varphi_1 \wedge \widehat{\varphi_2}$
1
$\widehat{\varphi_1} \wedge \widehat{\varphi_2}$
$\neg p_\chi \wedge *\varphi_1 \wedge *\varphi_2$
$p_\chi \wedge \widehat{\varphi_1} \wedge \widehat{\varphi_2}$
$\sim\varphi_1 \wedge \widehat{\varphi_2}$

Fig. 3. The automata a) $\mathcal{C}_{init}$ and $\mathcal{C}_\chi$ for b) $\chi = \varphi_1 \mathbf{U} \varphi_2$, and c) $\chi = \varphi_1 \mathbf{R} \varphi_2$.

for $\mathsf{mm}(\alpha_1 \vee \alpha_2)$ to be true, either $\mathsf{mm}(\alpha_1)$ is true and α_2 does not hold, or vice versa, or $\alpha_1 \vee \alpha_2$ is indeed true, but not because of any of the triggers it contains.

Component Büchi automata for LTL. We are now ready to present the component Büchi automata that we consider, for an LTL formula φ. Instead of building a monolithic Büchi automaton $\mathcal{B}_\varphi$ directly from the alternating automaton, as in [19], we build small component Büchi automata, that are language-equivalent to the automaton $\mathcal{B}_\varphi$, once synchronised. There is an initial component $\mathcal{C}_{init}$, and a component Büchi automaton $\mathcal{C}_\chi$, for each $\chi \in \Phi$ (see Fig. 2). Consider the case where $\chi = \varphi_1 \mathbf{U} \varphi_2$ for instance. Component $\mathcal{C}_\chi$ has two locations 0 and 1 with the following intended meaning: $\mathcal{C}_\chi$ is in location 1 iff the trigger p_χ has been pulled in the past by $\mathcal{C}_{init}$, in which case $p_\chi \in \mathcal{P}_\varphi$, or by a unique component $\mathcal{C}_{\psi_1 \mathbf{U}_I \psi_2}$ (or $\mathcal{C}_{\psi_1 \mathbf{R}_I \psi_2}$) such that $p_\chi \in \mathcal{P}_{\psi_1}$ or $p_\chi \in \mathcal{P}_{\psi_2}$, and χ has not been satisfied yet. When component $\mathcal{C}_\chi$ is in location 1, we say that we have an *obligation* for χ. To satisfy this obligation, we must see a letter in the future where φ_2 holds. Thus, there is a self-loop on location 1 whose label ensures that φ_2 does not hold (because of $\sim\varphi_2$), while φ_1 still holds (this is ensured by $\widehat{\varphi_1}$, which also pulls a minimal set of triggers for φ_1 to be fulfilled). $\mathcal{C}_\chi$ moves back from 1 to 0 when φ_2 holds, while no trigger of φ_1 should be pulled at this instant (which is translated by $*\varphi_1$). From location 0, if we do not read trigger p_χ, nothing has to be checked and we do not pull any trigger. However, if p_χ is pulled, then, either φ_2 holds right away and the obligation is fulfilled immediately, or we jump to location 1.

Corrected an error here.

Example 4. Consider the LTL formula $\mathbf{G}(p \Rightarrow \mathbf{F}q)$ that can be rewritten into negative normal form as $\varphi = \bot \mathbf{R} (\neg p \vee \top \mathbf{U} q)$. Then, the three component Büchi automata $\mathcal{C}_{init}$, $\mathcal{C}_\varphi$ and $\mathcal{C}_{\mathbf{F}q}$, after the constraints on the transitions are simplified, are depicted on the top of Fig. 3, The automaton $\mathcal{C} = \mathcal{C}_{init} \times \mathcal{C}_\varphi \times \mathcal{C}_{\mathbf{F}q}$ is depicted in the middle. Once atomic propositions in AP_φ are projected away, one obtains an automaton isomorphic to the one at the bottom of the figure that accepts $[\![\varphi]\!]$.

Proposition 5. *For all* LTL *formulae* φ, $\mathsf{proj}_{\mathsf{AP}}([\![\mathcal{C}_{init} \times \prod_{\chi \in \Phi} \mathcal{C}_\chi]\!]) = [\![\varphi]\!]$.

We will provide a sketch of proof in the case of $\mathsf{MITL}_{0,\infty}$. A full proof can be found in the full version of this paper.

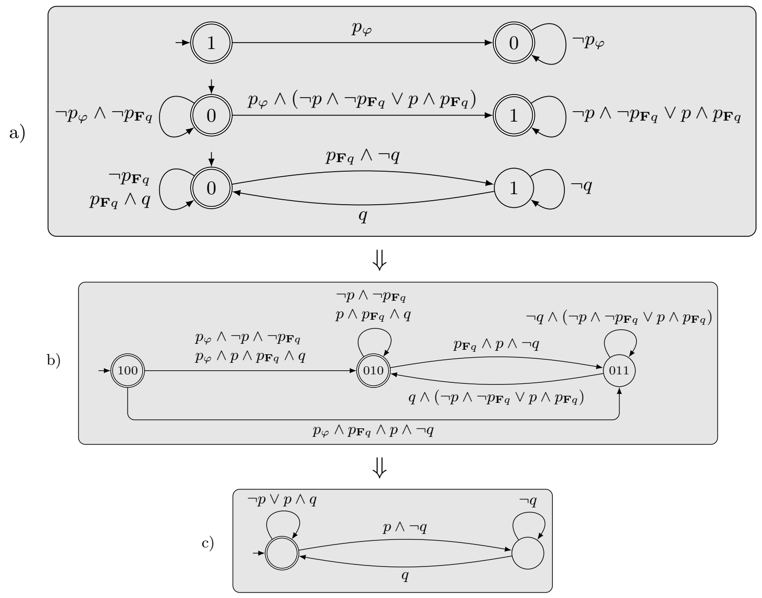
a)
p_φ
$\neg p_\varphi$
$\neg p_\varphi \wedge \neg p_{\mathbf{F}q}$
$p_\varphi \wedge (\neg p \wedge \neg p_{\mathbf{F}q} \vee p \wedge p_{\mathbf{F}q})$
$\neg p \wedge \neg p_{\mathbf{F}q} \vee p \wedge p_{\mathbf{F}q}$
$\neg p_{\mathbf{F}q}$
$p_{\mathbf{F}q} \wedge q$
$p_{\mathbf{F}q} \wedge \neg q$
$\neg q$
q
b)
$\neg p \wedge \neg p_{\mathbf{F}q}$
$p \wedge p_{\mathbf{F}q} \wedge q$
$\neg q \wedge (\neg p \wedge \neg p_{\mathbf{F}q} \vee p \wedge p_{\mathbf{F}q})$
$p_\varphi \wedge \neg p \wedge \neg p_{\mathbf{F}q}$
$p_\varphi \wedge p \wedge p_{\mathbf{F}q} \wedge q$
$p_{\mathbf{F}q} \wedge p \wedge \neg q$
100
010
011
$q \wedge (\neg p \wedge \neg p_{\mathbf{F}q} \vee p \wedge p_{\mathbf{F}q})$
$p_\varphi \wedge p_{\mathbf{F}q} \wedge p \wedge \neg q$
c)
$\neg p \vee p \wedge q$
$\neg q$
$p \wedge \neg q$
q

Fig. 4. a) Component Büchi automata for the formula $\varphi = \bot \,\mathbf{R}\, (\neg p \vee \top \,\mathbf{U}\, q)$; b) Büchi automaton obtained by the product of the components; c) Büchi automaton obtained by projecting away AP_φ (and merging two identical locations).

4 The case of $\mathsf{MITL}_{\mathbf{0},\boldsymbol{\infty}}$

We now describe how to lift the translation we described earlier to the timed operators of $\mathsf{MITL}_{0,\infty}$. The new components for $\mathbf{U}_{[0,a]}$, $\mathbf{R}_{[0,a]}$, and $\mathbf{R}_{[a,\infty)}$ are depicted in Fig. 4. They have the same shape as the components for untimed $\mathbf{U}$ and $\mathbf{R}$ (see Fig. 2); only the guards are changed to reflect the more involved semantics of the timed operators. Observe that these automata have only one clock. To understand why this is sufficient, consider the formula $\mathbf{G}(p \Rightarrow \chi)$ with $\chi = p \,\mathbf{U}_{[0,2]}\, q$. After reading $(\{p\}, 0)(\{p\}, 0.4)(\{p\}, 1)$, the OCATA $\mathcal{A}_\varphi$ reaches the configuration $\{(\varphi, 0), (\chi, 0), (\chi, 0.6), (\chi, 1)\}$, meaning intuitively that, to satisfy the formula, one must fulfil three obligations related to χ: to see q's within 2, 1.4, and 1 time units, respectively. Hence, we can store the earliest obligation, corresponding to $(\chi, 1)$, only (as already observed in [11]). Indeed, if the corresponding instance of χ is satisfied, it means that there will be a q occurring within less that 1 time unit, which will also satisfy all the other obligations. More generally, for operators $\mathbf{U}_{[0,a]}$ and $\mathbf{R}_{[a,\infty)}$, it is always the case that only the oldest obligation has to be stored, while for operators $\mathbf{R}_{[0,a]}$ and $\mathbf{U}_{[a,\infty)}$, only the earliest obligation has to be stored. This is translated in the components by the absence/presence of resets on transitions that leave state 1 (which is reached when an obligation is currently active) and read p_χ.

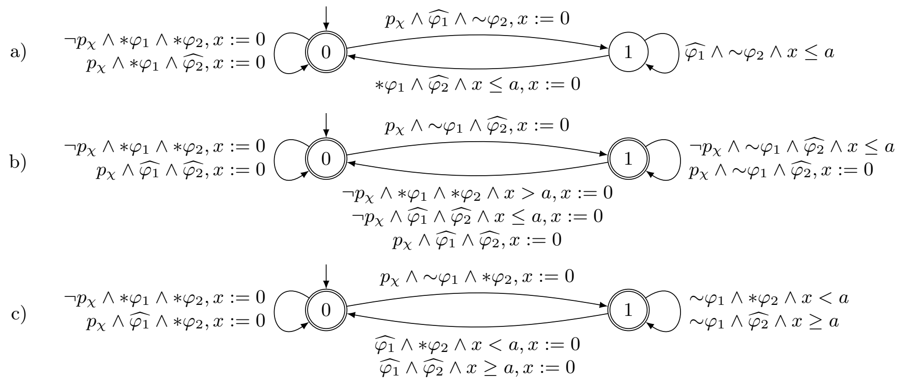
a)
$\neg p_\chi \wedge *\varphi_1 \wedge *\varphi_2, x := 0$
$p_\chi \wedge *\varphi_1 \wedge \widehat{\varphi_2}, x := 0$
0
$p_\chi \wedge \widehat{\varphi_1} \wedge \sim\varphi_2, x := 0$
$*\varphi_1 \wedge \widehat{\varphi_2} \wedge x \leq a, x := 0$
1
$\widehat{\varphi_1} \wedge \sim\varphi_2 \wedge x \leq a$
b)
$\neg p_\chi \wedge *\varphi_1 \wedge *\varphi_2, x := 0$
$p_\chi \wedge \widehat{\varphi_1} \wedge \widehat{\varphi_2}, x := 0$
0
$p_\chi \wedge \sim\varphi_1 \wedge \widehat{\varphi_2}, x := 0$
$\neg p_\chi \wedge *\varphi_1 \wedge *\varphi_2 \wedge x > a, x := 0$
$\neg p_\chi \wedge \widehat{\varphi_1} \wedge \widehat{\varphi_2} \wedge x \leq a, x := 0$
$p_\chi \wedge \widehat{\varphi_1} \wedge \widehat{\varphi_2}, x := 0$
1
$\neg p_\chi \wedge \sim\varphi_1 \wedge \widehat{\varphi_2} \wedge x \leq a$
$p_\chi \wedge \sim\varphi_1 \wedge \widehat{\varphi_2}, x := 0$
c)
$\neg p_\chi \wedge *\varphi_1 \wedge *\varphi_2, x := 0$
$p_\chi \wedge \widehat{\varphi_1} \wedge *\varphi_2, x := 0$
0
$p_\chi \wedge \sim\varphi_1 \wedge *\varphi_2, x := 0$
$\widehat{\varphi_1} \wedge *\varphi_2 \wedge x < a, x := 0$
$\widehat{\varphi_1} \wedge \widehat{\varphi_2} \wedge x \geq a, x := 0$
1
$\sim\varphi_1 \wedge *\varphi_2 \wedge x < a$
$\sim\varphi_1 \wedge \widehat{\varphi_2} \wedge x \geq a$

Fig. 5. One-clock TA for the subformulae: a) $\chi = \varphi_1 \mathbf{U}_{[0,a]} \varphi_2$, b) $\chi = \varphi_1 \mathbf{R}_{[0,a]} \varphi_2$, and c) $\chi = \varphi_1 \mathbf{R}_{[a,\infty)} \varphi_2$.

For $\chi = \varphi_1 \mathbf{U}_{[a,\infty)} \varphi_2$, the situation is slightly more complicated, although one clock is again sufficient. The corresponding component is in Fig. 5 and has four locations. To understand why, consider the case when there is an obligation for χ associated with the current valuation $v \geq a$ of clock x ($\mathcal{C}_\chi$ is in location 1), the current letter contains p_χ and satisfies both $\widehat{\varphi_1}$ and $\widehat{\varphi_2}$. Since the trigger has been pulled, $\mathcal{C}_\chi$ should stay in the non-accepting location 1. On the other hand, the pending obligation has also been fulfilled, and an accepting location should be visited. So, instead of staying in 1, $\mathcal{C}_\chi$ moves to $1'$ in this case: $1'$ is a copy of 1 as far as transitions are concerned, but it is accepting. The location $1''$ is used to deal with the situation where p_χ is launched infinitely often but no two occurrences of p_χ are separated by more than a time units; in this case, we non-deterministically move to $1''$ and add a new obligation (by resetting x) after the current obligation has been verified. Notice that this problem cannot occur for $\varphi_1 \mathbf{U} \varphi_2$, or $\varphi_1 \mathbf{U}_{[0,a]} \varphi_2$: in these cases, the new obligation is immediately fulfilled, and the automaton moves to the initial, accepting, location.

We will now present the extension of Proposition 5 to the case of $\mathsf{MITL}_{0,\infty}$. The proof will rely on a decoration of the OCATA $\mathcal{A}_\varphi$ with the triggers. Formally, we replace $\Gamma(L)$ (defined on page 5) by $\Gamma_\varphi(L)$, the set of formulae defined by

$$\gamma := \top \mid \bot \mid \gamma \vee \gamma \mid \gamma \wedge \gamma \mid \ell \mid x \bowtie c \mid x_{\mathcal{Q}}.\gamma$$

where $c \in \mathbb{N}$, $\bowtie \in \{\leq, <, \geq, >\}$, $\ell \in L$, and $\mathcal{Q} \subseteq \mathsf{AP}_\varphi$. Then, we redefine the transition function of $\mathcal{A}_\varphi$ as a function from $L \times \Sigma$ to $\Gamma_\varphi(L)$ and label the clock x in $x.\gamma$ by the triggers it pulls, i.e. we replace the rules for $\mathbf{U}_I$ and $\mathbf{R}_I$ by

$$\delta(\varphi_1 \mathbf{U}_I \varphi_2, \sigma) = (x_{\mathcal{P}_{\varphi_2}}.\delta(\varphi_2, \sigma) \wedge x \in I) \vee (x_{\mathcal{P}_{\varphi_1}}.\delta(\varphi_1, \sigma) \wedge \varphi_1 \mathbf{U}_I \varphi_2 \wedge x \leq \sup I)$$
$$\delta(\varphi_1 \mathbf{R}_I \varphi_2, \sigma) = (x_{\mathcal{P}_{\varphi_2}}.\delta(\varphi_2, \sigma) \vee x \notin I) \wedge (x_{\mathcal{P}_{\varphi_1}}.\delta(\varphi_1, \sigma) \vee \varphi_1 \mathbf{R}_I \varphi_2 \vee x > \sup I).$$

These decorations do not affect the evaluation of models of a transition. However they allow us to define the set of triggers associated with a model M. Consider a

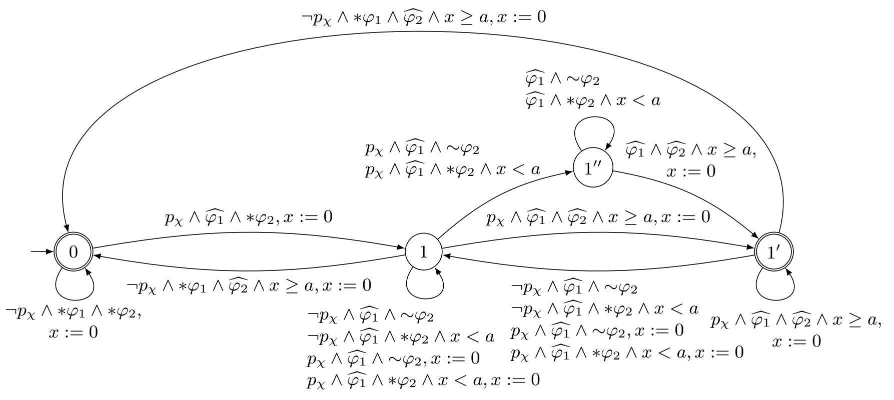
$\neg p_\chi \wedge *\varphi_1 \wedge \widehat{\varphi_2} \wedge x \geq a, x := 0$
$\widehat{\varphi_1} \wedge \sim\varphi_2$
$\widehat{\varphi_1} \wedge *\varphi_2 \wedge x < a$
$p_\chi \wedge \widehat{\varphi_1} \wedge \sim\varphi_2$
$p_\chi \wedge \widehat{\varphi_1} \wedge *\varphi_2 \wedge x < a$
$1''$
$\widehat{\varphi_1} \wedge \widehat{\varphi_2} \wedge x \geq a,$
$x := 0$
$p_\chi \wedge \widehat{\varphi_1} \wedge *\varphi_2, x := 0$
$p_\chi \wedge \widehat{\varphi_1} \wedge \widehat{\varphi_2} \wedge x \geq a, x := 0$
0
1
$1'$
$\neg p_\chi \wedge *\varphi_1 \wedge \widehat{\varphi_2} \wedge x \geq a, x := 0$
$\neg p_\chi \wedge \widehat{\varphi_1} \wedge \sim\varphi_2$
$\neg p_\chi \wedge \widehat{\varphi_1} \wedge *\varphi_2 \wedge x < a$
$p_\chi \wedge \widehat{\varphi_1} \wedge \sim\varphi_2, x := 0$
$p_\chi \wedge \widehat{\varphi_1} \wedge *\varphi_2 \wedge x < a, x := 0$
$\neg p_\chi \wedge *\varphi_1 \wedge *\varphi_2,$
$x := 0$
$\neg p_\chi \wedge \widehat{\varphi_1} \wedge \sim\varphi_2$
$\neg p_\chi \wedge \widehat{\varphi_1} \wedge *\varphi_2 \wedge x < a$
$p_\chi \wedge \widehat{\varphi_1} \wedge \sim\varphi_2, x := 0$
$p_\chi \wedge \widehat{\varphi_1} \wedge *\varphi_2 \wedge x < a, x := 0$
$p_\chi \wedge \widehat{\varphi_1} \wedge \widehat{\varphi_2} \wedge x \geq a,$
$x := 0$

Fig. 6. One-clock TA for the subformula $\chi = \varphi_1 \, \mathbf{U}_{[a,\infty)} \, \varphi_2$.

formula $\gamma \in \mathit{\Gamma}_\varphi(L)$ and a minimal model M of γ with respect to a clock valuation $v \in \mathbb{R}^+$. We denote by $\mathsf{trig}(M, \gamma, v)$ the unique subset of AP_φ inductively defined by:

– $\mathsf{trig}(M, \gamma_1 \wedge \gamma_2, v) = \mathsf{trig}(M, \gamma_1, v) \cup \mathsf{trig}(M, \gamma_2, v)$;
– $\mathsf{trig}(M, \gamma_1 \vee \gamma_2, v) = \begin{cases} \mathsf{trig}(M, \gamma_1, v) & \text{if } M \models_v \gamma_1 \\ \mathsf{trig}(M, \gamma_2, v) & \text{otherwise;} \end{cases}$
– $\mathsf{trig}(M, x_{\mathcal{Q}}.\gamma, v) = \mathcal{Q} \cup \mathsf{trig}(M, \gamma, 0)$; and $\mathsf{trig}(M, \gamma, v) = \emptyset$ otherwise.

Proposition 6. *For all* $\mathsf{MITL}_{0,\infty}$ *formulae* φ, $\mathsf{proj}_{\mathsf{AP}}([\![\mathcal{C}_{init} \times \prod_{\chi \in \mathit{\Phi}} \mathcal{C}_\chi]\!]) = [\![\varphi]\!]$.

Proof (Sketch of proof). We rely on Theorem 2 showing that $[\![\varphi]\!] = [\![\mathcal{A}_\varphi]\!]$. Therefore, we must relate the synchronous product of all component one-clock Büchi timed automata $\mathcal{C} = \mathcal{C}_{init} \times \prod_{\chi \in \mathit{\Phi}} \mathcal{C}_\chi$, with the very-weak OCATA $\mathcal{A}_\varphi$. We consider a timed word $\rho \in [\![\mathcal{A}_\varphi]\!]$ and an accepting run $G = (V, \rightarrow)$ of $\mathcal{A}_\varphi$ over ρ. We also let $K_0, K_1, \ldots$ be the sequence of configurations associated with the run G.

First, we construct a timed word ρ' over $2^{\mathsf{AP} \cup \mathsf{AP}_\varphi}$ from ρ and G, by adding the triggers in AP_φ according to the minimal models selected in the run G. Precisely, let $\rho = (\sigma_1, \tau_1)(\sigma_2, \tau_2)\ldots$. For all $i \geq 0$, we associate every configuration (ℓ, v) of K_i with the pair $(\gamma_{\ell,v}, M_{\ell,v})$ of transition formula and minimal model chosen in the run G when reading σ_{i+1} after a delay $\tau_{i+1} - \tau_i$. We then gather all the triggers in a set $\mathcal{Q}_i = \bigcup_{(\ell,v) \in K_i} \mathsf{trig}(M_{\ell,v}, \gamma_{\ell,v}, v + \tau_{i+1} - \tau_i)$, and let $\rho' = (\sigma_1 \cup \mathcal{Q}_1, \tau_1)(\sigma_2 \cup \mathcal{Q}_2, \tau_2)\ldots$ be the Hintikka sequence we will read in $\mathcal{C}$.

Second, we map the sequence $(K_i)_{i \geq 0}$ of configurations in $\mathcal{A}_\varphi$ to a run $(C_i)_{i \geq 0}$ in $\mathcal{C}$. For all $i \geq 0$, we let $C_i = (b_{init}, (b_\chi, v_\chi)_{\chi \in \mathit{\Phi}})$ with b_{init} being the configuration of $\mathcal{C}_{init}$ (that is a TA without clocks), and (b_χ, v_χ) the configuration of $\mathcal{C}_\chi$, for all $\chi \in \mathit{\Phi}$, defined by: $b_{init} = 1$ if and only if $i = 0$ (remember that $K_0 = \{(\varphi_{init}, 0, 0)\}$, and that location φ_{init} of $\mathcal{A}_\varphi$ is no longer reached afterwards), $b_\chi = 1$ if and only if $i > 0$ and $(\chi, v) \in K_i$, while $v_\chi(x) = \max_{(\chi,v) \in K_i} v$

if $\chi = \varphi_1 \mathbf{U}_{[0,a]} \varphi_2$ or $\chi = \varphi_1 \mathbf{R}_{[a,\infty)} \varphi_2$, and $v_\chi(x) = \min_{(\chi,v)\in K_i} v$ otherwise (according to the previous remark regarding the fact that only the oldest/earliest obligation should be stored).

Then, we can show that $C_0, C_1, \dots$ is an accepting run of $\mathcal{C}$, by proving that the projection of the configurations on each component is an accepting run of the component (the generalised Büchi acceptance condition on $\mathcal{C}$ is fulfilled exactly when each Büchi acceptance condition of the components is fulfilled). This is achieved by a close inspection of the constraints over transitions of the components (see Appendix B.1 for more details).

The proof of the reciprocal consists in building a DAG G that is an accepting run of $\mathcal{A}_\varphi$ over $\mathsf{proj}_{\mathsf{AP}}(\rho')$, from an accepting run of $\mathcal{C}$ over $\rho' \in [\![\mathcal{C}]\!]$. This is done level after level (initiating with a single root $(\varphi_{init}, 0, 0)$). The detailed construction and proof can be found in Appendix B.2. $\square$

5 Handling full MITL

We finally extend our translation to the full MITL logic, i.e. considering operators $\mathbf{U}_{[a,b]}$ and $\mathbf{R}_{[a,b]}$ with $0 < a < b < +\infty$. For these two operators, we cannot rely on a single clock in the components anymore. For instance, consider the formula $\varphi = \mathbf{G}(p \Rightarrow \chi)$ with $\chi = \mathbf{F}_{[1,2]} q$. Imagine that $\mathcal{A}_\varphi$ reads the prefix $(\{p\}, 0)(\{p\}, 0.5)$. At this point, its configuration is $\{(\varphi, 0), (\chi, 0), (\chi, 0.5)\}$. It is not possible, as before, to drop one of the two states in location χ as the following futures can happen: if we read $(\{q\}, 1)$, obligation $(\chi, 0)$ is fulfilled but not $(\chi, 0.5)$; while reading $(\{q\}, 2.5)$ to fulfil obligation $(\chi, 0.5)$ fails to satisfy obligation $(\chi, 0)$. Therefore, we must keep track of the two obligations separately.

It is not even clear a priori how to find a bound on the number of clocks. This is the role of the *interval semantics* of very-weak OCATA that has been introduced in [11] over infinite words. In this interpretation of OCATA, valuations of the clocks are no longer *points* but *intervals* meant to approximate sets of (punctual) valuations: $(\ell, [\alpha, \beta])$ means that there *are* clock copies with valuations α and β in ℓ, and that there *could be* more copies in ℓ with valuations in (α, β). In this semantics, we can *merge* non-deterministically two copies $(\ell, [\alpha_1, \beta_1])$ and $(\ell, [\alpha_2, \beta_2])$ into a single copy $(\ell, [\alpha_1, \beta_2])$ (assuming $\alpha_1 \leq \beta_2$), in order to keep the number of clock copies below a fixed threshold, and thus obtain an equivalent TA. It has been shown in [11] that, for the OCATA $\mathcal{A}_\varphi$, with $\varphi \in$ MITL, the interval semantics is sufficient to retain the language of the formula, with TAs having at most $M(\varphi) = |\varphi| \times \max_{I \in \mathcal{I}_\varphi}(\max(4 \times \lceil \inf(I)/|I| \rceil + 2, 2 \times \lceil \sup(I)/|I| \rceil + 2))$ clocks, where $\mathcal{I}_\varphi$ is the set of intervals that appear in φ: more precisely, each subformula with topmost operator $\mathbf{U}_I$ (respectively, $\mathbf{R}_I$) contributes to $4 \times \lceil \inf(I)/|I| \rceil + 2$ (respectively, $2 \times \lceil \sup(I)/|I| \rceil + 2$) more clocks.

Our solution is twofold in this context: (i) we propose a better approximation by intervals that allows us to cut, up to a factor of two, the number of clock copies we must keep in memory; (ii) instead of a single TA, as in [11], we provide a GBTA, with one component per temporal subformula of φ. The component TA

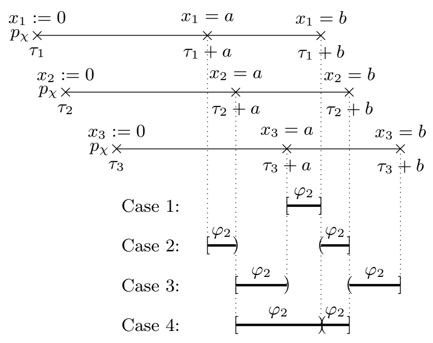

Fig. 7. How to split cases to fulfil formula $\chi = \varphi_1 \mathbf{U}_{[a,b]} \varphi_2$

are much more involved than for $\mathsf{MITL}_{0,\infty}$, thus we do not give them explicitly, but rather explain the main ideas.

We start by developing our new merging strategy on an example, to explain how it is different from [11]. Consider $\chi = \varphi_1 \mathbf{U}_{[a,b]} \varphi_2$ with $0 < a < b < +\infty$; and a situation, as in Fig. 6, where new triggers p_χ are pulled at three positions, of time stamps τ_1, τ_2, and τ_3. We suppose that φ_1 holds at those three positions. The picture presents four different cases corresponding to the four possible situations where the occurrence of φ_2 fulfil the three pending obligations. Case 1 is when a position in $[\tau_3 + a, \tau_1 + b]$ satisfies φ_2, hence all three obligations are resolved at once. This case can be checked using only clocks x_3 and x_2. In the other cases, at least two positions satisfying φ_2 are needed. In case 2, the first obligation is resolved by an occurrence of φ_2 with time stamp in $[\tau_1 + a, \tau_2 + a)$, while the two others are resolved by an occurrence in $(\tau_1 + b, \tau_2 + b]$. Thus, case 2 can be checked using only clocks x_1 and x_2. Now consider the remaining cases: if no occurrences of φ_2 appear in $[\tau_1 + a, \tau_2 + a) \cup [\tau_3 + a, \tau_1 + b]$, one occurrence of φ_2 must necessarily happen in $[\tau_2 + a, \tau_3 + a)$, while the other should be in $(\tau_1 + b, \tau_3 + b]$, which would require the three clocks x_1, x_2 and x_3 to be checked. We avoid this need for 3 clocks by splitting this situation into two further cases (3 and 4) that can be checked with only two clocks. Indeed, case 3 can be checked using x_2 and x_3 only; and case 4 using x_2 and x_1 only.

Notice that two main situations appear: one where formula φ_2 should be fulfilled in a single interval whose endpoints use two distinct clocks, another where φ_2 should be fulfilled in two (semi-open) intervals whose both endpoints use the same two distinct clocks. In each of these cases, it must be understood how a new trigger p_χ modifies the situation. With only one interval, if a new obligation for φ_2 appear as a new interval $[\tau + a, \tau + b]$, either the new obligation is implied by the current ones, in which case we are done, or two intervals intersect and we split (non-deterministically) in two cases (the intersection, and the symmetric difference), or they are disjoint and we keep both intervals in memory. Notice that the latter situation cannot happen too often, since intervals are non-singular; precisely this will happen at most $\lceil (\inf(I)+1)/|I| \rceil$ times. With two intervals, either the new obligation is already implied by current obligations or $[\tau + a, \tau + b]$ is not implied by the current obligations and we add this new interval in memory as before (again, this cannot happen more than $\lceil (\inf(I) + 1)/|I| \rceil$ times).

In the end, following the same lines as [11] to obtain a TA from a bound on the number of copies in $\mathcal{A}_\varphi$, this allows us to build a TA $\mathcal{C}_\chi$ for each subformula $\chi = \varphi_1 \mathbf{U}_{[a,b]} \varphi_2$ with a number of clocks $N(\chi) = 2 \times \lceil (\inf(I) + 1)/|I| \rceil + 2$ (the two additional clocks are to deal easily with some special cases), up to half the previous bound in [11]. In the locations, we can handle the clocks in pairs and use a queue of size $N(\chi)/2$ to keep track of which non-deterministic case (regarding the intervals) we fall into and which clocks are used to represent the endpoints of intervals. It follows that the number of locations is exponential in $N(\chi)$. A similar construction, using $2 \times \lceil (\inf(I) + 1)/|I| \rceil$ clocks, holds for subformulae $\varphi_1 \mathbf{R}_{[a,b]} \varphi_2$: here we have to consider unions of intervals, which are easier.

Theorem 7. *For all* MITL *formulae* φ, $\mathsf{proj}_{\mathsf{AP}}(\llbracket \mathcal{C}_{init} \times \prod_{\chi \in \Phi} \mathcal{C}_\chi \rrbracket) = \llbracket \varphi \rrbracket$.

Proof (Sketch of proof). We follow the same scheme of proof as for Proposition 6, relating accepting runs of $\mathcal{A}_\varphi$ with accepting runs of $\mathcal{C} = \mathcal{C}_{init} \times \prod_{\chi \in \Phi} \mathcal{C}_\chi$. To show $\llbracket \mathcal{A}_\varphi \rrbracket \subseteq \mathsf{proj}_{\mathsf{AP}}(\llbracket \mathcal{C} \rrbracket)$, e.g., we use the same construction of the Hintikka sequence over $2^{\mathsf{AP} \cup \mathsf{AP}_\varphi}$. The definition of the run $C_0, C_1, \ldots$ in $\mathcal{C}$ with respect to the run G in $\mathcal{A}_\varphi$ is more complex though, in particular because $\mathcal{C}$ is now non-deterministic (with respect to $\mathsf{AP} \cup \mathsf{AP}_\varphi$). However, notice that we have the full run G so that we know in advance *what the reason is for the obligation associated to a given trigger to be fulfilled in the future*. We use this knowledge to choose the good case in the previous disjunction of cases in a component $\mathcal{C}_{\varphi_1 \mathbf{U}_{[a,b]} \varphi_2}$. The proof then follows exactly the same lines. ☐

6 Implementation

We have implemented our translation from MITL formulae to generalised Büchi timed automata in a tool called MightyL, written in OCaml. From a formula φ, it produces the GBTA $\mathcal{C}$, described in previous sections, output in the XML format developed by Uppaal, as well as the generalised Büchi condition written as a very simple LTL formula. We can then use Uppaal [26] to check the satisfiability of φ over finite timed words, or LTSmin [23] with opaal front-end to check satisfiability over infinite timed words. To maximise compatibility, we use a Boolean variable for each atomic proposition and a `loc` variable in each component for the current location. The synchronisation is done in a round-robin fashion with a counter variable `N`: initially, `N` is set to 0, allowing the model (to be model-checked) to take a transition and set the truth values of the atomic propositions. Then, `N` loops from 1 to the number of components of $\mathcal{C}$, allowing each component to read the atomic propositions and take a corresponding transition. Finally, `N` is set back to 0 and we start over again. For the finite-word case, this also enables to check that all components have been synchronised properly (`N` $= 0$) while in the final location. Our tool is publicly available, and can even be

Table 1. Execution times for the satisfiability check of benchmarks of [11, 19]. For LTSmin, the three times reported correspond to the translation to C++, the compile time and the actual check, respectively.

Formula	MightyL	LTSmin	Uppaal
$F(5,[0,\infty))$	9ms	3.48s/2.18s/0.12s	0.75s
$F(5,[0,2])$	7ms	3.76s/2.23s/0.15s	0.84s
$F(5,[2,\infty))$	6ms	3.76s/2.26s/0.91s	1.64s
$F(3,[1,2])$	70ms	6m5.15s/38.01s/0.22s	9.00s
$F(5,[1,2])$	70ms	>15m	2m6s
$G(5,[0,\infty))$	10ms	3.83s/2.43s/0.05s	0.75s
$G(5,[0,2])$	10ms	4.01s/2.51s/0.10s	0.82s
$G(5,[2,\infty))$	9ms	4.06s/2.47s/0.04s	0.85s
$G(5,[1,2])$	15ms	7.81s/2.99s/0.09s	1.12s
$\mu(1)$	13ms	-	0.39s
$\mu(2)$	21ms	-	2.33s
$\mu(3)$	76ms	-	15.77s
$\mu(4)$	87ms	-	2m23s

Formula	MightyL	LTSmin	Uppaal
$U(5,[0,\infty))$	16ms	1.90s/1.44s/0.05s	0.41s
$U(5,[0,2])$	8ms	2.08s/1.54s/0.06s	0.42s
$U(5,[2,\infty))$	8ms	2.08s/1.53s/0.09s	0.52s
$U(3,[1,2])$	49ms	4m0.14s/23.54s/0.09s	4.92s
$U(5,[1,2])$	97ms	>15m	21.80s
$R(5,[0,\infty))$	7ms	1.86s/1.42s/0.03s	0.40s
$R(5,[0,2])$	7ms	1.97s/1.44s/0.03s	0.40s
$R(5,[2,\infty))$	7ms	1.92s/1.42s/0.03s	0.42s
$R(5,[1,2])$	10ms	5.37s/2.16s/0.04s	0.62s
$\theta(1,[100,1000])$	9ms	1.88s/1.74s/0.04s	0.25s
$\theta(2,[100,1000])$	13ms	5.04s/3.17s/0.19s	0.86s
$\theta(3,[100,1000])$	14ms	36.57s/16.27s/3.20s	21.84s
$\theta(4,[100,1000])$	15ms	5m30s/4m18s/2m16s	18m39s

run directly on the website `http://www.ulb.ac.be/di/verif/mightyl`. Compared to the simplified version we studied in this article, MightyL also allows for (semi-)open intervals. Since it can also deal with next and dual-next operators, we can verify formulae like $\neg\mathbf{X}_{[1,2)}p$. All the following tests have been performed on a MacBook Pro 2.7GHz with 8Go RAM.

We check the satisfiability of MITL formulae on examples, inspired by benchmarks of [11,19]. For $k \in \mathbb{N}$ and an interval I, we consider the satisfiable formulae: $F(k,I) = \bigwedge_{i=1}^{k}\mathbf{F}_I p_i$, $G(k,I) = \bigwedge_{i=1}^{k}\mathbf{G}_I p_i$, $U(k,I) = (\dots(p_1\mathbf{U}_I p_2)\mathbf{U}_I\dots)\mathbf{U}_I p_k$, $R(k,I) = (\dots(p_1\,\mathbf{R}_I\,p_2)\,\mathbf{R}_I\,\dots)\,\mathbf{R}_I\,p_k$, and $\theta(k,I) = \neg((\bigwedge_{i=1}^{k}\mathbf{GF}p_i) \Rightarrow \mathbf{G}(q \Rightarrow \mathbf{F}_I r))$. We also consider an example inspired from motion planning problems via MITL specifications as in [24,30]. In our benchmark, a robot must visit some target points $t_1, t_2, t_3, \dots, t_k$ within given time frames (in our case, t_i must be seen in time frame $[3(i-1), 3i]$), while enforcing a safety condition $\mathbf{G}\neg p$. This specification is modelled by the satisfiable MITL formula $\mu(k) = \bigwedge_{i=1}^{k}\mathbf{F}_{[3(i-1),3i]}t_i \wedge \mathbf{G}\neg p$. In Table 1, we report on the time taken by the execution of MightyL; LTSmin (split into the time taken by the Python translation of the model in C++, the C++ compile time, and the time for the actual check); and Uppaal, on all these examples (for the motion planning, only finite words are relevant, hence we report only on the Uppaal running time).

We also report on benchmarks found in [16], where the debugging of formal specifications of cyber-physical systems is reduced to MITL *non-satisfiability*. More precisely formulae should be checked for validity and redundancy. A formula φ is *valid* (or a tautology) if $\neg\varphi$ is *not satisfiable*. Conjunct φ_1 of formula $\varphi = \bigwedge_{i=1}^{k}\varphi_i$ is *redundant* iff $\bigwedge_{i=2}^{k}\varphi_i$ implies φ_1. This is true iff $\psi = \bigwedge_{i=2}^{k}\varphi_i \Rightarrow \varphi_1$ is valid, i.e. iff $\neg\psi$ is *not satisfiable*. For instance, $\mathbf{F}_{[0,30]}p$ is redundant in $\mathbf{F}_{[0,30]}p \wedge \mathbf{F}_{[0,20]}p$, and $\mathbf{G}_{[0,20]}\mathbf{F}_{[0,20]}p$ is redundant in $\mathbf{G}_{[0,20]}\mathbf{F}_{[0,20]}p \wedge \mathbf{G}_{[0,40]}p \wedge \mathbf{F}_{[20,40]}\top$. We check the validity and redundancy of several formulae considered in [16], and we summarise some of our results in Table 2: we also copy the ex-

Table 2. Validity and redundancy checking of MITL formulae

Formula	MightyL	LTSmin	Uppaal	[16]
$\mathbf{F}_{[0,30]}(p \Rightarrow \mathbf{G}_{[0,20]}p)$ valid	7ms	0.98s	0.32s	7s
$\mathbf{G}_{[0,30]}\neg p \vee \mathbf{F}_{[0,20]}p$ valid	7ms	0.95s	0.14s	not considered
$\mathbf{F}_{[0,30]}p \wedge \mathbf{F}_{[0,20]}p$ redundant	13ms	1.99s	0.44s	14s
$\mathbf{G}_{[0,20]}\mathbf{F}_{[0,20]}p \wedge \mathbf{G}_{[0,40]}p \wedge \mathbf{F}_{[20,40]}\top$ redundant	22ms	1m26s	2.63s	not considered

ecution time reported in [16] for these checks.[8] We also consider new formulae specific to our pointwise semantics.

Remember that one of the difficulties of the constructions of the component Büchi timed automata corresponds to the *minimal model* simplification $\mathsf{mm}(\varphi)$. However, our components are still correct if we replace everywhere $\mathsf{mm}(\varphi)$ by φ (in particular, $\widehat{\varphi}$ simply becomes $\overline{\varphi}$). On some examples of the previous benchmarks, the influence on the execution time of the satisfiability checks is tremendous (differences on the execution time of MightyL are small, and not that important since the tool always answers in less than a second). For instance, over $F(5,[0,\infty))$, LTSmin shows a 17% overhead. For $F(5,[0,2])$, LTSmin experiences a 5% overhead, while Uppaal has a 12% overhead. For formulae $F(5,[2,\infty))$, $F(3,[1,2])$, $F(5,[1,2])$, the situation is even worse since Uppaal does not even respond before the timeout of fifteen minutes. LTSmin does not answer anymore on $F(3,[1,2])$ before the timeout. On the motion planning example, the overhead is also important for Uppaal, e.g. 80% for μ_2, and, for μ_3 and μ_4, Uppaal does not answer anymore before the timeout. Finally, on the two unsatisfiable examples of the redundancy check, LTSmin and Uppaal present an overhead of 70%/3% and 630%/230%, respectively.

7 Conclusion and perspectives

In this work, we proposed a new compositional construction from MITL to timed automata which we implemented the tool MightyL, enabling easy automata-based model-checking of full MITL. For future work, since the structure of the formula is preserved in our construction, we want to investigate antichain-based heuristics to allow more performance boost. For MightyL, we plan to add native support for ECL operators which eases the writing of specifications, as well as past modalities/counting modalities.

Acknowledgements. We thank the reviewers of this article that help us clarify its overall presentation. The third author would like to thank Andreas Engelbredt Dalsgaard, Alfons Laarman and Jeroen Meijer for their technical help with opaal and LTSmin.

[8] These numbers are only for reference and should not be taken as a direct comparison since, contrary to us, [16] considered a bounded continuous semantics of MITL.

References

1. Abid, N., Dal-Zilio, S., Botlan, D.L.: A formal framework to specify and verify real-time properties on critical systems. International Journal of Critical Computer-Based Systems 5(1/2), 4–30 (2014)
2. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
3. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. Journal of the ACM 43(1), 116–146 (1996)
4. Alur, R., Henzinger, T.A.: Real-time logics: Complexity and expressiveness. Information and Computation 104(1), 35–77 (1993)
5. Barnat, J., Brim, L., Havel, V., Havlícek, J., Kriho, J., Lenco, M., Rockai, P., Still, V., Weiser, J.: DivinE 3.0 - an explicit-state model checker for multithreaded C & C++ programs. In: CAV'13. LNCS, vol. 8044, pp. 863–868. Springer (2013)
6. Bartocci, E., Bortolussi, L., Nenzi, L.: A temporal logic approach to modular design of synthetic biological circuits. In: CMSB'13. LNCS, vol. 8130, pp. 164–177. Springer (2013)
7. Bersani, M.M., Rossi, M., San Pietro, P.: A tool for deciding the satisfiability of continuous-time metric temporal logic. Acta Informatica 53(2), 171–206 (2016)
8. Bloem, R., Cimatti, A., Pill, I., Roveri, M.: Symbolic implementation of alternating automata. International Journal of Foundations of Computer Science 18(4), 727–743 (2007)
9. Bouyer, P., Colange, M., Markey, N.: Symbolic optimal reachability in weighted timed automata. In: CAV'16. LNCS, vol. 9779, pp. 513–530. Springer (2016)
10. Brihaye, T., Estiévenart, M., Geeraerts, G.: On MITL and alternating timed automata. In: FORMATS'13. LNCS, vol. 8053, pp. 47–61. Springer (2013)
11. Brihaye, T., Estiévenart, M., Geeraerts, G.: On MITL and alternating timed automata of infinite words. In: FORMATS'14. LNCS, vol. 8711. Springer (2014)
12. Bulychev, P.E., David, A., Larsen, K.G., Li, G.: Efficient controller synthesis for a fragment of $\mathrm{MTL}_{0,\infty}$. Acta Informatica 51(3-4), 165–192 (2014)
13. Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV2: An opensource tool for symbolic model checking. In: CAV'02. LNCS, vol. 2404, pp. 359–364. Springer (2002)
14. Claessen, K., Een, N., Sterin, B.: A circuit approach to LTL model checking. In: FMCAD'13. IEEE (2013)
15. De Moura, L., Bjørner, N.: Satisfiability modulo theories: introduction and applications. Communications of the ACM 54(9), 69–77 (Sep 2011)
16. Dokhanchi, A., Hoxha, B., Fainekos, G.: Formal requirement debugging for testing and verification of cyber-physical systems. Research Report 1607.02549, arXiv (2016)
17. D'Souza, D., Matteplackel, R.: A clock-optimal hierarchical monitoring automaton construction for mitl. Research Report 2013-1, IIS (2013), `http://www.csa.iisc.ernet.in/TR/2013/1/lics2013-tr.pdf`
18. Fu, J., Topcu, U.: Computational methods for stochastic control with metric interval temporal logic specifications. In: CDC'15. pp. 7440–7447. IEEE (2015)
19. Gastin, P., Oddoux, D.: Fast LTL to Büchi automata translation. In: CAV'01. LNCS, vol. 2102, pp. 53–65. Springer (2001)
20. Gerth, R., Peled, D., Vardi, M.Y., Wolper, P.: Simple on-the-fly automatic verification of linear temporal logic. In: PSTV'95. pp. 3–18. Chapman & Hall (1995)

21. Hammer, M., Knapp, A., Merz, S.: Truly on-the-fly LTL model checking. In: TACAS'05. LNCS, vol. 3440, pp. 191–205. Springer (2005)
22. Hirshfeld, Y., Rabinovich, A.M.: Logics for real time: Decidability and complexity. Fundamenta Informaticae 62(1), 1–28 (2004)
23. Kant, G., Laarman, A., Meijer, J., van de Pol, J., Blom, S., van Dijk, T.: LTSmin: High-performance language-independent model checking. In: TACAS'15. LNCS, vol. 9035, pp. 692–707. Springer (2015)
24. Karaman, S.: Optimal Planning with Temporal Logic Specifications. Master's thesis, Massachussetts Institute of Technology (2009)
25. Kupferman, O., Vardi, M.Y.: Weak alternating automata are not that weak. In: ISTCS'97. pp. 147–158. IEEE (1997)
26. Larsen, K.G., Pettersson, P., Yi, W.: Uppaal in a nutshell. International Journal on Software Tools for Technology Transfer 1(1-2), 134–152 (1997)
27. Maler, O., Nickovic, D., Pnueli, A.: From MITL to timed automata. In: FORMATS'06. LNCS, vol. 4202, pp. 274–289. Springer (2006)
28. Muller, D.E., Saoudi, A., Schupp, P.E.: Alternating automata, the weak monadic theory of the tree, and its complexity. In: ICALP'86. LNCS, vol. 226, pp. 275–283. Springer (1986)
29. Ouaknine, J., Worrell, J.: On the decidability and complexity of metric temporal logic over finite words. Logical Methods in Computer Science 3(1) (2007)
30. Plaku, E., Karaman, S.: Motion planning with temporal-logic specifications: Progress and challenges. AI Communications 29, 151–162 (2016)
31. Pnueli, A.: The temporal logic of programs. In: FOCS'77. pp. 46–57. IEEE (1977)
32. Rozier, K.Y., Vardi, M.Y.: A multi-encoding approach for LTL symbolic satisfiability checking. In: FM'11. LNCS, vol. 6664, pp. 417–431. Springer (2011)
33. Thierry-Mieg, Y.: Symbolic model-checking using ITS-tools. In: TACAS'15. LNCS, vol. 9035, pp. 231–237. Springer (2015)
34. Vardi, M.Y.: An automata-theoretic approach to linear temporal logic. In: Logics for Concurrency, LNCS, vol. 1043, pp. 238–266. Springer (1996)
35. Wilke, T.: Specifying timed state sequences in powerful decidable logics and timed automata. In: FTRTFT'94. LNCS, vol. 863, pp. 694–715. Springer (1994)
36. Zhou, Y., Maity, D., Baras, J.S.: Timed automata approach for motion planning using metric interval temporal logic. Research Report 1603.08246, arXiv (2016)

We start this appendix with a lemma that we will need afterwards, and that helps understanding better the intricate definition of $\widehat{\psi}$. Indeed, consider subformula $\psi = \psi_1 \mathbf{U}_I \psi_2$ of φ. Then, $x_{\mathcal{P}_{\psi_i}}.\delta(\psi_i, \sigma)$ appears in constraints of transitions of the OCATA $\mathcal{A}_\varphi$. Then, we would like to characterise the minimal models of $x_{\mathcal{P}_{\psi_i}}.\delta(\psi_i, \sigma)$ with respect to any valuation v (the valuation does not really matter since $\delta(\psi_i, \sigma)$ is prefixed by the reset $x_{\mathcal{P}_{\psi_i}}$). This is exactly what permits to do $\widehat{\psi}$ and $\mathsf{trig}(., \delta(\psi, \sigma), v)$.

Lemma 8. *Let $\psi = \psi_1 \mathbf{U}_I \psi_2$ be a subformula of φ, $\sigma \in \Sigma$, and $i \in \{1, 2\}$.*

1. *If M is a minimal model of $x_{\mathcal{P}_{\psi_i}}.\delta(\psi_i, \sigma)$ with respect to valuation $v \in \mathbb{R}^+$, then $\sigma \cup \mathsf{trig}(M, \delta(\psi, \sigma), v) \models \widehat{\psi_i}$.*
2. *Reciprocally, for all $\sigma' \subseteq \mathcal{P}_{\psi_i}$, if $\sigma \cup \sigma' \models \widehat{\psi_i}$, then $\sigma' = \mathsf{trig}(M, \delta(\psi, \sigma), v) \cap \mathcal{P}_{\psi_i}$ for a minimal model M of $x_{\mathcal{P}_{\psi_i}}.\delta(\psi_i, \sigma)$ with respect to any valuation $v \in \mathbb{R}^+$.*

A The case of LTL formulae: proof of Proposition 5

To prove Proposition 5, we rely on Theorem 2 showing that $[\![\varphi]\!] = [\![\mathcal{A}_\varphi]\!]$. Therefore, we must relate the synchronous product of all component Büchi automata $\mathcal{C} = \mathcal{C}_{init} \times \prod_{\chi \in \Phi} \mathcal{C}_\chi$, with the very weak alternating automaton $\mathcal{A}_\varphi$. To simplify our notations, we forget about the time and clock valuations all along this proof. We use the new definition of $\mathcal{A}_\varphi$ given in 11 that labels clock resets with the set of triggers it pulls.

Consider first a word $\rho \in [\![\mathcal{A}_\varphi]\!]$ and an accepting run $G = (V, \to)$ of $\mathcal{A}_\varphi$ over ρ. We also let $K_0, K_1, \ldots$ be the sequence of configurations associated with the run G.

First, we construct a new word ρ' over $2^{\mathsf{AP} \cup \mathsf{AP}_\varphi}$ from ρ and G, by adding the triggers in AP_φ according to the minimal models selected in the run G. More precisely, let $\rho = \sigma_1 \sigma_2 \ldots$. For all $i \geq 0$, we associate every state ℓ (once again, we forget about time, so that states are simply locations) of K_i with the pair (γ_ℓ, M_ℓ) of transition formula and minimal model chosen in the run G when reading σ_{i+1}. We then gather all the triggers in a set $\mathcal{Q}_i = \bigcup_{\ell \in K_i} \mathsf{trig}(M_\ell, \gamma_\ell)$, and let $\rho' = (\sigma_1 \cup \mathcal{Q}_1)(\sigma_2 \cup \mathcal{Q}_2) \ldots$ be the Hintikka sequence we will read in $\mathcal{C}$.

Second, let us map the sequence $K_0, K_1, \ldots$ of configurations in the OCATA, to a sequence $C_0, C_1, \ldots$ of configurations in $\mathcal{C}$. For all $i \geq 0$, we let $C_i = (b_{init}, (b_\chi)_{\chi \in \Phi})$ with $b_{init} \in \{0, 1\}$ being the location of component $\mathcal{C}_{init}$, and $b_\chi \in \{0, 1\}$ the location of component $\mathcal{C}_\chi$, for all $\chi \in \Phi$, defined by: $b_{init} = 1$ if and only if $i = 0$ (remember that $K_0 = \{(\varphi_{init}, 0)\}$, and that location φ_{init} of $\mathcal{A}_\varphi$ is no longer reached afterwards), and $b_\chi = 1$ if and only if $i > 0$ and χ is a location appearing in K_i.

Then, we show that $C_0, C_1, \ldots$ is an accepting run of $\mathcal{C}$, by showing that the projection of the configurations on each component is an accepting run of the component Büchi timed automaton (the generalised Büchi acceptance condition on $\mathcal{C}$ is fulfilled exactly when each Büchi acceptance condition of the components is fulfilled). The proof for the component $\mathcal{C}_{init}$ is easy. Let $\mathcal{C}_\chi$ be another component with $\chi = \varphi_1 \mathbf{U} \varphi_2$ (the case $\chi = \varphi_1 \mathbf{R} \varphi_2$ is very similar, thus not presented here). For $i \geq 0$, let b_χ (respectively, b'_χ) be the location of $\mathcal{C}_\chi$ in C_i (respectively, C_{i+1}). Let us prove that there is a transition linking b_χ and b'_χ reading letter σ'_{i+1}. There are two cases:

- If $b_\chi = 1$, then χ is a location of K_i (associated with clock valuation v). Therefore, there exists a minimal model M of $\delta(\chi, \sigma_{i+1})$. Then (looking at the shape of $\delta(\chi, \sigma_{i+1})$): either (i) M is a minimal model of $x_{\mathcal{P}_{\varphi_2}}.\delta(\varphi_2, \sigma_{i+1})$, $\mathcal{P}_{\varphi_1} \cap \mathcal{Q}_i = \emptyset$, and $\chi \notin K_{i+1}$ or (ii) M is not a model of $x_{\mathcal{P}_{\varphi_2}}.\delta(\varphi_2, \sigma_{i+1})$, $\mathcal{P}_{\varphi_2} \cap \mathcal{Q}_i = \emptyset$, M is a minimal model of $x_{\mathcal{P}_{\varphi_1}}.\delta(\varphi_1, \sigma_{i+1})$, and $\chi \in K_{i+1}$. In case (i), formula $\widehat{\varphi_2}$ holds over σ'_{i+1} (by Lemma 8), as well as $*\varphi_1$, and transition from 1 to 0 labelled with $*\varphi_1 \wedge \widehat{\varphi}_2$ can be fired, while χ is not in K_{i+1}. In case (ii), formula $\widehat{\varphi_1}$ holds over σ'_{i+1} (again by Lemma 8), as well as $\neg\widehat{\varphi_2}$ and $*\varphi_2$, so that transition from 1 to 1 labelled with $\widehat{\varphi}_1 \wedge {\sim}\varphi_2$ can be fired, while χ is in K_{i+1}.
- If $b_\chi = 0$, then χ is not a location of K_i. If $p_\chi \notin \mathcal{Q}_i$, no obligations of φ_1 or φ_2 can be triggered as well, so that the transition from 0 to 0 labelled with $\neg p_\chi \wedge *\varphi_1 \wedge *\varphi_2$ can be fired (whatever χ is an $\mathbf{U}$ or a $\mathbf{R}$ formula). In the contrary, if $\chi \in \mathcal{Q}_i$, it has been triggered in the minimal model of another state of K_i, so that the model M must fulfil $\delta(\chi, \sigma_{i+1})$. Then the same cases (i) and (ii) hold, as before. In case (i), transition from 0 to 0 labelled with $p_\chi \wedge *\varphi_1 \wedge \widehat{\varphi}_2$ can be fired, while χ is not in K_{i+1}. In case (ii), transition from 0 to 1 labelled with $p_\chi \wedge \widehat{\varphi}_1 \wedge {\sim}\varphi_2$ can be fired, while χ is in K_{i+1}.

Hence, we know that $C_0, C_1, \dots$ is a run of $\mathcal{C}$ over ρ'. Suppose that it is not accepting. Then, there exists one subformula of the form $\chi = \varphi_1 \mathbf{U} \varphi_2$ (the only component with a non-accepting location) that stays in location 1 forever, after some point. Consider the step when component $\mathcal{C}_\chi$ jumps in location 1 for the last time. Then, trigger p_χ holds, and φ_2 never holds anymore in the following. In the OCATA, this means that χ is a location in the according level in the DAG, and in all its successor levels (since φ_2 never holds): this exactly means that this branch is not accepting, which contradicts the fact that G is an accepting run.

Reciprocally, consider a word $\rho' \in [\![\mathcal{C}]\!]$. Let us prove that $\rho = \mathsf{proj}_{\mathsf{AP}}(\rho') \in [\![\mathcal{A}_\varphi]\!]$. For that, we take an accepting run of $\mathcal{C}$ over ρ', that is a sequence of configurations $C_0, C_1, \dots$, and show how to build a DAG $G = (V, \to)$ that is an accepting run of $\mathcal{A}_\varphi$ over ρ. Moreover, it will have the property that the sequence of configurations $K_0, K_1, \dots$ associated with G is such that for all $i \geq 0$, the configuration C_i of $\mathcal{C}$ satisfies: for all $\chi \in \Phi$, C_i is in state 1 of $\mathcal{C}_\chi$ if and only if location χ appears in K_i. The DAG is built level after level. First, the root of G is the state $(\varphi_{init}, 0)$, as expected. Then, supposing that level i of G has been constructed, with K_i verifying the previous property, we construct level $i+1$. For each vertex (χ, i) in the DAG, we know that component $\mathcal{C}_\chi$ is in state 1. Consider the case where $\chi = \varphi_1 \mathbf{U} \varphi_2$ (the case $\chi = \varphi_1 \mathbf{R} \varphi_2$ is solved similarly).

- Suppose that component $\mathcal{C}_\chi$ fires the transition from 1 to 1, labelled by $\widehat{\varphi_1} \wedge {\sim}\varphi_2$. Since $\sigma'_{i+1} \models \widehat{\varphi_1}$, by Lemma 8, there exists a minimal model

M of $x_{\mathcal{P}_{\varphi_1}}.\delta(\varphi_1, \sigma_{i+1})$ such that $\sigma'_{i+1} \cap \mathsf{AP}_{\varphi_1} = \mathsf{trig}(M, \delta(\varphi_1, \sigma_{i+1})) \cap \mathsf{AP}_{\varphi_1}$. Therefore, $M \cup \{\chi\}$ is a minimal model of $x_{\mathcal{P}_{\varphi_1}}.\delta(\varphi_1, \sigma_{i+1}) \wedge \chi$. Moreover, σ'_{i+1} satisfies ${\sim}\varphi_2 = \neg\overline{\varphi_2} \wedge {*}\varphi_2$, so that $\sigma'_{i+1} \cap \mathcal{P}_{\varphi_2} = \emptyset$, and, thus, $\sigma_{i+1} \models \neg\overline{\varphi_2}$: this implies that $\delta(\varphi_2, \sigma_{i+1})$ admits no models. Therefore, $M \cup \{\chi\}$ is also a minimal model of $\delta(\chi, \sigma_{i+1})$. For each $\xi \in M \cup \{\chi\}$, we add a state $(\xi, i+1)$ in the DAG, with (χ, i) as a parent. Notice that $\chi \in C_{i+1}$, since we fired transition from 1 to 1 in $\mathcal{C}_\chi$. Moreover, consider $\xi \in M$: $x_{\mathcal{P}_\xi}.\delta(\xi, \sigma_{i+1})$ is thus a subformula of $\delta(\varphi_1, \sigma_{i+1})$ that must be fulfilled by M, so that $p_\xi \in \mathsf{trig}(M, \delta(\varphi_1, \sigma_{i+1}))$. Moreover, letting $\xi = \xi_1 \mathbf{U} \xi_2$ (a similar reasoning holds if $\xi = \xi_1 \mathbf{R} \xi_2$), M is not a model of ξ_2 (otherwise, M would not be minimal). Therefore, in component $\mathcal{C}_\xi$, a transition labelled by a formula consistent with $p_\xi \wedge {\sim}\xi_2$ has to be fired: this is either the transition from 0 to 1, or from 1 to 1. Thus, component $\mathcal{C}_\xi$ is in state 1 in configuration C_{i+1}, which is consistent with $(\xi, i+1)$ being in K_{i+1}.

– Suppose then that component $\mathcal{C}_\chi$ fires the transition from 1 to 0, labelled by ${*}\varphi_1 \wedge \widehat{\varphi_2}$. Since $\sigma'_{i+1} \models \widehat{\varphi_2}$, by Lemma 8, there exists a minimal model M of $x_{\mathcal{P}_{\varphi_2}}.\delta(\varphi_2, \sigma_{i+1})$ such that $\sigma'_{i+1} \cap \mathsf{AP}_{\varphi_2} = \mathsf{trig}(M, \delta(\varphi_2, \sigma_{i+1})) \cap \mathsf{AP}_{\varphi_2}$. Moreover, $\sigma'_{i+1} \models {*}\varphi_1$, so that $\sigma'_{i+1} \cap \mathcal{P}_{\varphi_1} = \emptyset$. Therefore, M is also a minimal model of $\delta(\chi, \sigma_{i+1})$. As before, for each $\xi \in M$, we add a node $(\xi, i+1)$ in the DAG, with (χ, i) as a parent. First, χ is not in C_{i+1}, and χ not in K_{i+1}. Then, the same reasoning as before allows us to prove that, for each $\xi \in M$, component $\mathcal{C}_\xi$ is in state 1 in configuration C_{i+1}, which is consistent with ξ being in K_{i+1}.

Finally, let us show that K_{i+1} is exactly the set of locations χ such that component $\mathcal{C}_\chi$ is in location 1 in configuration C_{i+1}. By contradiction, suppose that component $\mathcal{C}_\chi$ is in 1, and $\chi \notin K_{i+1}$: by the previous case, this means that $\mathcal{C}_\chi$ has followed the transition from 0 to 1. Let us consider the case $\chi = \varphi_1 \mathbf{U} \varphi_2$. Then, $\sigma'_{i+1} \models p_\chi \wedge \widehat{\varphi_1} \wedge {\sim}\varphi_2$. This means that χ has been triggered by the smallest subformula of φ in which it appears: by the previous case, we have seen that χ would then appear in K_{i+1}. Consider then the symmetrical case where $\mathcal{C}_\chi$ is in state 0 in configuration C_{i+1}, but $\chi \in K_{i+1}$. We can suppose that $\chi \notin K_i$, since the previous construction takes care of this case. Thus, component $\mathcal{C}_\chi$ has followed transition from 0 to 0 while reading σ'_{i+1}. However, by the previous construction of level $i+1$ of the DAG, χ appears in a minimal model of a transition $\delta(\psi, \sigma_{i+1})$: this implies that $p_\chi \in \sigma'_{i+1}$, but this contradicts the fact that component $\mathcal{C}_\chi$ followed transition from 0 to 0 (we have seen that this component arrives necessarily in state 1).

Hence, we have built a run G over ρ. Suppose that it is not accepting. Then we know that there is a branch where, after some level i, no more accepting location is visited, i.e. only non-accepting locations are visited. Notice from the construction that it implies that some subformula $\varphi_1 \mathbf{U} \varphi_2$ appears in every location of this branch below level i. This implies that the run of component $\mathcal{C}_\chi$ is not accepting, which contradicts the generalised Büchi condition of $\mathcal{C}$. Therefore, G is accepting, so that $\mathsf{proj}_{\mathsf{AP}}(\rho') \in [\![\mathcal{A}_\varphi]\!]$.

B The case of $\mathsf{MITL}_{\mathbf{0},\infty}$

B.1 Proof of correctness of the construction of a run of $\mathcal{C}_\chi$ from a run of $\mathcal{A}_\varphi$

We show that $C_0, C_1, \ldots$ is a run of $\mathcal{C}$, by showing that the projection of the configurations on each component is a run of the component Büchi timed automaton (the generalised Büchi acceptance condition on $\mathcal{C}$ is fulfilled exactly when each Büchi acceptance condition of the components is fulfilled). The proof for the component $\mathcal{C}_{init}$ is the same as before. Let $\mathcal{C}_\chi$ be another component with $\chi = \varphi_1 \mathbf{U}_I \varphi_2$ (the case $\chi = \varphi_1 \mathbf{R}_I \varphi_2$ is very similar, thus not presented here). We focus first on the case $I = [0, a]$. For $i \geq 0$, let (b_χ, v_χ) (respectively, (b'_χ, v'_χ)) be the configuration of $\mathcal{C}_\chi$ in C_i (respectively, C_{i+1}). Let us prove that there is a transition linking them while reading letter σ'_{i+1} after a delay $\tau_{i+1} - \tau_i$. There are two cases:

- If $b_\chi = 1$, then χ is a location of K_i, associated with a certain clock valuation: v_χ is the greatest one associated with location χ. Therefore, there exists a minimal model M of $\delta(\chi, \sigma_{i+1})$ with respect to $v_\chi + \tau_{i+1} - \tau_i$. Then (looking at the shape of $\delta(\chi, \sigma_{i+1})$): either (i) M is a minimal model of $x_{\mathcal{P}_{\varphi_2}}.\delta(\varphi_2, \sigma_{i+1})$ with respect to $v_\chi + \tau_{i+1} - \tau_i$, $v_\chi + \tau_{i+1} - \tau_i \leq a$, $\mathcal{P}_{\varphi_1} \cap \mathcal{Q}_i = \emptyset$, and χ is no longer a location of K_{i+1} (indeed, as already explained before, the witness of satisfaction of this occurrence of χ is also a witness for all other occurrences) or (ii) M is not a model of $x_{\mathcal{P}_{\varphi_2}}.\delta(\varphi_2, \sigma_{i+1})$ with respect to $v_\chi + \tau_{i+1} - \tau_i$, $\mathcal{P}_{\varphi_2} \cap \mathcal{Q}_i = \emptyset$, M is a minimal model of $x_{\mathcal{P}_{\varphi_1}}.\delta(\varphi_1, \sigma_{i+1})$ with respect to $v_\chi + \tau_{i+1} - \tau_i$, $v_\chi + \tau_{i+1} - \tau_i \leq a$, and $(\chi, v_\chi + \tau_{i+1} - \tau_i) \in K_{i+1}$ (moreover, $v_\chi + \tau_{i+1} - \tau_i$ is the greatest valuation associated with χ in K_{i+1}). In case (i), formula $\widehat{\varphi_2}$ holds over σ'_{i+1} (by Lemma 8), as well as $*\varphi_1$, and transition from 1 to 0 labelled with $*\varphi_1 \wedge \widehat{\varphi}_2 \wedge x \leq a$ can be fired, while χ does not appear anymore in K_{i+1} (we reset the clock x on this transition, which is purely optional). In case (ii), formula $\widehat{\varphi_1}$ holds over σ'_{i+1} (again by Lemma 8), as well as $\neg\overline{\varphi_2}$ and $*\varphi_2$, so that transition from 1 to 1 labelled with $\widehat{\varphi}_1 \wedge {\sim}\varphi_2 \wedge x \leq a$ can be fired (here the clock x cannot be reset), while χ still appears in K_{i+1}.
- If $b_\chi = 0$, then χ is not a location of K_i. If $p_\chi \notin \mathcal{Q}_i$, no obligations of φ_1 or φ_2 can be triggered as well, so that the transition from 0 to 0 labelled with $\neg p_\chi \wedge *\varphi_1 \wedge *\varphi_2$ can be fired (whatever χ is an $\mathbf{U}$ or a $\mathbf{R}$ formula). In the contrary, if $\chi \in \mathcal{Q}_i$, it has been triggered in the minimal model of another state of K_i, so that the model M must fulfil $\delta(\chi, \sigma_{i+1})$ with respect to $v_\chi + \tau_{i+1} - \tau_i$. Then the same cases (i) and (ii) hold, as before. In case (i), transition from 0 to 0 labelled with $p_\chi \wedge *\varphi_1 \wedge \widehat{\varphi}_2$ can be fired, while χ is not in K_{i+1}. In case (ii), transition from 0 to 1 labelled with $p_\chi \wedge \widehat{\varphi}_1 \wedge {\sim}\varphi_2$ can be fired, while $(\chi, 0)$ is in K_{i+1} (this explains why clock x must be reset on this transition).

If the interval is $I = [a, +\infty)$, exactly the same reasoning allows us to check the validity of the guards and resets of transitions in the component: the main difference is that, when in location 1 with formula φ_2 being fulfilled, we jump in locations $1'$ or 0, depending on whether a new trigger p_χ is launched or not.

We then prove that the run $C_0, C_1, \dots$ is accepting. Suppose that it is not accepting. Then, there exists a subformula of the form $\chi = \varphi_1 \mathbf{U}_I \varphi_2$ (the only components with a non-accepting location) that stays in location 1 forever, after some point. Suppose first that $I = [0, a]$. Consider the step when component $\mathcal{C}_\chi$ jumps in location 1 for the last time. Then, trigger p_χ holds, and φ_2 never holds anymore in the following. In the OCATA, this means that χ is a location of the according level in the DAG G, and all its successor levels (since φ_2 never holds): this exactly means that this branch is not accepting, which contradicts the fact that G is an accepting run. If $I = [a, +\infty)$, consider also the last time component $\mathcal{C}_\chi$ jumps in location 1. At that time, either p_χ holds or, we were already in location $1'$, and the previous obligation for χ has not been fulfilled during this transition. We are then stuck in location 1 which means that, at every step, either φ_2 does not hold, or φ_2 holds but before entering in the interval $[a, +\infty)$, or a new trigger for χ is launched: as before, this allows us to create an infinite branch of the DAG G that is in location χ for ever, after some time. Once again, this contradicts the fact that G is an accepting run.

B.2 Construction of an accepting run of $\mathcal{A}_\varphi$ from an accepting run of $\mathcal{C}$

Consider a timed word $\rho' \in [\![\mathcal{C}]\!]$. Let us prove that $\rho = \mathsf{proj}_{\mathsf{AP}}(\rho') \in [\![\mathcal{A}_\varphi]\!]$. For that, we take an accepting run of $\mathcal{C}$ over ρ', that is a sequence of configurations $C_0, C_1, \dots$, and show how to build a DAG $G = (V, \rightarrow)$ that is an accepting run of $\mathcal{A}_\varphi$ over ρ. Moreover, it will have the property that the sequence of configurations $K_0, K_1, \dots$ associated with G is such that for all $i \geq 0$, the configuration C_i of $\mathcal{C}$ satisfies: for all $\chi \in \Phi$, C_i is in state $(1, v)$ (or $(1', v)$ in case of $\mathbf{U}_{[a,+\infty)}$) of $\mathcal{C}_\chi$ if and only if location χ appears in K_i, in which case v is the greatest/smallest (depending on the operator, as before) value associated with χ in K_i. The DAG is built level after level. First, the root of G is the state $(\varphi_{init}, 0, 0)$ as expected. Then, supposing that level i of G has been constructed, with K_i verifying the previous property, we construct level $i{+}1$. For each vertex (χ, v, i) in the DAG, we know that component $\mathcal{C}_\chi$ is in state 1 or $1'$. Consider the case where $\chi = \varphi_1 \mathbf{U}_I \varphi_2$ (the case $\chi = \varphi_1 \mathbf{R}_I \varphi_2$ is solved similarly). First, suppose that $I = [0, a]$.

– Suppose that component $\mathcal{C}_\chi$ fires the transition from 1 to 1, labelled by $\widehat{\varphi_1} \wedge \sim\varphi_2 \wedge x \leq a$. Since $\sigma'_{i+1} \models \widehat{\varphi_1}$, by Lemma 8, there exists a minimal model M of $x_{\mathcal{P}_{\varphi_1}}.\delta(\varphi_1, \sigma_{i+1})$ with respect to $v + \tau_{i+1} - \tau_i$ such that $\sigma'_{i+1} \cap \mathsf{AP}_{\varphi_1} = \mathsf{trig}(M, \delta(\varphi_1, \sigma_{i+1}), v + \tau_{i+1} - \tau_i) \cap \mathsf{AP}_{\varphi_1}$. Therefore, $M \cup \{\chi\}$ is a minimal model of $x_{\mathcal{P}_{\varphi_1}}.\delta(\varphi_1, \sigma_{i+1}) \wedge \chi$ with respect to $v + \tau_{i+1} - \tau_i$. Moreover, σ'_{i+1} satisfies $\sim\varphi_2 = \neg\overline{\varphi_2} \wedge *\varphi_2$, so that $\sigma'_{i+1} \cap \mathcal{P}_{\varphi_2} = \emptyset$, and, thus, $\sigma_{i+1} \models \neg\overline{\varphi_2}$: this implies that $\delta(\varphi_2, \sigma_{i+1})$ admits no models with respect to 0. Therefore, $M \cup \{\chi\}$ is also a minimal model of $\delta(\chi, \sigma_{i+1})$ with respect to $v + \tau_{i+1} - \tau_i$. First, we add a state $(\chi, v' {+} \tau_{i+1} {-} \tau_i, i{+}1)$ in G For each (χ, v', i) in G, child of every state $(\chi, v' + \tau_{i+1} - \tau_i, i + 1)$ of G. Notice that $(\chi, v + \tau_{i+1} - \tau_i) \in C_{i+1}$

(with $v \geq v'$), since we fired transition from 1 to 1 in $\mathcal{C}_\chi$. Then, for each $\xi \in M$, we add a state $(\xi, 0, i+1)$ in the G, with all (χ, v', i) of G as parents. Since $\xi \in M$, $x_{\mathcal{P}_\xi}.\delta(\xi, \sigma_{i+1})$ is a subformula of $\delta(\varphi_1, \sigma_{i+1})$ that must be fulfilled by M, so that $p_\xi \in \mathsf{trig}(M, \delta(\varphi_1, \sigma_{i+1}), 0)$. Moreover, letting $\xi = \xi_1 \mathbf{U}_I \xi_2$ (a similar reasoning holds if $\xi = \xi_1 \mathbf{R}_I \xi_2$), M is not a model of ξ_2 (otherwise, M would not be minimal). Therefore, in component $\mathcal{C}_\xi$, a transition labelled by a formula consistent with $p_\xi \wedge {\sim}\xi_2$ has to be fired: this is either the transition from 0 to 1, or from 1 to 1. Thus, component $\mathcal{C}_\xi$ is in state $(1,0)$ in configuration C_{i+1}, which is consistent with $(\xi, 0)$ being a state in K_{i+1}.

– Suppose then that component $\mathcal{C}_\chi$ fires the transition from 1 to 0, labelled by $*\varphi_1 \wedge \widehat{\varphi_2} \wedge x \leq a$. Since $\sigma'_{i+1} \models \widehat{\varphi_2}$, by Lemma 8, there exists a minimal model M of $x_{\mathcal{P}_{\varphi_2}}.\delta(\varphi_2, \sigma_{i+1})$ with respect to $v + \tau_{i+1} - \tau_i$ such that $\sigma'_{i+1} \cap \mathsf{AP}_{\varphi_2} = \mathsf{trig}(M, \delta(\varphi_2, \sigma_{i+1}), v + \tau_{i+1} - \tau_i) \cap \mathsf{AP}_{\varphi_2}$. Moreover, $\sigma'_{i+1} \models *\varphi_1$, so that $\sigma'_{i+1} \cap \mathcal{P}_{\varphi_1} = \emptyset$. Therefore, M is also a minimal model of $\delta(\chi, \sigma_{i+1})$ with respect to $v + \tau_{i+1} - \tau_i$. As before, for each $\xi \in M$ and (χ, v', i) in G, we add a state $(\xi, 0, i+1)$ in the DAG, with (χ, v', i) as parent. First, χ is not in C_{i+1}, and χ not in K_{i+1}. Then, the same reasoning as before allows us to prove that, for each $\xi \in M$, component $\mathcal{C}_\xi$ is in state $(1,0)$ in configuration C_{i+1}, which is consistent with $(\xi, 0)$ being in K_{i+1}.

In the case $I = [a, +\infty)$, we apply a similar reasoning, without considering locations 1 and 1′ differently here.

True?

Finally, let us show that K_{i+1} contains exactly the locations χ such that component $\mathcal{C}_\chi$ is in location 1 in configuration C_{i+1}. By contradiction, suppose first that component $\mathcal{C}_\chi$ is in location 1, and χ does not appear in K_{i+1}: by the previous case, this means that $\mathcal{C}_\chi$ has followed the transition from 0 to 1. Let us consider the case $\chi = \varphi_1 \mathbf{U} \varphi_2$. Then, $\sigma'_{i+1} \models p_\chi \wedge \widehat{\varphi_1} \wedge {\sim}\varphi_2$. This means that χ has been triggered by the smallest subformula of φ in which it appears: by the previous case, we have seen that $(\chi, 0, i+1)$ would then appear in K_{i+1}. Consider then the symmetrical case where $\mathcal{C}_\chi$ is in location 0 in configuration C_{i+1}, but χ is in K_{i+1}. We can suppose that χ is not in K_i, since the previous construction takes care of this case. Thus, component $\mathcal{C}_\chi$ has followed transition from 0 to 0 while reading σ'_{i+1}. However, by the previous construction of level $i+1$ of the DAG, χ appears in a minimal model of a transition $\delta(\psi, \sigma_{i+1})$: this implies that $p_\chi \in \sigma'_{i+1}$, but this contradicts the fact that component $\mathcal{C}_\chi$ followed transition from 0 to 0 (we have seen that this component arrives necessarily in state 1).

Hence, we have built a run G over ρ. Suppose that it is not accepting. Then we know that there is a branch where, after some level i, no more accepting location is visited, i.e. only non-accepting locations are visited. Notice from the construction that it implies that some subformula $\varphi_1 \mathbf{U}_I \varphi_2$ appears in every location of this branch below level i. This implies that the run of component $\mathcal{C}_\chi$ is not accepting, which contradicts the generalised Büchi condition of $\mathcal{C}$. Therefore, G is accepting, so that $\mathsf{proj}_{\mathsf{AP}}(\rho') \in [\![\mathcal{A}_\varphi]\!]$.