Context-free timed formalisms: Robust automata and linear temporal logics [☆]

Laura Bozzelli, Aniello Murano, Adriano Peron [*]

Department of Elect. Engineering and Information Technologies, University of Napoli "Federico II", Italy

Keywords:
Timed automata
Temporal logic
Metric temporal logic
Model checking

A B S T R A C T

The paper focuses on automata and linear temporal logics for real-time pushdown reactive systems bridging tractable formalisms specialized for expressing separately dense-time real-time properties and context-free properties though preserving tractability. As for automata, we introduce Event-Clock Nested Automata (ECNA), a formalism that combines Event Clock Automata (ECA) and Visibly Pushdown Automata (VPA). ECNA enjoy the same closure and decidability properties of ECA and VPA expressively extending any previous attempt of combining ECA and VPA. As for temporal logics, we introduce two formalisms for specifying quantitative timing context-free requirements: *Event-Clock Nested Temporal Logic* (EC_NTL) and *Nested Metric Temporal Logic* (NMTL). EC_NTL is an extension of both the logic CaRet and *Event-Clock Temporal Logic* having EXPTIME-complete satisfiability of EC_NTL and visibly model-checking of Visibly Pushdown Timed Systems (VPTS) against EC_NTL. NMTL is a context-free extension of standard Metric Temporal Logic (MTL) which is in general undecidable having, though, a fragment expressively equivalent to EC_NTL with EXPTIME-complete satisfiability and visibly model-checking of VPTS problems.

1. Introduction

Model checking is a well-established formal-method technique to automatically check for global correctness of reactive systems [10,24,32]: the (potentially infinite) dynamic behavior of a system is described by a mathematical model and checked against a behavioral constraint expressed by a suitable specification. In this setting, both automata theory over infinite words and temporal logics provide fundamental (related) frameworks for the description of the dynamic behavior of reactive systems. The focus of this paper is on the challenge of bridging formalisms specialized for expressing separately dense-time real-time properties and context-free properties though ensuring decidability and tractability in the combined setting.

In this framework, one should be able to specify real-time non-regular properties relevant for recursive real-time systems. As an example, properties like:

- Local bounded-time responses such as "in the local computation of a procedure A, every request p is followed by a response q within k time units".

☆ The work by Adriano Peron and Aniello Murano has been supported by the GNCS project *Formal methods for verification and synthesis of discrete and hybrid systems*.

* Corresponding author.
E-mail addresses: lr.bozzelli@gmail.com (L. Bozzelli), aniello.murano@unina.it (A. Murano), adrperon@unina.it (A. Peron).

- Bounded-time total correctness requirements such as "if the pre-condition p holds when the procedure A is invoked, then the procedure must return within k time units and q must hold upon return".
- Real-time security properties which require the inspection of the call-stack such as "a module A should be invoked only if module B belongs to the call stack and within k time units since the activation of module B".

In this paper we shall study the problem first in the context of automata theory and, then, we exploit the obtained results to investigate the problem in the related context of linear temporal logics.

The automata view. Automata theory over infinite words plays a crucial role in model checking: the set of possible (potentially infinite) behaviors of the system and the set of admissible behaviors of the correctness specification can be modeled as languages accepted by automata. The verification problem of checking that a system meets its specification then reduces to testing language inclusion between two automata over infinite words.

In the last two decades, the analysis of infinite-state sequential systems through the model checking of pushdown automata (PDA) has received serious attention (e.g. see [28,36,17,18]). PDA represent an infinite-state formalism suitable to model the control flow of typical sequential programs with nested and recursive procedure calls. Although the general problem of checking context-free properties of PDA is undecidable [28], algorithmic solutions have been proposed for interesting subclasses of context-free requirements [3,7,8,23]. A well-known approach is that of *Visibly Pushdown Automata* (VPA) [7,8], a subclass of PDA where the input symbols over a *visibly pushdown alphabet* control the admissible operations on the stack. Precisely, the alphabet is partitioned into a set of *calls*, representing a procedure call and forcing a push stack-operation, a set of *returns*, representing a procedure return and forcing a pop stack-operation, and a set of *internal* actions that cannot access or modify the content of the stack. A call and the matching return (if any) along a given word denotes the exit from this procedure (i.e., a pop stack-operation). VPA push onto the stack only when a call is read, pop the stack only at returns, and do not use the stack on reading internal symbols. This restriction makes the class of resulting languages (*Visibly Pushdown Languages* or VPL) very similar in tractability and robustness to that of regular languages [7,8]. In particular, VPL are closed under Boolean operations, and language inclusion is Exptime-complete. VPA capture all regular properties, and, additionally, can specify regular requirements over two kinds of *non-regular* patterns on input words: *abstract paths* and *caller paths*. An abstract path captures the local computation within a procedure with the removal of subcomputations corresponding to nested procedure calls, while a caller path represents the call-stack content at a given position of the input. The benefits of exploiting a visibly pushdown alphabet has been investigated also for multistack automata (e.g. [21,22]).

Recently, many works [1,11,12,16,25,26,35] have investigated real-time extensions of PDA by combining PDA with *Timed Automata* (TA) [2], a model widely used to represent real-time systems. TA are finite automata augmented with a finite set of real-valued clocks, which operate over words where each symbol is paired with a real-valued timestamp (*timed words*). The clocks record the elapsed time among events, and while transitions are instantaneous, time can elapse in a control state. All clocks progress at same speed and can be reset by transitions (thus, each clock keeps track of the elapsed time since the last reset). Constraints on clocks are associated with transitions to restrict the behavior of the automaton. The emptiness problem for TA is decidable and Pspace complete [2]. However, since in TA, clocks can be reset non-deterministically and independently of each other, the resulting class of timed languages is not closed under complement and, in particular, language inclusion is undecidable [2]. As a consequence, the general verification problem (i.e., language inclusion) of formalisms combining unrestricted TA with robust subclasses of PDA such as VPA, i.e. *Visibly Pushdown Timed Automata* (VPTA), is undecidable as well: checking language inclusion for VPTA is undecidable even in the restricted case of specifications using at most one clock [26].

Event-clock automata (ECA) [5] are an interesting subclass of TA where the explicit reset of clocks is disallowed. In ECA, clocks have a predefined association with the input alphabet symbols. Precisely, for each symbol a, there are two clocks: the *global recorder clock*, recording the time elapsed since the last occurrence of a, and the *global predictor clock*, measuring the time required for the next occurrence of a. Hence, the clock valuations are determined only by the input timed word being independent of the automaton behavior. Such a restriction makes the resulting class of timed languages closed under Boolean operations, and in particular, language inclusion is Pspace-complete [5]. Note that common real-time regular requirements like bounded-response time can be naturally modeled by ECA. A robust subclass of VPTA, called *Event-Clock Visibly Pushdown Automata* (ECVPA), has been proposed in [34], combining ECA with VPA. ECVPA are closed under Boolean operations, and language inclusion is Exptime-complete. However, ECVPA do not take into account the nested hierarchical structure induced by a timed word over a pushdown alphabet, namely, they do not provide any explicit mechanism to relate the use of a stack with event clocks.

The idea of clocks used to measure procedures calls and returns is somehow related also to *interrupt* Timed Automata [14] (although the formalisms seems not to be immediately comparable). Moreover, non-regular (context-sensitive) timed languages can be accepted also by *parametric* TA with only 2 clocks (only one of which is compared to a parameter [9]) a formalism that enjoys some decidability results [13].

In this paper, we introduce an extension of ECVPA, called *Event-Clock Nested Automata* (ECNA) that, differently from ECVPA, allows us to relate the use of event clocks and the use of the stack. To this end, we add for each input symbol a, three additional event clocks: the *abstract recorder clock* (resp., *abstract predictor clock*), measuring the time elapsed since the last occurrence (resp., the time for the next occurrence) of a along the maximal abstract path visiting the current position; the *caller clock*, measuring the time elapsed since the last occurrence of a along the caller path from the current position. In this way, ECNA allow us to specify the relevant real-time non-regular properties listed above. In general, we show that ECNA are strictly more expressive than ECVPA and, as for ECVPA, the resulting class of languages is closed under all Boolean operations. Moreover, language inclusion and visibly model-checking of VPTA against ECNA specifications are decidable and EXPTIME-complete.

The temporal logic counterpart. Temporal logics provide a fundamental framework for the description of the dynamic behavior of reactive systems. In this paper, we introduce two real-time linear temporal logics, called *Event-Clock Nested Temporal Logic* (EC_NTL) and *Nested Metric Temporal Logic* (NMTL) for specifying quantitative timing context-free requirements in a pointwise semantics setting (models of formulas are timed words).

The proposed logic EC_NTL is an extension of both EC_TL and CaRet by means of non-regular versions of the timed modalities of EC_TL, which allows one to refer to abstract and caller paths. *Event-Clock Temporal Logic* (EC_TL) [33] is a tractable real-time logical framework related to ECA. EC_TL extends LTL + past with timed temporal modalities, which specify time constraints on the distances from the previous/next timestamp where a given subformula holds. Instead, CaRet [3] is a context-free extension of LTL having VPA as automata-theoretic generalization. CaRet formulas are interpreted on words over a *visibly pushdown alphabet* (as for VPA). CaRet allows the specification of LTL requirements over the *non-regular* patterns induced by the visibly pushdown alphabet: *abstract paths* and *caller paths*.

In this paper we address expressiveness and complexity issues for the logic EC_NTL showing that satisfiability of EC_NTL and visibly model-checking of VPTA against EC_NTL are decidable and EXPTIME-complete. The key step in the proposed decision procedures is a translation of EC_NTL into ECNA accepting suitable encodings of the models of the given formula. The second logic we introduce, called NMTL, is a context-free extension of standard Metric Temporal Logic (MTL). This extension is obtained by adding to MTL timed versions of the caller and abstract temporal modalities of CaRet. In the considered pointwise-semantics settings, it is well known that satisfiability of future MTL is undecidable when it is interpreted over infinite timed words [30], but it is decidable over finite timed words [31]. We show that adding the future abstract timed modalities to future MTL makes the satisfiability problem undecidable also over finite timed words. On the other hand, we show that the fragment $\mathsf{NMITL}_{(0,\infty)}$ of NMTL (the NMTL counterpart of the well-known tractable fragment $\mathsf{MITL}_{(0,\infty)}$ [4] of MTL) has the same expressiveness as the logic EC_NTL and the related satisfiability and visibly model-checking problems are, consequently, EXPTIME-complete.

The paper is structured as follows. In Section 2 we introduce some preliminary notation. In Section 3 we introduce ECNA proving their closure properties under Boolean operations and their expressiveness properties. In Section 4 we present the decidability and complexity results (EXPTIME-completeness) of the decision problems (satisfiability and visibly model checking) for ECNA. In Section 5 we introduce the logic EC_NTL and we prove that its satisfiability and visibly model checking are EXPTIME-complete. In Section 6 we introduce the logic NMTL and we prove that future NMTL over finite timed words is undecidable and that the fragment $\mathsf{NMITL}_{(0,\infty)}$ of NMTL has the same expressiveness as EC_NTL.

The paper combines and extends with complete proofs the content of the conference papers [20] and [19].

2. Preliminaries

In the following, $\mathbb{N}$ denotes the set of natural numbers and $\mathbb{R}_+$ the set of non-negative real numbers. Let w be a finite or infinite word over some alphabet. By $|w|$ we denote the length of w (we set $|w| = \infty$ if w is infinite). For all $i, j \in \mathbb{N}$, with $i \leq j$, w_i is i-th letter of w, while $w[i, j]$ is the finite subword $w_i \cdots w_j$.

An *infinite timed word* w over a finite alphabet Σ is an infinite word $w = (a_0, \tau_0)(a_1, \tau_1), \ldots$ over $\Sigma \times \mathbb{R}_+$ (intuitively, τ_i is the time at which a_i occurs) such that the sequence $\tau = \tau_0, \tau_1, \ldots$ of timestamps satisfies: (1) $\tau_i \leq \tau_{i+1}$ for all $i \geq 0$ (monotonicity), and (2) for all $t \in \mathbb{R}_+$ there is some $i \geq 0$, such that $\tau_i \geq t$ (divergence). The timed word w is also denoted by the pair (σ, τ), where σ is the untimed word $a_0 a_1 \ldots$ and τ is the sequence of timestamps. An *ω-timed language* over Σ is a set of infinite timed words over Σ.

Pushdown alphabets, abstract paths, and caller paths A *pushdown alphabet* is a finite alphabet $\Sigma = \Sigma_{call} \cup \Sigma_{ret} \cup \Sigma_{int}$ which is partitioned into a set Σ_{call} of *calls*, a set Σ_{ret} of *returns*, and a set Σ_{int} of *internal actions*. The pushdown alphabet Σ induces a nested hierarchical structure in a given word over Σ obtained by associating to each call the corresponding matching return (if any) in a well-nested manner. Formally, the set of *well-matched words* is the set of finite words σ_w over Σ inductively defined by the following grammar:

$$\sigma_w := \varepsilon \;\big|\; a \cdot \sigma_w \;\big|\; c \cdot \sigma_w \cdot r \cdot \sigma_w$$

where ε is the empty word, $a \in \Sigma_{int}$, $c \in \Sigma_{call}$, and $r \in \Sigma_{ret}$.

Let us fix an infinite word σ over Σ. For a call position $i \geq 0$, if there is $j > i$ such that j is a return position of σ and $\sigma[i+1, j-1]$ is a well-matched word (note that j is uniquely determined if it exists), we say that j is the *matching return* of i along σ. For a position $i \geq 0$, the *abstract successor of i along σ*, denoted $\mathsf{succ}(\mathsf{a}, \sigma, i)$, is defined as follows:

- If i is a call, then $\mathsf{succ}(\mathsf{a}, \sigma, i)$ is the matching return of i, if such a matching return exists; otherwise, $\mathsf{succ}(\mathsf{a}, \sigma, i) = \vdash$ ($\vdash$ denotes the *undefined* value).
- If i is not a call, then $\mathsf{succ}(\mathsf{a}, \sigma, i) = i + 1$ if $i + 1$ is not a return position, and $\mathsf{succ}(\mathsf{a}, \sigma, i) = \vdash$, otherwise.

The *caller of i along σ*, denoted $\mathsf{succ}(\mathsf{c}, \sigma, i)$, is instead defined as follows:

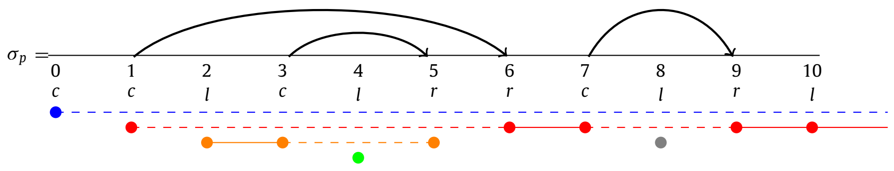


Fig. 1. An untimed word over a pushdown alphabet.

- if there exists the greatest call position $j_c < i$ such that either $\mathrm{succ}(\mathrm{a}, \sigma, j_c) = \vdash$ or $\mathrm{succ}(\mathrm{a}, \sigma, j_c) > i$, then $\mathrm{succ}(\mathrm{c}, \sigma, i) = j_c$; otherwise, $\mathrm{succ}(\mathrm{c}, \sigma, i) = \vdash$.

In the analysis of recursive programs, a *maximal abstract path* captures the local computation within a procedure removing computation fragments corresponding to nested calls, while the *caller path* represents the call-stack content at a given position of the input. Formally, a *maximal abstract path* (*MAP*) of σ is a *maximal* (finite or infinite) increasing sequence of natural numbers $\nu = i_0 < i_1 < \ldots$ such that $i_j = \mathrm{succ}(\mathrm{a}, \sigma, i_{j-1})$ for all $1 \leq j < |\nu|$. Note that for every position i of σ, there is exactly one *MAP* of σ visiting position i. For each $i \geq 0$, the *caller path of σ from position i* is the maximal (finite) decreasing sequence of natural numbers $j_0 > j_1 \ldots > j_n$ such that $j_0 = i$ and $j_{h+1} = \mathrm{succ}(\mathrm{c}, \sigma, j_h)$ for all $0 \leq h < n$. The abstract path ν is *maximal* (both in the past and in the future) if there is no abstract path of σ having ν as a proper subsequence. Note that the positions of a *MAP* have the same caller (if any). Let $w = (\sigma, \tau)$ be an infinite timed word over Σ and $i \geq 0$. We denote by $Pos(\mathrm{a}, w, i)$ the set of positions visited by the *MAP* of σ associated with position i, and by $Pos(\mathrm{c}, w, i)$ the set of positions visited by the caller path of σ from position i. In order to use a uniform notation, we write $Pos(\mathrm{g}, w, i)$ to mean the full set $\mathbb{N}$ of positions.

Example 1. Let $w = (\sigma, \tau)$ be the timed word over the pushdown alphabet Σ with $\Sigma_{call} = \{c\}$, $\Sigma_{ret} = \{r\}$, and $\Sigma_{int} = \{l\}$ such that the untimed word σ is $\sigma_p \cdot l^\omega$ with σ_p the prefix of length 10 depicted in Fig. 1. Note that 0 is the unique unmatched call position of σ: hence, the *MAP* visiting 0 consists of position 0 (i.e. $Pos(\mathrm{a}, w, 0) = \{0\}$), since it corresponds to a call devoid of matching return ($succ(a, \sigma, 0) = \vdash$), and has no caller. The *MAP* visiting position 1 is the infinite sequence $1, 6, 7, 9, 10, 11, 12, 13 \ldots$, namely $Pos(\mathrm{a}, w, 1) = \{1, 6, 7\} \cup \{I : i \geq 9\}$ and $Pos(\mathrm{c}, w, 1) = \{1, 0\}$. Analogously, we have $Pos(\mathrm{a}, w, 2) = \{2, 3, 5\}$ and $Pos(\mathrm{c}, w, 2) = \{2, 1, 0\}$; $Pos(\mathrm{a}, w, 4) = \{4\}$ and $Pos(\mathrm{c}, w, 4) = \{4, 3, 1, 0\}$.

3. Event-clock nested automata

In this section, we define the formalism of *Event-Clock Nested Automata* (ECNA), which combines the use of event clocks with visible operations on the stack. To this end, we augment the standard set of event clocks [5] with a set of *abstract event clocks* and a set of *caller event clocks* whose values are determined by considering maximal abstract paths and caller paths of the given word, respectively.

In the following, we fix a pushdown alphabet $\Sigma = \Sigma_{call} \cup \Sigma_{ret} \cup \Sigma_{int}$. The set C_Σ of event clocks associated with Σ is given by $C_\Sigma := \bigcup_{b \in \Sigma} \{x_b^{\mathrm{g}}, y_b^{\mathrm{g}}, x_b^{\mathrm{a}}, y_b^{\mathrm{a}}, x_b^{\mathrm{c}}\}$. Thus, we associate with each symbol $b \in \Sigma$, five event clocks: the *global recorder clock* x_b^{g} (resp., the *global predictor clock* y_b^{g}) recording the time elapsed since the last occurrence of b, if any, (resp., the time required to the next occurrence of b if any); the *abstract recorder clock* x_b^{a} (resp., the *abstract predictor clock* y_b^{a}) recording the time elapsed since the last occurrence of b, if any, (resp. the time required to the next occurrence of b) along the *MAP* visiting the current position; and the *caller (recorder) clock* x_b^{c} recording the time elapsed since the last occurrence of b if any along the caller path from the current position.

Fixed the input word w, when the automaton reads the i-th position σ_i at time τ_i, the values of the clocks are uniquely determined as follows.

Definition 1 *(Input determinisitic clock valuations).* A *clock valuation* over C_Σ is a mapping $val : C_\Sigma \mapsto \mathbb{R}_+ \cup \{\vdash\}$, assigning to each event clock a value in $\mathbb{R}_+ \cup \{\vdash\}$ ($\vdash$ denotes the *undefined* value). Given an infinite timed word $w = (\sigma, \tau)$ over Σ and a position i, the *clock valuation val_i^w over C_Σ*, specifying the values of the event clocks at position i along w, is defined as follows for each $b \in \Sigma$, where $dir \in \{\mathrm{g}, \mathrm{a}, \mathrm{c}\}$ and $dir' \in \{\mathrm{g}, \mathrm{a}\}$:

$$val_i^w(x_b^{dir}) = \begin{cases} \tau_i - \tau_j & \text{if } \exists j < i : b = \sigma_j,\ j \in Pos(dir, w, i), \text{ and} \\ & \quad \forall k : (j < k < i \text{ and } k \in Pos(dir, w, i)) \Rightarrow b \neq \sigma_k \\ \vdash & \text{otherwise;} \end{cases}$$

$$val_i^w(y_b^{dir'}) = \begin{cases} \tau_j - \tau_i & \text{if } \exists j > i : b = \sigma_j,\ j \in Pos(dir', w, i), \text{ and} \\ & \quad \forall k : (i < k < j \text{ and } k \in Pos(dir', w, i)) \Rightarrow b \neq \sigma_k \\ \vdash & \text{otherwise.} \end{cases}$$

Thus, at position i of the given word, the value of the global recorder clock x_b is the time since the last occurrence of symbol b if such an occurrence exists and it is undefined otherwise. Symmetrically, the value of the global predictor clock

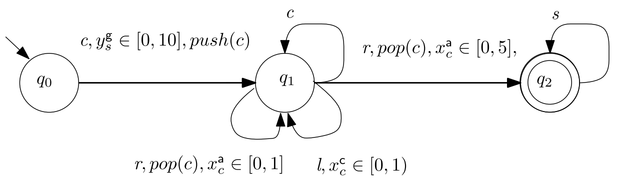

$c, y^{\mathsf{g}}_s \in [0,10], push(c)$
c
$r, pop(c), x^{\mathsf{a}}_c \in [0,5],$
s
q_0
q_1
q_2
$r, pop(c), x^{\mathsf{a}}_c \in [0,1]$
$l, x^{\mathsf{c}}_c \in [0,1)$

Fig. 2. An example of ECNA.

y_b is the time to wait for the next occurrence of b if such an occurrence exists and it is undefined otherwise. It is worth noting that while the values of the global clocks are obtained by considering the full set of positions in w, the values of the abstract clocks (resp., caller clocks) are defined with respect to the *MAP* visiting the current position (resp., with respect to the caller path from the current position). Notice that the values of the abstract recorder clock x^{a}_b and the abstract predictor clock y^{a}_b are not immediately defined in the timed word w as the global clocks but with respect to the *MAP* of w visiting the current valuation position.

For $C \subseteq C_\Sigma$ and a clock valuation *val* over C_Σ, $val_{|C}$ denotes the restriction of *val* to the set C. A *clock constraint* over C is a conjunction of atomic formulas of the form $z \in I$, where $z \in C$, and I is either an interval in $\mathbb{R}_+$ with bounds in $\mathbb{N} \cup \{\infty\}$, or the singleton $\{\vdash\}$ (also denoted by $[\vdash, \vdash]$). For a clock valuation *val* and a clock constraint θ, *val* satisfies θ, written $val \models \theta$, if for each conjunct $z \in I$ of θ, $val(z) \in I$. We denote by $\Phi(C)$ the set of clock constraints over C.

For technical convenience, we first introduce an extension of the known class of *Visibly Pushdown Timed Automata* (VPTA) [16,26], called *nested* VPTA. Nested VPTA are simply VPTA augmented with event clocks. Therefore, transitions of *nested* VPTA are constrained by a pair of disjoint finite sets of clocks: a finite set C_{st} of standard clocks and a disjoint set $C \subseteq C_\Sigma$ of event clocks. As usual, a standard clock can be reset when a transition is taken; hence, its value at a position of an input word depends in general on the behavior of the automaton and not only, as for event clocks, on the word.

The class of *Event-Clock Nested Automata* (ECNA) corresponds to the subclass of nested VPTA where the set of standard clocks C_{st} is empty. A (standard) clock valuation over C_{st} is a mapping $sval : C_{st} \mapsto \mathbb{R}_+$ (note that the undefined value $\vdash$ is not admitted). For $t \in \mathbb{R}_+$ and a reset set $Res \subseteq C_{st}$, $sval + t$ and $sval[Res]$ denote the valuations over C_{st} defined as follows for all $z \in C_{st}$: $(sval + t)(z) = sval(z) + t$, and $sval[Res](z) = 0$ if $z \in Res$ and $sval[Res](z) = sval(z)$ otherwise. For $C \subseteq C_\Sigma$ and a valuation *val* over C, $sval \cup val$ denotes the valuation over $C_{st} \cup C$ defined in the obvious way.

Definition 2 *(Nested VPTA).* A Büchi *nested* VPTA over $\Sigma = \Sigma_{call} \cup \Sigma_{int} \cup \Sigma_{ret}$ is a tuple $\mathcal{A} = (\Sigma, Q, Q_0, D = C \cup C_{st}, \Gamma \cup \{\top\}, \Delta, F)$, where Q is a finite set of (control) states, $Q_0 \subseteq Q$ is a set of initial states, $C \subseteq C_\Sigma$ is a set of event clocks, C_{st} is a set of standard clocks disjoint with C_Σ, $\Gamma \cup \{\top\}$ is a finite stack alphabet, $\top \notin \Gamma$ is the special *stack bottom symbol*, $F \subseteq Q$ is a set of accepting states, and $\Delta_c \cup \Delta_r \cup \Delta_i$ is a transition relation, where:

- $\Delta_c \subseteq Q \times \Sigma_{call} \times \Phi(D) \times 2^{C_{st}} \times Q \times \Gamma$ is the set of *push transitions*;
- $\Delta_r \subseteq Q \times \Sigma_{ret} \times \Phi(D) \times 2^{C_{st}} \times (\Gamma \cup \{\top\}) \times Q$ is the set of *pop transitions*;
- $\Delta_i \subseteq Q \times \Sigma_{int} \times \Phi(D) \times 2^{C_{st}} \times Q$ is the set of *internal transitions*.

We now describe how a nested VPTA $\mathcal{A}$ behaves over an infinite timed word w. Assume that on reading the i-th position of w, the current state of $\mathcal{A}$ is q, val^w_i is the event-clock valuation associated with w and i, *sval* is the current valuation of the standard clocks in C_{st}, and $t = \tau_i - \tau_{i-1}$ is the time elapsed from the last transition (where $\tau_{-1} = 0$). If $\mathcal{A}$ reads a call $c \in \Sigma_{call}$, it chooses a push transition of the form $(q, c, \theta, Res, q', \gamma) \in \Delta_c$ and pushes the symbol $\gamma \neq \top$ onto the stack. If $\mathcal{A}$ reads a return $r \in \Sigma_{ret}$, it chooses a pop transition of the form $(q, r, \theta, Res, \gamma, q') \in \Delta_r$ such that γ is the symbol on top of the stack, and pops γ from the stack (if $\gamma = \top$, then γ is read but not removed). Finally, on reading an internal action $a \in \Sigma_{int}$, $\mathcal{A}$ chooses an internal transition of the form $(q, a, \theta, Res, q') \in \Delta_i$, without any stack operation. In all the cases, the constraint θ of the chosen transition must be fulfilled by the valuation $(sval + t) \cup (val^w_i)_{|C}$, the control changes from q to q', and all the standard clocks in *Res* are reset (i.e., the valuation of the standard clocks is updated to $(sval + t)[Res]$).

Example 2. In Fig. 2 we depict an ECNA over the pushdown alphabet Σ with $\Sigma_{call} = \{c\}$, $\Sigma_{ret} = \{r\}$, and $\Sigma_{int} = \{l, s\}$ (the word of Fig. 1 paired with a sequence of timestamps could be an input for the automaton). The initial state is q_0 and the final state is q_2. The symbol c can be interpreted as a procedure call, r as a return from a call, l as an internal activity during the procedure execution and s an idling action. The transition from the initial state requires that the activity is idling within 10 units of time by exploiting the *global predictor clock* y^{g}_s. (Notice that the state q_2 can be reached with an non-empty stack i.e. with possibly unmatched procedure calls.) The return transition taking from q_1 to q_2 requires, by using the *abstract recorder* clock x^{a}_c, that the time elapsed from the matching call and the return is at most 5 units of time. In particular, if the word is well matched the constraint imposes that the overall computation takes at most 5 units of time. The *abstract recorder* clock x^{a}_c in the return transition looping on q_1 requires that any returning procedure call (but the first one taking from q_0) returns within 5 time units. The *caller clock* x^{c}_c in the internal transition looping on q_1 requires that each internal procedural activity is performed within 1 time unit from the activation of the procedure.

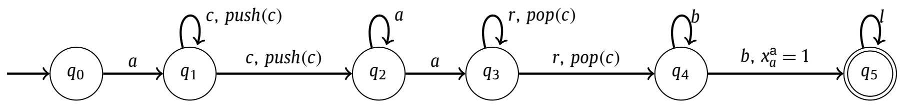


Fig. 3. Example of an abstract recording ECNA.

A configuration of $\mathcal{A}$ is a triple $(q, \beta, sval)$, where $q \in Q$, $\beta \in \Gamma^* \cdot \{\top\}$ is a stack content, and $sval$ is a valuation over C_{st}. A configuration features only the valuation of standard clocks since the valuation of event clocks deterministically depends on the input timed word. A run π of $\mathcal{A}$ over $w = (\sigma, \tau)$ is an infinite sequence of configurations $\pi = (q_0, \beta_0, sval_0), (q_1, \beta_1, sval_1), \ldots$ such that $q_0 \in Q_0$, $\beta_0 = \top$, $sval_0(z) = 0$ for all $z \in C_{st}$ (initialization requirement), and the following holds for all $i \geq 0$, where $t_i = \tau_i - \tau_{i-1}$ ($\tau_{-1} = 0$):

- **Push:** If $\sigma_i \in \Sigma_{call}$, then for some $(q_i, \sigma_i, \theta, Res, q_{i+1}, \gamma) \in \Delta_c$, $\beta_{i+1} = \gamma \cdot \beta_i$, $sval_{i+1} = (sval_i + t_i)[Res]$, and $(sval_i + t_i) \cup (val_i^w)_{|C} \models \theta$;
- **Pop:** If $\sigma_i \in \Sigma_{ret}$, then for some $(q_i, \sigma_i, \theta, Res, \gamma, q_{i+1}) \in \Delta_r$, $sval_{i+1} = (sval_i + t_i)[Res]$, $(sval_i + t_i) \cup (val_i^w)_{|C} \models \theta$, and *either* $\gamma \neq \top$ and $\beta_i = \gamma \cdot \beta_{i+1}$, *or* $\gamma = \beta_i = \beta_{i+1} = \top$;
- **Internal:** If $\sigma_i \in \Sigma_{int}$, then for some $(q_i, \sigma_i, \theta, Res, q_{i+1}) \in \Delta_i$, $\beta_{i+1} = \beta_i$, $sval_{i+1} = (sval_i + t_i)[Res]$, and $(sval_i + t_i) \cup (val_i^w)_{|C} \models \theta$.

The run π is *accepting* if there are infinitely many positions $i \geq 0$ such that $q_i \in F$. The *timed language* $\mathcal{L}_T(\mathcal{A})$ of $\mathcal{A}$ is the set of infinite timed words w over Σ such that there is an accepting run of $\mathcal{A}$ on w. The *greatest constant of* $\mathcal{A}$, denoted $K_{\mathcal{A}}$, is the greatest natural number used as bound in some clock constraint of $\mathcal{A}$. For technical convenience, we also consider nested VPTA equipped with a *generalized Büchi acceptance condition* $\mathcal{F}$ consisting of a family of sets of accepting states. In such a setting, a run π is accepting if for each Büchi component $F \in \mathcal{F}$, the run π visits infinitely often states in F.

A VPTA [26] corresponds to a nested VPTA whose set C of event clocks is empty. An *ECNA* is a nested VPTA whose set C_{st} of standard clocks is empty. For ECNA, we can omit the reset component Res from the transition function and the valuation component $sval$ from each configuration $(q, \beta, sval)$. Note that the class of Event-Clock Visibly Pushdown Automata (ECVPA) [34] corresponds to the subclass of ECNA where abstract and caller event-clocks are disallowed. We also consider three additional subclasses of ECNA: *abstract predicting* ECNA (AP_ECNA, for short) which do not use abstract recorder clocks and caller clocks, *abstract recording* ECNA (AR_ECNA, for short) which do not use abstract predictor clocks and caller clocks, and *caller* ECNA (C_ECNA, for short) which do not use abstract clocks. Note that these three subclasses of ECNA subsume ECVPA.

Example 3. Let us consider the AR_ECNA in Fig. 3, where $\Sigma_{call} = \{c\}$, $\Sigma_{ret} = \{r\}$, and $\Sigma_{int} = \{a, b, l\}$. The control part of the transition relation ensures that for each accepted word, the *MAP* visiting the b-position associated with the transition tr from q_4 to q_5 cannot visit the a-positions following the call positions. This implies that the abstract recorder constraint $x_a^{\mathsf{a}} = 1$ associated with tr is fulfilled *only if* all the occurrences of calls c and returns r are matched. Hence, constraint $x_a^{\mathsf{a}} = 1$ on transition tr ensures that the accepted language, denoted by $\mathcal{L}_T^{rec}$, consists of all the timed words of the form $(\sigma, \tau) \cdot (l^\omega, \tau')$ such that σ is a well-matched word of the form $a \cdot c^n \cdot a^m \cdot r^n \cdot b^k$ with $n, m, k > 0$, and the time difference in (σ, τ) between the first and last symbols is 1, i.e. $\tau_{|\sigma|-1} - \tau_0 = 1$. The example shows that ECNA can express a meaningful real-time property of recursive systems, namely the ability of bounding the time required to perform an internal activity consisting of an unbounded number of returning recursive procedure calls (and returns). Similarly, it is easy to define an AP_ECNA accepting the timed language, denoted by $\mathcal{L}_T^{pred}$, consisting of all the timed words of the form $(\sigma, \tau) \cdot (l^\omega, \tau')$ such that σ is a well-matched word of the form $a^k \cdot c^n \cdot b^m \cdot r^n \cdot b$, with $n, m, k > 0$, and the time difference in (σ, τ) between the first and last symbol is 1. Finally, we consider the timed language $\mathcal{L}_T^{caller}$, which can be defined by a C_ECNA, consisting of the timed words of the form $(c, t_0) \cdot (\sigma, \tau) \cdot (l^\omega, \tau')$ such that σ is a well-matched word of the form $a \cdot c^n \cdot a^m \cdot r^n \cdot b^k$, with $n, m, k > 0$, and the time difference in $(c, t_0) \cdot (\sigma, \tau)$ between the first and last symbols is 1.

Closure properties of Büchi ECNA In this section we prove that the class of languages accepted by Büchi ECNA is closed under Boolean operations.

Proposition 1. *The class of ω-timed languages accepted by Büchi ECNA is closed under union, intersection, and complementation. In particular, given two Büchi ECNA $\mathcal{A} = (\Sigma, Q, Q_0, C, \Gamma \cup \{\top\}, \Delta, F)$ and $\mathcal{A}' = (\Sigma, Q', Q_0', C', \Gamma' \cup \{\top\}, \Delta', F')$ over Σ, one can construct*

- *a Büchi ECNA accepting $\mathcal{L}_T(\mathcal{A}) \cup \mathcal{L}_T(\mathcal{A}')$ with $|Q| + |Q'|$ states, $|\Gamma| + |\Gamma'| + 1$ stacks symbols, and greatest constant* $\max(K_{\mathcal{A}}, K_{\mathcal{A}'})$*;*
- *a Büchi ECNA accepting $\mathcal{L}_T(\mathcal{A}) \cap \mathcal{L}_T(\mathcal{A}')$ with $2|Q||Q'|$ states, $|\Gamma||\Gamma'| + 1$ stacks symbols, and greatest constant* $\max(K_{\mathcal{A}}, K_{\mathcal{A}'})$.

The proofs of closure under union and intersection follow a standard construction. As for union, we can assume that $\mathcal{A}$ and $\mathcal{A}'$ have disjoint sets of states and disjoint sets of stack alphabets. The automaton accepting $\mathcal{L}_T(\mathcal{A}) \cup \mathcal{L}_T(\mathcal{A}')$ is then $(\Sigma, Q \cup Q', Q_0 \cup Q'_0, C \cup C', \Gamma \cup \Gamma' \cup \{\top\}, \Delta \cup \Delta', F \cup F')$. As for intersection, one can consider the standard synchronous product of automata with Büchi acceptance (we assume that the sets of clocks C and C' are disjoint). The automaton accepting $\mathcal{L}_T(\mathcal{A}) \cap \mathcal{L}_T(\mathcal{A}')$ is then $(\Sigma, Q \times Q' \times \{1,2\}, Q_0 \times Q'_0 \times \{1\}, C \cup C', \Gamma \times \Gamma' \cup \{\top\}, \overline{\Delta}, F \times Q' \times \{1\})$.
The set of push transitions $\overline{\Delta}_c$ has elements of the form $((q_1, q_2, b), \sigma, \theta_1 \wedge \theta_2, (q'_1, q'_2, b'), (\gamma_1, \gamma_2))$ such that $(q_1, \sigma, \theta_1, q'_1, \gamma_1) \in \Delta_c$ and $(q_2, \sigma, \theta_2, q'_2, \gamma_2) \in \Delta'_c$; $b' = b$ if either $b = 1$ and $q_1 \notin F$ or $b = 2$ and $q_2 \notin F'$, otherwise $b' \neq b$. The set of pop transitions $\overline{\Delta}_r$ and internal transitions $\overline{\Delta}_i$ can be defined similarly.

It remains to prove the closure under language complementation. For this, we adopt the technique exploited in [34] for the subclass of Büchi ECVPA. The result is obtained by exploiting the closure under complementation for Büchi VPA ([7,8]). A Büchi ECNA can be reduced to a Büchi VPA by a regionalization technique (an homomorphism from Büchi ECNA to Büchi VPA called *untimed homomorphism*), complemented, and then reduced again to Büchi ECNA by means a converse homomorphism (called *timed homomorphism*) mapping from Büchi VPA to Büchi ECNA. Note that a Büchi VPA is defined as a Büchi ECNA though omitting the set of event clocks and the set of clock constraints from the transition function. The notion of an (accepting) run of a Büchi VPA over an infinite word on Σ is similar to the notion of an (accepting) run of an ECNA over an infinite timed word on Σ, though omitting the requirements about the clock constraints. In order to define the untimed homomorphism we recall the preliminaries of the regionalization technique which encodes clocks and clock constraints of ECNA in terms of regions.

Let us fix a Büchi ECNA $\mathcal{A} = (\Sigma, Q, Q_0, C, \Gamma \cup \{\top\}, \Delta, F)$ and let $Const = \{c_0, \ldots, c_k\}$ be the set of constants used in the clock constraints of $\mathcal{A}$ ordered for increasing values, i.e. such that $0 \leq c_0 < c_1 \ldots < c_k$. We consider the following set $Intv$ of intervals over $\mathbb{R}_+ \cup \{\vdash\}$:

$$Intv := \{[\vdash, \vdash], [0,0], (0, c_0)\} \cup \bigcup_{i=0}^{i=k-1} \{[c_i, c_i], (c_i, c_{i+1})\} \cup \{[c_k, c_k], (c_k, \infty)\}.$$

A *region g of $\mathcal{A}$* is a mapping $g : C \mapsto Intv$ assigning to each event clock in $C \subseteq C_\Sigma$ an interval in $Intv$. The mapping g induces the clock constraint $\bigwedge_{z \in C} z \in g(z)$. We denote by $[g]$ the set of valuations over C satisfying the clock constraint associated with g, and by Reg the set of regions of $\mathcal{A}$. For a clock constraint θ over C, let $[\theta]$ be the set of valuations over C satisfying θ.

Remark 1. By construction, the following holds.

- The set Reg of regions represents a partition of the set of clock valuations over C, i.e.: (i) for all valuations val over C, there is a region $g \in Reg$ such that $val \in Reg$, and (ii) for all regions $g, g' \in Reg$, $g \neq g' \Rightarrow [g] \cap [g'] = \emptyset$.
- For each clock constraint θ of $\mathcal{A}$ and region $g \in Reg$, either $[g] \subseteq [\theta]$ or $[g] \cap [\theta] = \emptyset$.

We associate with $\Sigma = \Sigma_{call} \cup \Sigma_{ret} \cup \Sigma_{int}$ and the set of regions Reg a pushdown alphabet $\Lambda = \Sigma \times Reg$, called *interval pushdown alphabet*, whose set of calls (resp. returns, internal actions) is $\Sigma_{call} \times Reg$ (resp. $\Sigma_{ret} \times Reg$, $\Sigma_{int} \times Reg$). Elements of Λ are pairs of the form (a, g), where $a \in \Sigma$ and g is a region of $\mathcal{A}$ representing the associated constraint $\bigwedge_{z \in C} z \in g(z)$. An infinite word $\lambda = (a_0, g_0), (a_1, g_1), \ldots$ over Λ induces in a natural way a set of infinite timed words over Σ, denoted $tw(\lambda)$, defined as follows: $w = (\sigma, \tau) \in tw(\lambda)$ iff $\sigma = a_0 a_1 \ldots$ and for all $i \geq 0$, $val_i^w \in [g_i]$. We extend the mapping tw to ω-languages $\mathcal{L}$ over Λ in the obvious way: $tw(\mathcal{L}) := \bigcup_{\lambda \in \mathcal{L}} tw(\lambda)$. By means of the mapping tw, infinite words over Λ define a partition of the set of infinite timed words over Σ.

Lemma 1. *The following holds.*

1. *For each infinite timed word $w = (\sigma, \tau)$ over Σ, there is an infinite word λ over Λ of the form $(\sigma_0, g_0)(\sigma_1, g_1)$ such that $w \in tw(\lambda)$.*
2. *For all infinite words λ and λ' over Λ, $\lambda \neq \lambda' \Rightarrow tw(\lambda) \cap tw(\lambda') = \emptyset$.*

Proof. For Property 1, let $w = (\sigma, \tau)$ be an infinite timed word over Λ. By Remark 1, for all $i \geq 0$, there is a region $g_i \in Reg$ such that $val_i^w \in [g_i]$. Let $\lambda = (\sigma_0, g_0)(\sigma_1, g_1) \ldots$. We have that $w \in tw(\lambda)$, and the result follows.

For Property 2, let λ and λ' be two distinct infinite words over λ. Let us assume that $tw(\lambda) \cap tw(\lambda') \neq \emptyset$ and derive a contradiction. Hence, by construction, $\lambda = (a_0, g_0)(a_1, g_1) \ldots$, $\lambda' = (a_0, g'_0)(a_1, g'_1) \ldots$, and there is an infinite timed word w over Σ of the form $(a_0, \tau_0)(a_1, \tau_1)$ such that $val_i^w \in [g_i] \cap [g'_i]$ for all $i \geq 0$. Since $\lambda \neq \lambda'$, there exists $n \geq 0$ such that $g_n \neq g'_n$. By Remark 1, $[g_n] \cap [g'_n] = \emptyset$ which is a contradiction since $val_n^w \in [g_n] \cap [g'_n]$. □

We state now the correctness of reduction from Büchi ECNA to Büchi VPA.

Proposition 2 *(Untimed homomorphism). Let $\mathcal{A} = (\Sigma, Q, Q_0, C, \Gamma \cup \{\top\}, \Delta, F)$ be a Büchi ECNA, and Λ be the interval pushdown alphabet induced by $\mathcal{A}$. Then, one can construct a Büchi VPA Untimed$(\mathcal{A})$ over Λ of the form $(\Lambda, Q, Q_0, \Gamma \cup \{\top\}, \Delta', F)$ such that $tw(\mathcal{L}(Untimed(\mathcal{A}))) = \mathcal{L}_T(\mathcal{A})$.*

Proof. The transition function Δ' of *Untimed*$(\mathcal{A})$ is defined as follows:

- Push: If $(q, c, \theta, q', \gamma)$ is a push transition in Δ, then for each region g of $\mathcal{A}$ such that $[g] \subseteq [\theta]$, $(q, (c, g), q', \gamma) \in \Delta'$.
- Pop: If $(q, r, \theta, \gamma, q')$ is a pop transition in Δ, then for each region g of $\mathcal{A}$ such that $[g] \subseteq [\theta]$, $(q, (r, g), \gamma, q') \in \Delta'$.
- Internal: If (q, a, θ, q') is an internal transition in Δ, then for each region g of $\mathcal{A}$ such that $[g] \subseteq [\theta]$, $(q, (a, g), q') \in \Delta'$.

The correctness of the construction easily follows from Remark 1 and Lemma 1(1). $\square$

We state in the following the converse homomorphism from Büchi VPA to Büchi ECNA.

Proposition 3 *(Timed homomorphism). Let $\mathcal{A} = (\Lambda, Q, Q_0, \Gamma \cup \{\top\}, \Delta, F)$ be a Büchi VPA over an interval pushdown alphabet associated with Σ and a set $C \subseteq C_\Sigma$ of event clocks. Then, one can construct a Büchi ECNA Timed$(\mathcal{A})$ over Σ of the form $(\Sigma, Q, Q_0, C, \Gamma \cup \{\top\}, \Delta', F)$ such that $\mathcal{L}_T(Timed(\mathcal{A})) = tw(\mathcal{L}(\mathcal{A}))$.*

Proof. Let g be a region and $\theta_g := \bigwedge_{z \in C} z \in g(z)$. The transition function Δ' of *Timed*$(\mathcal{A})$ is defined as follows:

- Push: If $(q, (c, g), q', \gamma) \in \Delta$ is a push transition, then $(q, c, \theta_g, q', \gamma) \in \Delta'$;
- Pop: If $(q, (r, g), \gamma, q') \in \Delta$ is a pop transition, then $(q, r, \theta_g, \gamma, q') \in \Delta'$;
- Internal: If $(q, (r, g), q') \in \Delta$ is an internal transition, then $(q, r, \theta_g, q') \in \Delta'$.

The correctness of the construction easily follows from Remark 1 and Lemma 1(1). $\square$

The closure under complementations of ECNA follows now from Lemma 1, Propositions 2 and 3, and the known closure properties of Büchi Visibly Pushdown Automata (VPA) [7,8].

Theorem 2 *(Closure under complementation). Given a Büchi ECNA $\mathcal{A}$ over Σ with n states and set of constants Const, one can construct in singly exponential time a Büchi ECNA $\overline{\mathcal{A}}$ over Σ accepting the complement of $\mathcal{L}_T(\mathcal{A})$ having $2^{O(n^2)}$ states and $O(2^{O(n^2)} \cdot |\Sigma_{call}| \cdot |Const|^{O(|\Sigma|)})$ stack symbols.*

Proof. Let $\mathcal{A}$ be a Büchi ECNA over Σ with n states and set *Const* of integer constants, Λ be the interval pushdown alphabet induced by $\mathcal{A}$, and $\Lambda_c \subseteq \Lambda$ be the set of calls. By Proposition 2, we can construct a Büchi VPA *Untimed*$(\mathcal{A})$ over Λ with n states such that $tw(\mathcal{L}(Untimed(\mathcal{A}))) = \mathcal{L}_T(\mathcal{A})$. By [7,8], starting from the Büchi VPA *Untimed*$(\mathcal{A})$, one can construct in singly exponential time a Büchi VPA $\overline{Untimed(\mathcal{A})}$ over Λ accepting $\Lambda^\omega \setminus \mathcal{L}(Untimed(\mathcal{A}))$ with $2^{O(n^2)}$ states and $O(2^{O(n^2)} \cdot |\Lambda_c|)$ stack symbols.
By applying Proposition 3 to the Büchi VPA $\overline{Untimed(\mathcal{A})}$, we can build in linear time a Büchi ECNA $\overline{\mathcal{A}}$ over Σ with $2^{O(n^2)}$ states and $O(2^{O(n^2)} \cdot |\Lambda_c|)$ stack symbols such that $\mathcal{L}_T(\overline{\mathcal{A}}) = tw(\Lambda^\omega \setminus \mathcal{L}(Untimed(\mathcal{A})))$. Since $\mathcal{L}_T(\mathcal{A}) = tw(\mathcal{L}(Untimed(\mathcal{A})))$, by Lemma 1, $\overline{\mathcal{A}}$ accepts all and only the infinite timed words over Σ which are not in $\mathcal{L}_T(\mathcal{A})$. Moreover, we have $|\Lambda_c| = O(|\Sigma_{call}| \cdot |Const|^{O(|\Sigma|)})$. $\square$

Expressiveness results In this section we present the expressiveness results for ECNA. The overall picture is stated in Theorem 4. We start showing that the three subclasses AR_ECNA, AP_ECNA, and C_ECNA of ECNA are mutually incomparable. In fact, each of the timed languages $\mathcal{L}_T^{rec}$, $\mathcal{L}_T^{pred}$, and $\mathcal{L}_T^{caller}$ considered in Example 3 and definable in AR_ECNA, AP_ECNA, and C_ECNA, respectively, can be expressed only in one of these subclasses.

Lemma 3. *The language $\mathcal{L}_T^{rec}$ (resp. $\mathcal{L}_T^{pred}$, $\mathcal{L}_T^{caller}$) is not definable by Büchi ECNA without abstract recorder clocks (resp. abstract predictor clocks, caller clocks). Moreover, the language $\mathcal{L}_T^{rec} \cup \mathcal{L}_T^{pred} \cup \mathcal{L}_T^{caller}$ is not definable by Büchi AR_ECNA, Büchi AP_ECNA and Büchi C_ECNA.*

Proof. First, let us consider the timed language $\mathcal{L}_T^{rec}$ defined in Example 3 which consists of all the timed words of the form $(\sigma, \tau) \cdot (\imath^\omega, \tau')$ such that σ is a well-matched word of the form $a \cdot c^+ \cdot a^+ \cdot r^+ \cdot b^+$ and the time difference in (σ, τ) between the first and last symbols is 1. Let $v_1 = (a, 0) \cdot (c, 0.1) \cdot (a, 0.1) \cdot (r, 0.1) \cdot (b, 0.1) \cdot (b, 0.9)$ and $v_2 = (a, 0) \cdot (c, 0.1) \cdot (a, 0.1) \cdot (r, 0.1) \cdot (b, 0.1) \cdot (b, 1)$ be two timed words over Σ of length 6. For each $H \geq 1$, let $w_1^H = v_1 \cdot (\imath, H+2) \cdot (\imath, H+3) \ldots$ and $w_2^H = v_2 \cdot (\imath, H+2) \cdot (\imath, H+3) \ldots$. Let us denote by $val^{1,H}$ and $val^{2,H}$ the event-clock valuations over C_Σ associated with w_1^H and w_2^H, respectively. By construction, for all positions $i \geq 0$ and event-clock $z \in C_\Sigma$ such that z is *not* an abstract recorder clock, the following holds:

- either (i) $val_i^{1,H}(z) = val_i^{2,H}(z)$, or (ii) $0 < val_i^{1,H}(z) < 1$ and $0 < val_i^{2,H}(z) < 1$, or (iii) $val_i^{1,H}(z) > H$ and $val_i^{2,H}(z) > H$.

Hence, clock constraints which do not use abstract recorder clocks and whose maximum constant is at most H cannot distinguish the valuations $val^{1,H}$ and $val^{2,H}$. It follows that for each ECNA $\mathcal{A}$ over Σ which does not use abstract recorder clocks and has maximum constant H, $w_1^H \in \mathcal{L}_T(\mathcal{A})$ iff $w_2^H \in \mathcal{L}_T(\mathcal{A})$. On the other hand, by definition of the language $\mathcal{L}_T^{rec}$, for each $H \geq 1$, $w_2^H \in \mathcal{L}_T^{rec}$ and $w_1^H \notin \mathcal{L}_T^{rec}$. Hence, $\mathcal{L}_T^{rec}$ is not definable by Büchi ECNA which do not use abstract recorder clocks.

Now, let us consider the timed language $\mathcal{L}_T^{pred}$ which consists of all the timed words of the form $(\sigma, \tau) \cdot (l^\omega, \tau')$ such that σ is a well-matched word of the form $a^+ \cdot c^+ \cdot b^+ \cdot r^+ \cdot b$ and the time difference in (σ, τ) between the two extreme symbols is 1. Let $u_1 = (a,0) \cdot (a,0.1) \cdot (c,0.1) \cdot (b,0.1) \cdot (r,0.1) \cdot (b,0.9)$ and $u_2 = (a,0) \cdot (a,0.1) \cdot (c,0.1) \cdot (b,0.1) \cdot (r,0.1) \cdot (b,1)$ be two timed words over Σ of length 6. For each $H \geq 1$, let $r_1^H = u_1 \cdot (l, H+2) \cdot (l, H+3) \dots$ and $r_2^H = u_2 \cdot (l, H+2) \cdot (l, H+3) \dots$. By reasoning as in the case of language $\mathcal{L}_T^{rec}$, we have that for each ECNA $\mathcal{A}$ over Σ which does not use abstract predictor clocks and has as maximum constant H, $r_1^H \in \mathcal{L}_T(\mathcal{A})$ iff $r_2^H \in \mathcal{L}_T(\mathcal{A})$. On the other hand, by definition of the language $\mathcal{L}_T^{pred}$, for each $H \geq 1$, $r_2^H \in \mathcal{L}_T^{pred}$ and $r_1^H \notin \mathcal{L}_T^{pred}$ proving that $\mathcal{L}_T^{pred}$ is not definable by Büchi ECNA which do not use abstract predictor clocks.
The proof for the language $\mathcal{L}_T^{caller}$ is similar and omitted. Finally, by the above considerations, it follows that $\mathcal{L}_T^{rec} \cup \mathcal{L}_T^{pred} \cup \mathcal{L}_T^{caller}$ is not definable neither by an abstract-predicting Büchi ECNA nor by an abstract-recording Büchi ECNA nor by a caller Büchi ECNA. □

As a consequence of Lemma 3, the classes AR_ECNA, AP_ECNA, and C_ECNA strictly include the class of ECVPA, are strictly included in ECNA, and are pairwise incomparable.

As for ECNA, they are less expressive than Büchi VPTA. In fact, as will be shown in Theorem 5 of Section 4, Büchi ECNA can be converted into equivalent Büchi VPTA showing the inclusion ECNA $\subseteq$ VPTA. The inclusion is strict since Büchi VPTA are not closed under complementation (see [26]) whereas Büchi ECNA are closed as proved in Theorem 2.

Finally, we can compare ECNA with ECVPA. In [15], an equally-expressive extension of ECVPA over finite timed words, by means of a *timed* stack, is investigated. The Büchi version of such an extension can be trivially encoded in Büchi AR_ECNA. Moreover, the proof of Lemma 3 can also be used to show that Büchi ECVPA with timed stack are less expressive than Büchi AR_ECNA, Büchi AP_ECNA, and Büchi C_ECNA.
The following summarizes the general picture of the expressiveness results.

Theorem 4. *The classes AR_ECNA, AP_ECNA, and C_ECNA are mutually incomparable, and AP_ECNA $\cup$ AR_ECNA $\cup$ C_ECNA $\subset$ ECNA. Moreover,*

(1) *ECVPA $\subset$ AR_ECNA* (2) *ECVPA $\subset$ AP_ECNA*
(3) *ECVPA $\subset$ C_ECNA* (4) *ECNA $\subset$ VPTA*

Note that the expressiveness results above also hold for finite timed words.

4. Decision procedures for Büchi ECNA

In this section, we first investigate emptiness, universality, and language inclusion problems for Büchi ECNA. Then, we consider the *Visibly model-checking problem against Büchi ECNA*, i.e., given a *visibly pushdown timed system* $\mathcal{S}$ over Σ (that is a Büchi VPTA where all the states are accepting) and a Büchi ECNA $\mathcal{A}$ over Σ, the problem whether $\mathcal{L}_T(\mathcal{S}) \subseteq \mathcal{L}_T(\mathcal{A})$ holds. We prove that the above mentioned problems are decidable and EXPTIME-complete. Notice that in this approach the model is given by a Büchi VPTA, whereas the property to be checked is given by a Büchi ECNA. As proved in the previous Section 3, ECNA are less expressive than VPTA but are closed under boolean operations. This allow us to reduce the inclusion problem $\mathcal{L}_T(\mathcal{S}) \subseteq \mathcal{L}_T(\mathcal{A})$ to the emptiness problem of the intersection of the language $\mathcal{L}_T(\mathcal{S})$ with the complement of $\mathcal{L}_T(\mathcal{A})$ (i.e. $\mathcal{L}_T(\mathcal{S}) \cap \overline{\mathcal{L}_T(\mathcal{A})}$). The language $\overline{\mathcal{L}_T(\mathcal{A})}$ is accepted by a Büchi ECNA as proved in Theorem 2. The key intermediate result for the reduction is an exponential-time translation of Büchi ECNA into language-equivalent generalized Büchi VPTA. By exploiting such a translation we can intersect the Büchi VPTA for $\mathcal{L}_T(\mathcal{S})$ with the Büchi VPTA obtained by translating the Büchi ECNA for $\overline{\mathcal{L}_T(\mathcal{A})}$ to VPTA.

We start defining the translation from Büchi ECNA to Büchi VPTA. Since a VPTA is a nested VPTA with standard clocks but devoid of event clocks, we shall show how the event clocks in the considered ECNA can be encoded in terms of standard clocks with a single exponential blow-up. The transformation from a generalized Büchi *nested* VPTA $\mathcal{A}$ to a generalized Büchi VPTA $\mathcal{A}'$ is obtained by a sequence of transformation steps all preserving language equivalence. At each step, an event clock is replaced by a set of fresh standard clocks. To remove global event clocks we borrow the technique from [5].

In the following we sketch the ideas for removing an *abstract predictor* clock y_b^{a} with $b \in \Sigma$ (the removal of abstract recorder clocks and caller clocks can be dealt with similarly) reporting the details of the construction in the proof of Theorem 5. Let us fix a generalized Büchi nested VPTA $\mathcal{A}$ and a clock y_b^{a} belonging to the set of clocks C of $\mathcal{A}$. Without loss of generality, we assume that for each transition *tr* of $\mathcal{A}$, there is exactly one atomic constraint $y_b^{\mathsf{a}} \in I$ involving y_b^{a} used as conjunct in the clock constraint of tr. If $I \neq \{\vdash\}$, then $y_b^{\mathsf{a}} \in I$ is equivalent to a constraint of the form $y_b^{\mathsf{a}} \succ \ell \wedge y_b^{\mathsf{a}} \prec u$, where $\succ \in \{>, \geq\}$, $\prec \in \{<, \leq\}$, $\ell \in \mathbb{N}$, and $u \in \mathbb{N} \cup \{\infty\}$. We call $y_b^{\mathsf{a}} \succ \ell$ (resp., $y_b^{\mathsf{a}} \prec u$) a lower-bound (resp., upper-bound) constraint. Note that if $u = \infty$, the constraint $y_b^{\mathsf{a}} \prec u$ is always fulfilled, but we include it to have a uniform notation. We construct a generalized Büchi nested VPTA $\mathcal{A}_{y_b^{\mathsf{a}}}$ equivalent to $\mathcal{A}$ whose set of event clocks is $C \setminus \{y_b^{\mathsf{a}}\}$, and whose set of standard clocks is $C_{st} \cup C_{new}$, where C_{new} consists of the fresh standard clocks $z_{\succ\ell}$ (resp., $z_{\prec u}$), for each lower-bound constraint $y_b^{\mathsf{a}} \succ \ell$ (resp., upper-bound constraint $y_b^{\mathsf{a}} \prec u$) of $\mathcal{A}$ involving y_b^{a}.

The sketch of the translation follows. Consider a lower-bound constraint $y_b^{\mathsf{a}} \succ \ell$. Assume that a prediction $y_b^{\mathsf{a}} \succ \ell$ is done by $\mathcal{A}$ at position i of the input word for the first time. Then, the simulating automaton $\mathcal{A}_{y_b^{\mathsf{a}}}$ exploits the standard clock $z_{\succ\ell}$ to check that the prediction holds by resetting it at position i. Moreover, if i is not a call (resp., i is a call), $\mathcal{A}_{y_b^{\mathsf{a}}}$ carries *the obligation* $\succ\ell$ in its control state (resp., pushes the obligation $\succ\ell$ onto the stack) in order to check that the constraint $z_{\succ\ell} \succ \ell$ holds when the next b occurs at a position j_{check} along the *MAP* ν visiting position i. We observe that:

- If a new prediction $y_b^{\mathsf{a}} \succ \ell$ is done by $\mathcal{A}$ at a position $j > i$ of ν strictly preceding j_{check}, $\mathcal{A}_{y_b^{\mathsf{a}}}$ can rewrite the old obligation by resetting the clock $z_{\succ\ell}$ at position j. This is safe since the fulfillment of the lower-bound prediction $y_b^{\mathsf{a}} \succ \ell$ at j guarantees that the prediction $y_b^{\mathsf{a}} \succ \ell$ is fulfilled at i along ν.
- If a call position $i_c \geq i$ occurs in ν before j_{check}, the next position of i_c in ν is the matching return i_r of i_c, and any *MAP* visiting a position $h \in [i_c+1, i_r-1]$ is finite and ends at a position $k < i_r$. Thus, the clock $z_{\succ\ell}$ can be safely reset to check the prediction $y_b^{\mathsf{a}} \succ \ell$ raised in positions in $[i_c+1, i_r-1]$ since this check ensures that $z_{\succ\ell} \succ \ell$ holds at position j_{check}.

Thus, previous obligations on a constraint $y_b^{\mathsf{a}} \succ \ell$ are always rewritten by more recent ones. At each position i, $\mathcal{A}_{y_b^{\mathsf{a}}}$ records in its control state the lower-bound obligations for the current *MAP* ν (i.e., the *MAP* visiting the current position i). Whenever a call i_c occurs, the lower-bound obligations are pushed on the stack in order to be recovered at the matching return i_r. If $i_c + 1$ is not a return (i.e., $i_r \neq i_c + 1$), then $\mathcal{A}_{y_b^{\mathsf{a}}}$ moves to a control state having an empty set of lower-bound obligations (position $i_c + 1$ starts the *MAP* visiting $i_c + 1$).

The treatment of an upper-bound constraint $y_b^{\mathsf{a}} \prec u$ is symmetric. Whenever a prediction $y_b^{\mathsf{a}} \prec u$ is done by $\mathcal{A}$ at a position i, and the simulating automaton $\mathcal{A}_{y_b^{\mathsf{a}}}$ has no obligation on the constraint $y_b^{\mathsf{a}} \prec u$, $\mathcal{A}_{y_b^{\mathsf{a}}}$ resets the standard clock $z_{\prec u}$. If i is not a call (resp., i is a call) the fresh obligation $(\mathit{first}, \prec u)$ is recorded in the control state (resp., $(\mathit{first}, \prec u)$ is pushed onto the stack). When, along the *MAP* ν visiting position i, the next b occurs at a position j_{check}, the constraint $z_{\prec u} \prec u$ is checked, and the obligation $(\mathit{first}, \prec u)$ is removed or confirmed (in the latter case, resetting the clock $z_{\prec u}$), depending on whether the prediction $y_b^{\mathsf{a}} \prec u$ is asserted at position j_{check} or not. We observe that:

- If a new prediction $y_b^{\mathsf{a}} \prec u$ occurs in a position $j > i$ of ν strictly preceding j_{check}, $\mathcal{A}_{y_b^{\mathsf{a}}}$ ignores it (the clock $z_{\prec u}$ is not reset at position j) since checking the prediction $y_b^{\mathsf{a}} \prec u$ at position i guarantees the prediction $y_b^{\mathsf{a}} \prec u$ at a position $j > i$ along ν.
- If a call position $i_c \geq i$ occurs in ν before j_{check}, then all the predictions $y_b^{\mathsf{a}} \prec u$ occurring in a *MAP* visiting a position $h \in [i_c+1, i_r-1]$, with $i_r \leq j_{check}$ being the matching-return of i_c, can be safely ignored (i.e., $z_{\prec u}$ is not reset in the *MAP*) since they are subsumed by the prediction at position i.

Thus, for new obligations on an upper-bound constraint $y_b^{\mathsf{a}} \prec u$, the clock $z_{\prec u}$ is not reset. Whenever a call i_c occurs, the updated set O of upper-bound and lower-bound obligations is pushed onto the stack to be recovered at the matching return i_r of i_c. Moreover, if $i_c + 1$ is not a return (i.e., $i_r \neq i_c + 1$), then $\mathcal{A}_{y_b^{\mathsf{a}}}$ moves to a control state where the set of lower-bound obligations is empty and the set of upper-bound obligations is obtained from O by replacing each upper-bound obligation $(f, \prec u)$, for $f \in \{\mathit{live}, \mathit{first}\}$, with the live obligation $(\mathit{live}, \prec u)$. The latter asserted at the initial position $i_c + 1$ of the *MAP* ν visiting $i_c + 1$ (note that ν ends at $i_r - 1$) is used by $\mathcal{A}_{y_b^{\mathsf{a}}}$ to remember that the clock $z_{\prec u}$ cannot be reset along ν. Intuitively, live upper-bound obligations are propagated from the caller *MAP* to the called *MAP*. Note that fresh upper-bound obligations $(\mathit{first}, \prec u)$ always refer to predictions done along the current *MAP* and, differently from the live upper-bound obligations, they can be removed when the next b occurs along the current *MAP*.

Extra technicalities are needed to ensure that the lower-bound obligations and fresh upper-bound obligations at the current position are eventually checked, i.e., the current *MAP* eventually visits a b-position. A different treatment is required for finite and infinite *MAP*s. If a *MAP* is finite, we can ensure the requirement in the transition function of $\mathcal{A}_{y_b^{\mathsf{a}}}$ by using suitable guesses about the finiteness of the current *MAP* and whether the current position is the last position (if the *MAP* is finite). Obviously, in this case a b-position must be visited (and the obligations checked) at least when the last position is reached. For the infinity case we observe that at most one infinite *MAP* ν exists along a word. In this case, the b-liveness requirement is checked by suitably exploiting the acceptance condition (a Büchi component of $\mathcal{A}_{y_b^{\mathsf{a}}}$ is devoted to that purpose).

In more detail, at each position i, the automaton $\mathcal{A}_{y_b^{\mathsf{a}}}$ guesses whether i is the last position of the current *MAP* (i.e., the *MAP* visiting i). For this, it keeps track in its control state of the guessed type (call, return, or internal symbol) of the next input symbol. In particular, when i is a call, $\mathcal{A}_{y_b^{\mathsf{a}}}$ guesses whether the call i has a matching return. If the guess is "no", $\mathcal{A}_{y_b^{\mathsf{a}}}$ pushes onto the stack a special symbol, say *bad*, and the guess is correct iff the symbol is never popped from the stack.

To discriminate positions belonging to finite *MAP*s from those belonging to the unique infinite *MAP* (if any), $\mathcal{A}_{y_b^{\mathrm{a}}}$ exploits a special proposition p_∞ whose Boolean value is carried in the control state. The intended meaning is that p_∞ *does not hold* at a position j of the input iff the *MAP* visiting j has a caller whose matching return exists. If the *MAP* ν is infinite, then it visits only positions where p_∞ holds. Moreover, each position i greater than the initial position i_0 of ν is either a ν-position, or a position where p_∞ does not hold. If an infinite word has no infinite *MAP*, then p_∞ holds at infinitely many positions as well. The transition function of $\mathcal{A}_{y_b^{\mathrm{a}}}$ ensures that the Boolean value of p_∞ is propagated consistently with the guesses. Moreover, the guesses about the matched calls are correct iff p_∞ is asserted infinitely often along a run (this can be ensured by the acceptance condition by exploiting a Büchi component of $\mathcal{A}_{y_b^{\mathrm{a}}}$).

Theorem 5 *(Removal of event clocks from nested VPTA). Given a generalized Büchi* nested *VPTA $\mathcal{A}$, one can construct in singly exponential time a generalized Büchi VPTA $\mathcal{A}'$ (not using event clocks) such that $\mathcal{L}_T(\mathcal{A}') = \mathcal{L}_T(\mathcal{A})$ and $K_{\mathcal{A}'} = K_{\mathcal{A}}$. Moreover, $\mathcal{A}'$ has $n \cdot 2^{O(p\cdot|\Sigma|)}$ states and $m + O(p)$ clocks, where n is the number of $\mathcal{A}$-states, m is the number of standard $\mathcal{A}$-clocks, and p is the number of* event-clock *atomic constraints used by $\mathcal{A}$.*

Proof. Let us fix a generalized Büchi nested VPTA $\mathcal{A} = (\Sigma, Q, Q_0, C \cup C_{st}, \Gamma \cup \{\top\}, \Delta, \mathcal{F})$. In the proof we focus on the removal of an *abstract predictor* clock y_b^{a} with $b \in \Sigma$, referring to Appendix A and Appendix B for the treatment of abstract recorder clocks and caller clocks, respectively. We need some preliminary definitions.

An *obligation set* O (for the fixed abstract predictor clock y_b^{a} and the fixed generalized Büchi nested VPTA $\mathcal{A}$) is a set consisting of lower-bound obligations $\succ\ell$ and upper-bound obligations $(f, \prec u)$, where $f \in \{live, first\}$, such that $y_b^{\mathrm{a}} \succ \ell$ and $y_b^{\mathrm{a}} \prec u$ are associated to interval constraints of $\mathcal{A}$, and $(f, \prec u), (f', \prec u) \in O$ implies $f = f'$. For an obligation set O, $live(O)$ is the obligation set consisting of the live upper-bound obligations of O.

Let us consider the expression $\diamond^{\mathrm{a}} b$ (actually a CaRet formula [3]) with the intended meaning that $\diamond^{\mathrm{a}} b$ holds at position $i \geq 0$ if the *MAP* visiting i also visits a position $j \geq i$ where b holds. A *check set* H is a subset of $\{call, ret, int, b, \diamond^{\mathrm{a}} b, p_\infty\}$ such that $H \cap \{call, ret, int\}$ is a singleton. A check set is done by $\mathcal{A}_{y_b^{\mathrm{a}}}$ for keeping track of: (i) the guessed type (call, return, or internal symbol) of the next input symbol, (ii) whether the next input symbol is b, (iii) whether p_∞ holds at the current position, and (iv) whether $\diamond^{\mathrm{a}} b$ holds at the current position.

Let C_{new} be the set of standard clocks consisting of the fresh standard clocks $z_{\succ\ell}$ (resp., $z_{\prec u}$) for each lower-bound constraint $y_b^{\mathrm{a}} \succ \ell$ (resp., upper-bound constraint $y_b^{\mathrm{a}} \prec u$) of $\mathcal{A}$ involving y_b^{a}. For an input symbol $a \in \Sigma$ and an obligation set O, we denote by $con(O, a)$ the constraint over the new set C_{new} of standard clocks defined as: $con(O, a) = \top$ if either $O = \emptyset$ or $b \neq a$; otherwise, $con(O, a)$ is obtained from O by adding for each obligation $\succ \ell$ (resp., $(f, \prec u)$) in O, the conjunct $z_{\succ\ell} \succ \ell$ (resp., $z_{\prec u} \prec u$). The nested VPTA $\mathcal{A}_{y_b^{\mathrm{a}}}$ is given by

$$\mathcal{A}_{y_b^{\mathrm{a}}} = (\Sigma, Q', Q_0', C \setminus \{y_b^{\mathrm{a}}\} \cup C_{st} \cup C_{new}, (\Gamma \times \Gamma') \cup \{bad, \top\}, \Delta', \mathcal{F}').$$

The set Q' of states consists of triples of the form (q, O, H) such that $q \in Q$ is a state of $\mathcal{A}$, O is an obligation set, and H is a check set, while the set Q_0' of initial states consists of states of the form $(q_0, \emptyset, H)$ such that $q_0 \in Q_0$ (initially there are no obligations). The stack alphabet is $(\Gamma \times \Gamma') \cup \{bad, \top\}$, where Γ' is the set of pairs (O, H) such that O is an obligation set and H is a check set.

We now define the transition function Δ'. For this, we first define a predicate *Abs* over tuples of the form $((O, H), a, y_b^{\mathrm{a}} \in I, Res, (O', H'))$ where $(O, H), (O', H')$ are pairs of obligation sets and check sets, $a \in \Sigma$, $y_b^{\mathrm{a}} \in I$ is a constraint of $\mathcal{A}$ involving y_b^{a}, and $Res \subseteq C_{new}$. Intuitively, O (resp., H) represents the obligation set (resp., check set) at the current position i of the input, a is the input symbol associated with position i, $y_b^{\mathrm{a}} \in I$ is the prediction about y_b^{a} done by $\mathcal{A}$ at position i, *Res* is the set of new standard clocks reset by $\mathcal{A}_{y_b^{\mathrm{a}}}$ on reading a, and O' (resp., H') represents the obligation set (resp., check set) at the position j following i along the *MAP* visiting i (if i is a call, then j is the matching-return of i). Formally, $Abs((O, H), a, y_b^{\mathrm{a}} \in I, Res, (O', H'))$ iff the following holds:

1. ($p_\infty \in H$ iff $p_\infty \in H'$), $a \in \Sigma_{call}$ (resp., $a \in \Sigma_{ret}$, resp. $a \in \Sigma_{int}$) implies $call \in H$ (resp., $ret \in H$, resp., $int \in H$);
2. $\diamond^{\mathrm{a}} b \in H$ iff ($b = a$ or $\diamond^{\mathrm{a}} b \in H'$), and ($\diamond^{\mathrm{a}} b \in H'$ iff $I \neq \{\vdash\}$);
3. If $I = \{\vdash\}$, then $O' = live(O)$, $Res = \emptyset$, and $b \neq a$ implies $O = live(O)$. Otherwise, let $y_b^{\mathrm{a}} \in I \equiv y_b^{\mathrm{a}} \succ \ell \wedge y_b^{\mathrm{a}} \prec u$. Let O'' be O if $b \neq a$, and $O'' = live(O)$ otherwise. Then, $O' = O'' \cup \{\succ\ell\} \cup \{(f, \prec u)\}$, where $f = live$ if $(live, \prec u) \in O''$, and $f = first$ otherwise. Moreover, $Res \subseteq \{z_{\succ\ell}, z_{\prec u}\}$, $z_{\succ\ell} \in Res$, and $z_{\prec u} \in Res$ iff either $\prec u$ does not appear in O, or $b = a$ and $(first, \prec u) \in O$.

Condition 1 requires that the Boolean value of proposition p_∞ is invariant along the positions of a *MAP*, and the current check set is consistent with the type (call, return, or internal symbol) of the current input symbol. Condition 2 provides the abstract-local propagation rules of formula $\diamond^{\mathrm{a}} b$. Finally, Condition 3 provides the rules for updating the obligations on moving to the abstract next position along the current *MAP* and for resetting new clocks on reading the current input symbol a. Note that if $I = \{\vdash\}$ and $b \neq a$, then the current obligation set must contain only live upper-bound obligations. If, instead, $y_b^{\mathrm{a}} \in I$ is equivalent to $y_b^{\mathrm{a}} \succ \ell \wedge y_b^{\mathrm{a}} \prec u$, then the clock $z_{\succ\ell}$ is reset, while the clock $z_{\prec u}$ is reset iff either there is no obligation $(f, \prec u)$ in O, or $b = a$ and the obligation $(f, \prec u)$ is fresh, i.e., $f = first$.

The transition function Δ' of $\mathcal{A}_{y^{a}_{b}}$ is defined as follows. Recall that we assume that each clock constraint of $\mathcal{A}$ is of the form $\theta \wedge y^{a}_{b} \in I$, where θ does not contain occurrences of y^{a}_{b}.

Push transitions for each push transition $q \xrightarrow{a,\theta\wedge y^{a}_{b}\in I,Res,push(\gamma)} q'$ of $\mathcal{A}$, we have the push transitions $(q, O, H) \xrightarrow{a,\theta\wedge con(O,a),Res\cup Res',push(\gamma')} (q', O', H')$ such that $b = a$ iff $b \in H$, and

1. Case $\gamma' \neq bad$. Then, $\gamma' = (\gamma, O_{ret}, H_{ret})$ and $Abs((O, H), a, y^{a}_{b} \in I, Res', (O_{ret}, H_{ret}))$ holds. Moreover, if $ret \in H'$ then $H_{ret} = H'$ and $O' = O_{ret}$; otherwise, $p_{\infty} \notin H'$ and O' consists of the live obligations $(live, \prec u)$ such that $(f, \prec u) \in O_{ret}$ for some $f \in \{live, first\}$.
2. Case $\gamma' = bad$: $call \in H$, $I = \{\vdash\}$, ($\diamond^{a}b \in H$ iff $b = a$), $p_{\infty} \in H$, $p_{\infty} \in H'$, $ret \notin H'$, $O' = \emptyset$, $Res' = \emptyset$, and $b \neq a$ implies $O = \emptyset$.

Note that if $b = a$, the obligations in the current state are checked by the constraint on C_{new} given by $con(O, a)$ (recall that if $b \neq a$, then $con(O, a) = \top$). The push transitions of point 1 consider the case where $\mathcal{A}_{y^{a}_{b}}$ guesses that the current call position i_c has a matching return i_r. In this case, the set of obligations and the check state for the next abstract position i_r along the current *MAP* are pushed on the stack in order to be recovered at the matching-return i_r. Moreover, if $\mathcal{A}_{y^{a}_{b}}$ guesses that the next position $i_c + 1$ is not i_r (i.e., $ret \notin H'$), then all the upper-bound obligations in O_{ret} are propagated as live obligations at the next position $i_c + 1$ (note that the *MAP* visiting $i_c + 1$ starts at $i_c + 1$, terminates at $i_r - 1$, and does not satisfy proposition p_{∞}). The push transitions of point 2 consider instead the case where $\mathcal{A}_{y^{a}_{b}}$ guesses that the current call position i_c has no matching return i_r, i.e., i_c is the last position of the current *MAP*. In this case, $\mathcal{A}_{y^{a}_{b}}$ pushes the symbol *bad* on the stack and the transition relation is consistently updated.

Internal transitions for each internal transition $q \xrightarrow{a,\theta\wedge y^{a}_{b}\in I,Res} q'$ of $\mathcal{A}$, we add the internal transitions $(q, O, H) \xrightarrow{a,\theta\wedge con(O,a),Res\cup Res'} (q', O', H')$, where $b = a$ iff $b \in H$, and

1. Case $ret \in H'$: $int \in H$, $I = \{\vdash\}$, $Res' = \emptyset$, ($\diamond^{a}b \in H$ iff $b = a$), and $b \neq a$ implies $O = live(O)$;
2. Case $ret \notin H'$: $Abs((O, H), a, y^{a}_{b} \in I, Res', (O', H'))$ holds.

In the former case, $\mathcal{A}_{y^{a}_{b}}$ guesses that the current position i is the last one of the current *MAP*($ret \in H'$); in the later, the current *MAP* visits the next non-return position $i + 1$. Note that if $b = a$, the obligations in the current state are checked by the constraint $con(O, a)$.

Pop transitions for each pop transition $q \xrightarrow{a,\theta\wedge y^{a}_{b}\in I,Res,pop(\gamma)} q' \in \Delta_r$, we have the pop transitions $(q, O, H) \xrightarrow{a,\theta\wedge con(O,a),Res\cup Res',pop(\gamma')} (q', O', H')$, where $b = a$ iff $b \in H$, and

1. Case $\gamma \neq \top$: $ret \in H$ and $\gamma' = (\gamma, (O, H))$; if $ret \notin H'$, then $Abs((O, H), a, y^{a}_{b} \in I, Res', (O', H'))$; otherwise, $I = \{\vdash\}$, $Res' = \emptyset$, ($\diamond^{a}b \in H$ iff $b = a$), and $b \neq a$ implies $O = live(O)$;
2. Case $\gamma = \top$: $ret \in H$, $O = \emptyset$, $\gamma' = \top$, $p_{\infty} \in H$, and $p_{\infty} \in H'$; if $ret \notin H'$, then $Abs((O, H), a, y^{a}_{b} \in I, Res', (O', H'))$; otherwise, $I = \{\vdash\}$, $Res' = \emptyset$, $O' = \emptyset$, and ($\diamond^{a}b \in H$ iff $b = a$).

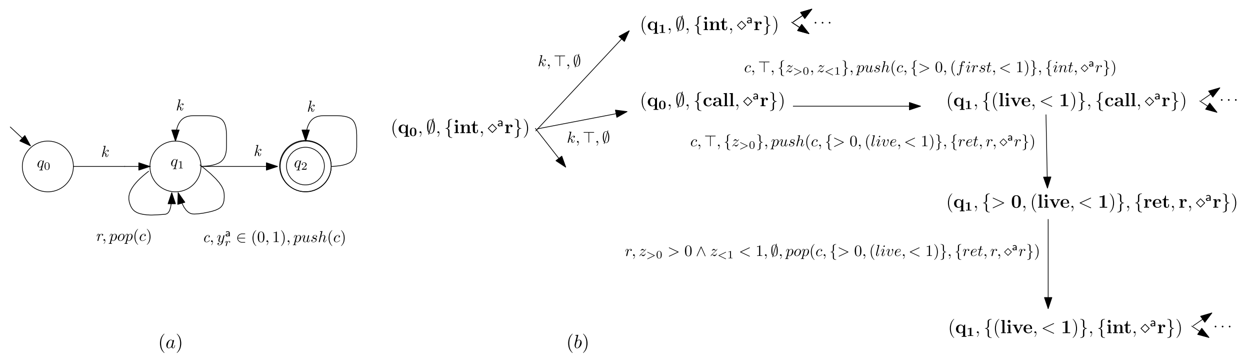
If $\gamma \neq \top$, then the current return position has a matched-call. Thus, $\mathcal{A}_{y^{a}_{b}}$ pops from the stack γ together with an obligation set and a check set, and verifies that the last two sets correspond to the ones associated with the current control state. If $\gamma = \top$, then the current position is also the initial position of the associated *MAP*. For an example see Fig. 4.

Finally, the generalized Büchi condition $\mathcal{F}'$ of $\mathcal{A}_{y^{a}_{b}}$ is defined as follows. For each Büchi component F of $\mathcal{A}$, $\mathcal{A}_{y^{a}_{b}}$ has the Büchi component consisting of the states (q, O, H) such that $q \in F$. Moreover, $\mathcal{A}_{y^{a}_{b}}$ has an additional component consisting of the states (q, O, H) such that $p_{\infty} \in H$, and either $\diamond^{a}b \notin H$ or $b \in H$. Such a component ensures that the guesses about the matched calls are correct (p_{∞} occurs infinitely often), and that the liveness requirement b of $\diamond^{a}b$ is fulfilled whenever $\diamond^{a}b$ is asserted at a position of an infinite *MAP*. By construction, $\mathcal{A}$ and $\mathcal{A}_{y^{a}_{b}}$ accept the same language. $\square$

By closure properties and Theorem 5, we can state the main result of the Section.

Theorem 6. *Emptiness, universality, and language inclusion for Büchi ECNA, and visibly model-checking against Büchi ECNA are* EXPTIME*-complete.*

Proof. For the upper bounds, first observe that by [16,1] the emptiness problem of generalized Büchi VPTA is EXPTIME-complete and solvable in time $O(n^4 \cdot 2^{O(m \cdot \log Km)})$, where n is the number of states, m is the number of clocks, and K is the largest constant used in the clock constraints of the automaton (hence, the time complexity is polynomial in the number of states). Now, given two Büchi ECNA $\mathcal{A}_1$ and $\mathcal{A}_2$ over Σ, checking whether $\mathcal{L}_T(\mathcal{A}_1) \subseteq \mathcal{L}_T(\mathcal{A}_2)$ reduces to check emptiness of



q_0
k
q_1
k
q_2
k
k
$r, pop(c)$
$c, y^{\mathsf{a}}_r \in (0,1), push(c)$
(a)
$(\mathbf{q_0}, \emptyset, \{\mathbf{int}, \diamond^{\mathsf{a}}\mathbf{r}\})$
$k, \top, \emptyset$
$k, \top, \emptyset$
$(\mathbf{q_1}, \emptyset, \{\mathbf{int}, \diamond^{\mathsf{a}}\mathbf{r}\})$
$(\mathbf{q_0}, \emptyset, \{\mathbf{call}, \diamond^{\mathsf{a}}\mathbf{r}\})$
$c, \top, \{z_{>0}, z_{<1}\}, push(c, \{>0, (first, <1)\}, \{int, \diamond^{\mathsf{a}}r\})$
$(\mathbf{q_1}, \{(\mathbf{live}, <\mathbf{1})\}, \{\mathbf{call}, \diamond^{\mathsf{a}}\mathbf{r}\})$
$c, \top, \{z_{>0}\}, push(c, \{>0, (live, <1)\}, \{ret, r, \diamond^{\mathsf{a}}r\})$
$(\mathbf{q_1}, \{>\mathbf{0}, (\mathbf{live}, <\mathbf{1})\}, \{\mathbf{ret}, \mathbf{r}, \diamond^{\mathsf{a}}\mathbf{r}\})$
$r, z_{>0} > 0 \wedge z_{<1} < 1, \emptyset, pop(c, \{>0, (live, <1)\}, \{ret, r, \diamond^{\mathsf{a}}r\})$
$(\mathbf{q_1}, \{(\mathbf{live}, <\mathbf{1})\}, \{\mathbf{int}, \diamond^{\mathsf{a}}\mathbf{r}\})$
(b)

Fig. 4. A fragment of the VPTA $\mathcal{A}_{y^{\mathsf{a}}_b}$ (figure (b)) obtained from the removal of the abstract predicting clock y^{a}_b from the ECNA $\mathcal{A}$ depicted on figure (a). The initial fragment of $\mathcal{A}_{y^{\mathsf{a}}_b}$ shows a path starting with an internal transition, followed by two call transitions and a return.

the language $\mathcal{L}_T(\mathcal{A}_1) \cap \overline{\mathcal{L}_T(\mathcal{A}_2)}$. Similarly, given a Büchi VPTA $\mathcal{S}$ where all the states are accepting and a Büchi ECNA $\mathcal{A}$ over the same pushdown alphabet Σ, model-checking $\mathcal{S}$ against $\mathcal{A}$ reduces to check emptiness of the language $\mathcal{L}_T(\mathcal{S}) \cap \overline{\mathcal{L}_T(\mathcal{A})}$. Since Büchi VPTA are polynomial-time closed under intersection and universality can be reduced in linear-time to language inclusion, by the closure properties of Büchi ECNA (Proposition 1 and Theorem 2) and Theorem 5, membership in EXPTIME for the considered problems directly follows. For the matching lower-bounds, the proof of EXPTIME-hardness for emptiness of Büchi VPTA can be easily adapted to the class of Büchi ECNA. For the other problems, the result directly follows from EXPTIME-hardness of the corresponding problems for Büchi VPA [7,8] which are subsumed by Büchi ECNA. □

5. The Event-Clock Nested Temporal Logic

In this section we study the linear temporal logic counterpart of ECNA. Similarly to the case of automata setting, we define a linear temporal logic which compose real-time and context-free properties.

The logical framework related to the class of Event-Clock automata (ECA) is the so called *Event-Clock Temporal Logic* (EC_TL) [33]. EC_TL is a decidable extension of standard LTL with past obtained by means of two indexed modal operators $\lhd_I$ and $\rhd_I$, with I an interval, which express real-time constraints. The intuition is that $\rhd_I \varphi$ holds true at a given position i of a timed word w, if the timed elapsed to reach the least position $j \geq i$ of w where ϕ holds belongs to the interval I (and symmetrically for the past modality $\lhd_I$). As for the class of VPA, a related logical framework is the temporal logic of nested calls and returns CaRet [3]. It is a well-known context-free extension of LTL with past by means of non-regular variants of the standard LTL temporal future operators next $\bigcirc$ and until U and the past operators previous $\ominus$ and S. While the standard operator are interpreted over words, the *abstract* version of this operators (i.e. $\bigcirc^{\mathsf{a}}$, U^{a}, $\ominus^{\mathsf{a}}$ and S^{a}) are interpreted over abstract paths (i.e. *MAP*s) allowing to express context free properties. A variant of the past operators also is interpreted over call paths (i.e. $\ominus^{\mathsf{c}}$ and S^{c}). The two sets of variants allow us to express context free properties.

In this section, we introduce an extension of both EC_TL and CaRet, called *Event-Clock Nested Temporal Logic* (EC_NTL) able to specify non-regular context-free real-time properties. We assume to have a set $\mathcal{P}$ of atomic propositions containing the special propositions *call*, *ret*, and *int*. The syntax of EC_NTL formulas φ is as follows:

$$\varphi := \top \mid p \mid \varphi \vee \varphi \mid \neg\varphi \mid \bigcirc^{dir}\varphi \mid \ominus^{dir'}\varphi \mid \varphi\,\mathsf{U}^{dir}\varphi \mid \varphi\,\mathsf{S}^{dir'}\varphi \mid \rhd^{dir}_I\varphi \mid \lhd^{dir'}_I\varphi$$

where $p \in \mathcal{P}$, I is an interval in $\mathbb{R}_+$ with bounds in $\mathbb{N} \cup \{\infty\}$, $dir \in \{\mathsf{g}, \mathsf{a}\}$, and $dir' \in \{\mathsf{g}, \mathsf{a}, \mathsf{c}\}$. The operators $\bigcirc^{\mathsf{g}}$, $\ominus^{\mathsf{g}}$, U^{g}, and S^{g} are the standard 'next', 'previous', 'until', and 'since' LTL modalities, respectively, $\bigcirc^{\mathsf{a}}$, $\ominus^{\mathsf{a}}$, U^{a}, and S^{a} are their non-regular abstract versions, and $\ominus^{\mathsf{c}}$ and S^{c} are the non-regular caller versions of the 'previous' and 'since' LTL modalities. Intuitively, the abstract and caller modalities specify LTL requirements on the abstract and caller paths of the given timed word over $\Sigma_{\mathcal{P}}$. Real-time constraints are specified by the indexed operators $\rhd^{\mathsf{g}}_I$, $\lhd^{\mathsf{g}}_I$, $\rhd^{\mathsf{a}}_I$, $\lhd^{\mathsf{a}}_I$, and $\lhd^{\mathsf{c}}_I$. The formula $\rhd^{\mathsf{g}}_I\varphi$ requires that the delay t before the next position where φ holds satisfies $t \in I$; symmetrically, $\lhd^{\mathsf{g}}_I\varphi$ constraints the previous position where φ holds. The abstract versions $\rhd^{\mathsf{a}}_I\varphi$ and $\lhd^{\mathsf{a}}_I\varphi$ are similar, but the notions of next and previous position where φ holds refer to the *MAP* visiting the current position. Analogously, for the caller version $\lhd^{\mathsf{c}}_I\varphi$ of $\lhd^{\mathsf{g}}_I\varphi$, the notion of previous position where φ holds refers to the caller path visiting the current position.

Full CaRet [3] corresponds to the fragment of EC_NTL obtained by disallowing the real-time operators, while the logic EC_TL [33] is obtained from EC_NTL by disallowing the abstract and caller modalities. As pointed out in [33], the real-time operators $\lhd$ and $\rhd$ generalize the semantics of event clock variables since they allow recursion, i.e., they can constraint arbitrary formulas and not only atomic propositions. Accordingly, the *non-recursive fragment* of EC_NTL is obtained by replacing the clauses $\rhd^{dir}_I\varphi$ and $\lhd^{dir'}_I\varphi$ in the syntax with the clauses $\rhd^{dir}_I p$ and $\lhd^{dir'}_I p$, where $p \in \mathcal{P}$. We use standard shortcuts in EC_NTL: the formula $\diamond^{\mathsf{g}}\psi$ stands for $\top\,\mathsf{U}^{\mathsf{g}}\psi$ (the standard LTL operator), and $\square^{\mathsf{g}}\psi$ stands for $\neg\diamond^{\mathsf{g}}\neg\psi$ (the LTL always operator). For an EC_NTL formula φ, $|\varphi|$ denotes the number of distinct subformulas of φ and $Const_\varphi$ the set of constants used as finite endpoints in the intervals associates with the real-time modalities. The size of φ is $|\varphi| + k$, where k is the size of the binary encoding of the largest constant in $Const_\varphi$.

The set $\mathcal{P}$ induces a pushdown alphabet $\Sigma_\mathcal{P} = \Sigma_{call} \cup \Sigma_{ret} \cup \Sigma_{int}$, where $\Sigma_{call} = \{P \subseteq \mathcal{P} \mid P \cap \{call, ret, int\} = \{call\}\}$, $\Sigma_{ret} = \{P \subseteq \mathcal{P} \mid P \cap \{call, ret, int\} = \{ret\}\}$, and $\Sigma_{int} = \{P \subseteq \mathcal{P} \mid P \cap \{call, ret, int\} = \{int\}\}$. Given an EC_NTL formula φ, a timed word $w = (\sigma, \tau)$ over $\Sigma_\mathcal{P}$ and a position $0 \leq i < |w|$, the satisfaction relation $(w, i) \models \varphi$ is inductively defined as follows (we omit the clauses for the atomic propositions and Boolean connectives which are standard):

$$
\begin{array}{lcl}
(w,i) \models \bigcirc^{dir}\varphi & \Leftrightarrow & \text{there is } j > i \text{ such that } j = \mathsf{succ}(dir, \sigma, i) \text{ and } (w,j) \models \varphi \\
(w,i) \models \ominus^{dir'}\varphi & \Leftrightarrow & \text{there is } j < i \text{ such that } (w,j) \models \varphi \text{ and } either\ (dir' \neq \mathsf{c} \text{ and} \\
 & & i = \mathsf{succ}(dir', \sigma, j)),\ \text{or } (dir' = \mathsf{c} \text{ and } j = \mathsf{succ}(\mathsf{c}, \sigma, i)) \\
(w,i) \models \varphi_1 \mathsf{U}^{dir} \varphi_2 & \Leftrightarrow & \text{there is } j \geq i \text{ such that } j \in Pos(dir, \sigma, i),\ (w,j) \models \varphi_2 \text{ and} \\
 & & (w,k) \models \varphi_1 \text{ for all } k \in [i, j-1] \cap Pos(dir, \sigma, i) \\
(w,i) \models \varphi_1 \mathsf{S}^{dir'} \varphi_2 & \Leftrightarrow & \text{there is } j \leq i \text{ such that } j \in Pos(dir', \sigma, i),\ (w,j) \models \varphi_2 \text{ and} \\
 & & (w,k) \models \varphi_1 \text{ for all } k \in [j+1, i] \cap Pos(dir', \sigma, i) \\
(w,i) \models \rhd_I^{dir}\varphi & \Leftrightarrow & \text{there is } j > i \text{ s.t. } j \in Pos(dir, \sigma, i),\ (w,j) \models \varphi,\ \tau_j - \tau_i \in I, \\
 & & \text{and } (w,k) \not\models \varphi \text{ for all } k \in [i+1, j-1] \cap Pos(dir, \sigma, i) \\
(w,i) \models \lhd_I^{dir'}\varphi & \Leftrightarrow & \text{there is } j < i \text{ s.t. } j \in Pos(dir', \sigma, i),\ (w,j) \models \varphi,\ \tau_i - \tau_j \in I, \\
 & & \text{and } (w,k) \not\models \varphi \text{ for all } k \in [j+1, i-1] \cap Pos(dir', \sigma, i)
\end{array}
$$

with $dir \in \{\mathsf{g}, \mathsf{a}\}$, and $dir' \in \{\mathsf{g}, \mathsf{a}, \mathsf{c}\}$. A timed word w satisfies a formula φ (we also say that w is a model of φ) if $(w, 0) \models \varphi$. The timed language $\mathcal{L}_T(\varphi)$ (resp. ω-timed language $\mathcal{L}_T^\omega(\varphi)$) of φ is the set of finite (resp., infinite) timed words over $\Sigma_\mathcal{P}$ satisfying φ. As shown in Example 4, EC_NTL allows one to express in a natural way real-time LTL-like properties over non-regular patterns capturing local computations of procedures or the stack content.

Example 4. We consider three relevant examples.

- *Real-time total correctness:* a bounded-time total correctness requirement for a procedure A specifies that if the pre-condition p holds when the procedure A is invoked, then the procedure must return within k time units and q must hold upon return. Such a requirement can be expressed by the following non-recursive formula, where proposition p_A characterizes calls to procedure A: $\square^\mathsf{g}\big((call \wedge p \wedge p_A) \rightarrow (\bigcirc^\mathsf{a} q \wedge \rhd^\mathsf{a}_{[0,k]} ret)\big)$.
- *Local bounded-time response properties:* the requirement that in the local computation (abstract path) of a procedure A, every request p is followed by a response q within k time units can be expressed by the following non-recursive formula, where c_A denotes that the control is inside procedure A: $\square^\mathsf{g}\big((p \wedge c_A) \rightarrow \rhd^\mathsf{a}_{[0,k]} q\big)$.
- *Real-time properties over the stack content:* the real-time security requirement that a procedure A is invoked only if procedure B belongs to the call stack and within k time units since the activation of B can be expressed as follows (the calls to procedure A and B are marked by proposition p_A and p_B, respectively): $\square^\mathsf{g}\big((call \wedge p_A) \rightarrow \lhd^\mathsf{c}_{[0,k]} p_B\big)$.

Expressiveness results. We now compare the expressive power of the formalisms EC_NTL, ECNA, and VPTA with respect to the associated classes of (ω-)timed languages. It is known that ECA and the logic EC_TL are expressively incomparable [33]. This result trivially generalizes to ECNA and EC_NTL. In fact, to embed incomparability it suffices to consider (ω-)timed languages over $\Sigma_\mathcal{P} = \Sigma_{call} \cup \Sigma_{int} \cup \Sigma_{ret}$ with $\Sigma_{call} = \emptyset$ and $\Sigma_{ret} = \emptyset$ (i.e. timed words consisting only of internal actions). Any ECNA under this restricted alphabet is actually an ECA, and symmetrically the logic EC_NTL corresponds to EC_TL. As a consequence, under this kind of alphabets ECNA and EC_NTL are incomparable, proving the incomparability of the formalisms. Moreover, in Theorem 4 we have shown that ECNA are strictly less expressive than VPTA. In Section 5.1, we show that EC_NTL is subsumed by VPTA (in particular, every EC_NTL formula can be translated into an equivalent VPTA). The inclusion is strict since the logic EC_NTL is closed under complementation, while VPTA are not [26]. Hence, we can state the following expressiveness results.

Theorem 7. *Over finite (resp., infinite) timed words, EC_NTL and ECNA are expressively incomparable, and EC_NTL is strictly less expressive than VPTA.*

In addition, we consider the expressiveness of the novel timed temporal modalities $\lhd^\mathsf{a}_I$, $\rhd^\mathsf{a}_I$, and $\lhd^\mathsf{c}_I$. We prove that these modalities add expressive power.

Theorem 8. *Let $\mathcal{F}$ be the fragment of EC_NTL obtained by disallowing the modalities $\lhd^\mathsf{a}_I$, $\lhd^\mathsf{c}_I$, and $\rhd^\mathsf{a}_I$. Then, $\mathcal{F}$ is strictly less expressive than EC_NTL.*

Proof. We focus on the case of finite timed words (the case of infinite timed words is similar). Let $\mathcal{P} = \{call, ret\}$ and $\mathcal{L}_T$ be the timed language consisting of the finite timed words of the form (σ, τ) such that σ is a well-matched word of the form

$\{call\}^n \cdot \{ret\}^n$ for some $n > 0$, and there is a call position i_c of σ such that $\tau_{i_r} - \tau_{i_c} = 1$, where i_r is the matching-return of i_c in σ. $\mathcal{L}_T$ can be easily expressed in EC_NTL. On the other hand, one can show that $\mathcal{L}_T$ is not definable in $\mathcal{F}$ (a proof is given in Appendix C).

5.1. Decision procedures for the logic EC_NTL

In this section we consider the following decision problems for the logic EC_NTL:

- *Satisfiability:* does a given EC_NTL formula have a finite (resp., infinite) model?
- *Visibly model-checking:* given a *visibly pushdown timed system* $\mathcal{S}$ over $\Sigma_{\mathcal{P}}$ (that is a VPTA where all the states are accepting) and an EC_NTL formula φ over $\mathcal{P}$, does $\mathcal{L}_T(\mathcal{S}) \subseteq \mathcal{L}_T(\varphi)$ (resp., $\mathcal{L}^{\omega}_T(\mathcal{S}) \subseteq \mathcal{L}^{\omega}_T(\varphi)$) hold?

As for satisfiability and visibly model-checking for EC_NTL, we follow an automata-theoretic approach which generalizes both the automatic-theoretic approach of CaRet [3] and the one for EC_TL [33]. We focus on infinite timed words (the case of finite timed words is similar). Given an EC_NTL formula φ over $\mathcal{P}$, we construct in singly exponential time a generalized Büchi ECNA $\mathcal{A}_{\varphi}$ over an extension of the pushdown alphabet $\Sigma_{\mathcal{P}}$ accepting suitable encodings of the infinite models of φ. In this way, the problem of model checking a *visibly pushdown timed system* $\mathcal{S}$ against a property expressed by an EC_NTL formula can be reduced to the problem of visibly model checking against Büchi ECNA investigated in Section 4 (see Theorem 6).

Fix an EC_NTL formula φ over $\mathcal{P}$. For each infinite timed word $w = (\sigma, \tau)$ over $\Sigma_{\mathcal{P}}$ we associate to w an infinite timed word $\pi = (\sigma_e, \tau)$ over an extension of $\Sigma_{\mathcal{P}}$, called *fair Hintikka sequence*, where $\sigma_e = A_0 A_1 \ldots$, and for all $i \geq 0$, A_i is an *atom* which, intuitively, describes a maximal set of subformulas of φ which hold at position i along w. The notion of *atom* syntactically captures the semantics of the Boolean connectives and the local fixpoint characterization of the variants of until (resp., since) modalities in terms of the corresponding variants of the next (resp., previous) modalities. Additional requirements on the timed word π, which can be easily checked by the transition function of an ECNA, capture the semantics of the variants of next and previous modalities, and the semantics of the real-time operators. Finally, the global *fairness* requirement, which can be easily checked by a standard generalized Büchi acceptance condition, captures the liveness requirements ψ_2 in until subformulas of the form $\psi_1 \mathsf{U}^{\mathsf{g}} \psi_2$ (resp., $\psi_1 \mathsf{U}^{\mathsf{a}} \psi_2$) of φ. In particular, when an abstract until formula $\psi_1 \mathsf{U}^{\mathsf{a}} \psi_2$ is asserted at position i along an infinite timed word w over $\Sigma_{\mathcal{P}}$ and the *MAP* ν visiting position i is infinite, we have to ensure that the liveness requirement ψ_2 holds at some position $j \geq i$ of the *MAP* ν. To this end, we use a special proposition p_{∞} which *does not* hold at a position i of w iff i has a caller whose matching return is defined. We now proceed with the technical details. The closure $\mathsf{Cl}(\varphi)$ of φ is the smallest set containing:

- any proposition $p \in \mathcal{P} \cup \{p_{\infty}\}$, formulas $\top$, $\bigcirc^{\mathsf{a}}\top$, $\ominus^{\mathsf{a}}\top$ and all the subformulas of φ;
- the formulas $\bigcirc^{dir}(\psi_1 \mathsf{U}^{dir} \psi_2)$ (resp., $\ominus^{dir'}(\psi_1 \mathsf{S}^{dir'} \psi_2)$) for all the subformulas $\psi_1 \mathsf{U}^{dir} \psi_2$ (resp., $\psi_1 \mathsf{S}^{dir'} \psi_2$) of φ, where $dir \in \{\mathsf{g}, \mathsf{a}\}$ (resp., $dir' \in \{\mathsf{g}, \mathsf{a}, \mathsf{c}\}$);
- all the negations of the above formulas (we identify $\neg\neg\psi$ with ψ).

Note that $\varphi \in \mathsf{Cl}(\varphi)$ and $|\mathsf{Cl}(\varphi)| = O(|\varphi|)$. In the following, elements of $\mathsf{Cl}(\varphi)$ are seen as atomic propositions, and we consider the pushdown alphabet $\Sigma_{\mathsf{Cl}(\varphi)}$ induced by $\mathsf{Cl}(\varphi)$. In particular, for a timed word π over $\Sigma_{\mathsf{Cl}(\varphi)}$, we consider the clock valuation val^{π}_i specifying the values of the event clocks x_{ψ}, y_{ψ}, x^{a}_{ψ}, y^{a}_{ψ}, and x^{c}_{ψ} at position i along π, where $\psi \in \mathsf{Cl}(\varphi)$. An *atom* A of φ is a subset of $\mathsf{Cl}(\varphi)$ satisfying the following constraints:

- A is a maximal subset of $\mathsf{Cl}(\varphi)$ which is propositionally consistent, i.e.:
  - $\top \in A$ and for each $\psi \in \mathsf{Cl}(\varphi)$, $\psi \in A$ iff $\neg\psi \notin A$;
  - for each $\psi_1 \vee \psi_2 \in \mathsf{Cl}(\varphi)$, $\psi_1 \vee \psi_2 \in A$ iff $\{\psi_1, \psi_2\} \cap A \neq \emptyset$;
  - A contains exactly one atomic proposition in $\{call, ret, int\}$;
- for all $dir \in \{\mathsf{g}, \mathsf{a}\}$ and $\psi_1 \mathsf{U}^{dir} \psi_2 \in \mathsf{Cl}(\varphi)$, either $\psi_2 \in A$ or $\{\psi_1, \bigcirc^{dir}(\psi_1 \mathsf{U}^{dir} \psi_2)\} \subseteq A$;
- for all $dir' \in \{\mathsf{g}, \mathsf{a}, \mathsf{c}\}$ and $\psi_1 \mathsf{S}^{dir'} \psi_2 \in \mathsf{Cl}(\varphi)$, either $\psi_2 \in A$ or $\{\psi_1, \ominus^{dir'}(\psi_1 \mathsf{S}^{dir'} \psi_2)\} \subseteq A$;
- if $\bigcirc^{\mathsf{a}}\top \notin A$, then for all $\bigcirc^{\mathsf{a}}\psi \in \mathsf{Cl}(\varphi)$, $\bigcirc^{\mathsf{a}}\psi \notin A$;
- if $\ominus^{\mathsf{a}}\top \notin A$, then for all $\ominus^{\mathsf{a}}\psi \in \mathsf{Cl}(\varphi)$, $\ominus^{\mathsf{a}}\psi \notin A$.

We introduce the notion of Hintikka sequence π which corresponds to an infinite timed word over $\Sigma_{\mathsf{Cl}(\varphi)}$ satisfying additional constraints. These constraints capture the semantics of the variants of next, previous, and real-time modalities, and (partially) the intended meaning of proposition p_{∞} along the associated timed word over $\Sigma_{\mathcal{P}}$ (the projection of π over $\Sigma_{\mathcal{P}} \times \mathbb{R}_+$). For an atom A, let $Caller(A)$ be the set of caller formulas $\ominus^{\mathsf{c}}\psi$ in A. For atoms A and A', we define a predicate $Next(A, A')$ which holds if the global next (resp., global previous) requirements in A (resp., A') are the ones that hold in A' (resp., A), i.e.: (i) for all $\bigcirc^{\mathsf{g}}\psi \in \mathsf{Cl}(\varphi)$, $\bigcirc^{\mathsf{g}}\psi \in A$ iff $\psi \in A'$, and (ii) for all $\ominus^{\mathsf{g}}\psi \in \mathsf{Cl}(\varphi)$, $\ominus^{\mathsf{g}}\psi \in A'$ iff $\psi \in A$.
Similarly, the predicate $AbsNext(A, A')$ holds if: (i) for all $\bigcirc^{\mathsf{a}}\psi \in \mathsf{Cl}(\varphi)$, $\bigcirc^{\mathsf{a}}\psi \in A$ iff $\psi \in A'$, and (ii) for all $\ominus^{\mathsf{a}}\psi \in \mathsf{Cl}(\varphi)$,

$\ominus^{\mathsf{a}}\psi \in A'$ iff $\psi \in A$, and additionally (iii) $Caller(A) = Caller(A')$. Note that for $AbsNext(A, A')$ to hold we also require that the caller requirements in A and A' coincide consistently with the fact that the positions of a *MAP* have the same caller (if any).

Definition 3. An infinite timed word $\pi = (\sigma, \tau)$ over $\Sigma_{\mathsf{Cl}(\varphi)}$, where $\sigma = A_0 A_1 \ldots$, is an *Hintikka sequence of* φ, if for all $i \geq 0$, A_i is a φ-atom and the following holds:

1. *Initial consistency:* for all $dir \in \{\mathsf{g}, \mathsf{a}, \mathsf{c}\}$ and $\ominus^{dir}\psi \in \mathsf{Cl}(\varphi)$, $\neg\ominus^{dir}\psi \in A_0$;
2. *Global next and previous requirements:* $Next(A_i, A_{i+1})$;
3. *Abstract and caller requirements:* we distinguish three cases
   - $call \notin A_i$ and $ret \notin A_{i+1}$: $AbsNext(A_i, A_{i+1})$, ($p_\infty \in A_i$ iff $p_\infty \in A_{i+1}$);
   - $call \notin A_i$ and $ret \in A_{i+1}$: $\bigcirc^{\mathsf{a}}\top \notin A_i$, and ($\ominus^{\mathsf{a}}\top \in A_{i+1}$ iff the matching call of the return position $i+1$ is defined); Moreover, if $\ominus^{\mathsf{a}}\top \notin A_{i+1}$, then $p_\infty \in A_i \cap A_{i+1}$ and $Caller(A_{i+1}) = \emptyset$;
   - $call \in A_i$: if $\mathsf{succ}(\mathsf{a}, \sigma, i) = \vdash$ then $\bigcirc^{\mathsf{a}}\top \notin A_i$ and $p_\infty \in A_i$; otherwise $AbsNext(A_i, A_j)$ and ($p_\infty \in A_i$ iff $p_\infty \in A_j$), where $j = \mathsf{succ}(\mathsf{a}, \sigma, i)$. Moreover, if $ret \notin A_{i+1}$, then $Caller(A_{i+1}) = \{\ominus^{\mathsf{c}}\psi \in \mathsf{Cl}(\varphi) \mid \psi \in A_i\}$ and ($\bigcirc^{\mathsf{a}}\top \in A_i$ iff $p_\infty \notin A_{i+1}$).
4. *Real-time requirements:*
   - for all $dir \in \{\mathsf{g}, \mathsf{a}, \mathsf{c}\}$ and $\lhd_I^{dir}\psi \in \mathsf{Cl}(\varphi)$, $\lhd_I^{dir}\psi \in A_i$ iff $val_i^{\pi}(x_\psi^{dir}) \in I$;
   - for all $dir \in \{\mathsf{g}, \mathsf{a}\}$ and $\rhd_I^{dir}\psi \in \mathsf{Cl}(\varphi)$, $\rhd_I^{dir}\psi \in A_i$ iff $val_i^{\pi}(y_\psi^{dir}) \in I$.

In order to capture the liveness requirements of the global and abstract until subformulas of φ, and fully capture the intended meaning of proposition p_∞, we consider the following additional global fairness constraint. An Hintikka sequence $\pi = (A_0, t_0)(A_1, t_1)$ of φ is *fair* if

- for infinitely many $i \geq 0$, $p_\infty \in A_i$;
- for all $\psi_1 \mathsf{U}^{\mathsf{g}} \psi_2 \in \mathsf{Cl}(\varphi)$, there are infinitely many $i \geq 0$ s.t. $\{\psi_2, \neg(\psi_1 \mathsf{U}^{\mathsf{g}} \psi_2)\} \cap A_i \neq \emptyset$, and for all $\psi_1 \mathsf{U}^{\mathsf{a}} \psi_2 \in \mathsf{Cl}(\varphi)$, there are infinitely many $i \geq 0$ such that $p_\infty \in A_i$ and $\{\psi_2, \neg(\psi_1 \mathsf{U}^{\mathsf{a}} \psi_2)\} \cap A_i \neq \emptyset$.

The Hintikka sequence π is *initialized* if $\varphi \in A_0$. Note that according to the intended meaning of proposition p_∞, for each infinite timed word $w = (\sigma, \tau)$ over $\Sigma_{\mathcal{P}}$, p_∞ holds at infinitely many positions. Moreover, along σ, there is at a most one *infinite MAP* ν, and for such a *MAP* ν and each position i greater than the starting position of ν, either i belongs to ν and p_∞ holds, or p_∞ does not hold. Hence, the fairness requirement for an abstract until subformula $\psi_1 \mathsf{U}^{\mathsf{a}} \psi_2$ of φ ensures that whenever $\psi_1 \mathsf{U}^{\mathsf{a}} \psi_2$ is asserted at some position i of ν, then ψ_2 eventually holds at some position $j \geq i$ along ν. We denote by $Proj_\varphi$ the mapping associating to each *fair* Hintikka sequence $\pi = (A_0, t_0)(A_1, t_1) \ldots$ of φ the infinite timed word over $\Sigma_{\mathcal{P}}$ $Proj(\pi) = (A_0 \cap \mathcal{P}, t_0)(A_1 \cap \mathcal{P}, t_1) \ldots$.

The properties of fair Hintikka sequence of an EC_NTL formula expressed by Lemma 9 and Lemma 10 provide a characterization of the infinite models of φ.

Lemma 9. *Let $\pi = (A_0, t_0)(A_1, t_1)$ be a fair Hintikka sequence of an EC_NTL formula φ and $\sigma = A_0 A_1 \ldots$. Then, for all $i \geq 0$, the following holds:*

1. *$p_\infty \notin A_i$ iff i has a caller whose matching return exists;*
2. *for all $\psi \in Cl(\varphi) \setminus \{p_\infty, \neg p_\infty\}$, $\psi \in A_i$ iff $(Proj_\varphi(\pi), i) \models \psi$.*

Proof. Let $\pi = (A_0, t_0)(A_1, t_1)$ be a fair Hintikka sequence of φ, $\sigma = A_0 A_1 \ldots$, and P_f be the set of positions $i \geq 0$ such that i has a caller in σ with existing matching return.
Proof of Property 1: let $i \geq 0$ and ν be the *MAP* of σ visiting position i. We need to show that $p_\infty \notin A_i$ iff $i \in P_f$. By Property 3 in Definition 3, *either* for all positions j visited by ν, $p_\infty \in A_j$, *or* for all positions j visited by ν, $p_\infty \notin A_j$. We distinguish the following cases:

1. ν is finite and leads to an unmatched call: hence, for all positions j visited by ν, $j \notin P_f$. Since π is an Hintikka sequence, by Property 3 in Definition 3, ν visits only positions j where $p_\infty \in A_j$, and the result follows.
2. ν is finite and leads to a non-call position k such that $k+1$ is a return position. If $k+1$ has no matched call, then for all positions j visited by ν, $j \notin P_f$. Moreover, by Property 3 in Definition 3, $p_\infty \in A_k \cap A_{k+1}$. Hence, ν visits only positions j where $p_\infty \in A_j$, and the result follows. Now, assume that $k+1$ has a matched call i_c. This means that ν starts at $i_c + 1$ and for all positions j visited by ν, $j \in P_f$. By Property 3 in Definition 3, $p_\infty \notin A_{i_c+1}$. Hence, ν visits only positions j where $p_\infty \notin A_j$, and the result follows in this case as well.

3. ν is infinite: hence, for all positions j visited by ν, $j \notin P_f$. By definition of abstract path, ν is the unique infinite *MAP* of σ, and there is $k \geq 0$ such that for all $m \geq k$, either m is visited by ν (hence, $m \notin P_f$), or $m \in P_f$. By the previous case, if $m \in P_f$, then $p_\infty \notin A_m$. Since π is fair, for infinitely many $h \geq 0$, $p_\infty \in A_h$. Thus, we deduce that for all positions j visited by ν, $p_\infty \in A_j$, and the result follows.

Proof of Property 2: let $i \geq 0$ and $\psi \in \mathsf{Cl}(\varphi) \setminus \{p_\infty, \neg p_\infty\}$. We prove by induction on the structure of ψ that $\psi \in A_i$ *iff* $(\mathit{Proj}_\varphi(\pi), i) \models \psi$. Here, we focus on the cases where the root (in the syntactic tree) modality of ψ is either U^{a} or $\rhd^{\mathsf{a}}_I$. The other cases are similar or simpler. The base case is trivial.

1. $\psi = \psi_1 \mathsf{U}^{\mathsf{a}} \psi_2$: first, assume that $(\mathit{Proj}_\varphi(\pi), i) \models \psi$. Hence, there exists an infix of the *MAP* of σ visiting i of the form $j_0 < j_1 \ldots < j_n$ such that $j_0 = i$, $(\mathit{Proj}_\varphi(\pi), j_n) \models \psi_2$ and $(\mathit{Proj}_\varphi(\pi), j_k) \models \psi_1$ for all $0 \leq k < n$. By the induction hypothesis, $\psi_2 \in A_{j_n}$ and $\psi_1 \in A_{j_k}$ for all $0 \leq k < n$. Since π is an Hintikka sequence, by definition of atom and Property 3 in Definition 3, it follows that $\psi_1 \mathsf{U}^{\mathsf{a}} \psi_2 \in A_{j_h}$ for all $0 \leq h \leq n$. Hence, being $i = j_0$, we obtain that $\psi \in A_i$ and the result follows. Now assume that $\psi \in A_i$. We need to show that $(\mathit{Proj}_\varphi(\pi), i) \models \psi$. Let ν be the *MAP* of σ visiting position i. Assume that ν is infinite (the other case being simpler). Let $\nu^i = j_0 < j_1 \ldots$ be the suffix of ν starting from position i, where $j_0 = i$. Since π is an Hintikka sequence, by definition of atom and Property 3 in Definition 3, one of the following holds:

   - there is $n \geq 0$ such that $\psi_2 \in A_{j_n}$ and $\psi_1 \in A_{j_k}$ for all $0 \leq k < n$. Since $i = j_0$, from the induction hypothesis, we obtain that $(\mathit{Proj}_\varphi(\pi), i) \models \psi$ proving the thesis.
   - for all $n \geq 0$, $\psi \in A_{j_n}$ and $\psi_2 \notin A_{j_n}$: we show that this case cannot hold. Since the *MAP* ν is infinite, there is $k \geq 0$ such that for all positions $m \geq k$, $m \notin P_f$ iff position m is visited by ν^i. By Property 1, it follows that there is $k \geq 0$ such that for all positions $m \geq k$, $p_\infty \in A_m$ iff position m is visited by ν^i. Since π is fair, it holds that for infinitely many $m \geq 0$, $p_\infty \in A_m$ and $\{\psi_2, \neg(\psi_1 \mathsf{U} \psi_2)\} \cap A_m \neq \emptyset$. Hence, for infinitely many $n \geq 0$, either $\psi \notin A_{j_n}$ or $\psi_2 \in A_{j_n}$, which is a contradiction.

2. $\psi = \rhd^{\mathsf{a}}_I \theta$: we have that $(\mathit{Proj}_\varphi(\pi), i) \models \psi$ *if and only if* there exists $j > i$ such that $j \in \mathit{Pos}(\mathsf{a}, \sigma, i)$, $(\mathit{Proj}_\varphi(\pi), j) \models \theta$, $t_j - t_i \in I$, and for all $k \in \mathit{Pos}(\mathsf{a}, \sigma, i)$ such that $i < k < j$, $(\mathit{Proj}_\varphi(\pi), k) \not\models \theta$ *if and only if* (from the induction hypothesis) there exists $j > i$ such that $j \in \mathit{Pos}(\mathsf{a}, \sigma, i)$, $\theta \in A_j$, $t_j - t_i \in I$, and for all $k \in \mathit{Pos}(\mathsf{a}, \sigma, i)$ such that $i < k < j$, $\theta \notin A_k$ *if and only if* $\mathit{val}^\pi_i(y^{\mathsf{a}}_\theta) \in I$ *if and only if* (from Property 4 in Definition 3) $\rhd^{\mathsf{a}}_I \theta \in A_i$. $\square$

Lemma 10. *For all EC_NTL formulas φ, the mapping Proj_φ is a bijection between the set of fair Hintikka sequences of φ and the set of infinite timed words over $\Sigma_{\mathcal{P}}$.*

Proof. First, we show that Proj_φ is injective. Let π and π' two fair Hintikka sequences such that $\mathit{Proj}_\varphi(\pi) = \mathit{Proj}_\varphi(\pi') = (\sigma, \tau)$. Hence, $\pi = (A_0, \tau_0)(A_1, \tau_1) \ldots$ and $\pi' = (A'_0, \tau_0)(A'_1, \tau_1) \ldots$. By Lemma 9, for all $i \geq 0$, $A_i = A'_i$. Hence, $\pi = \pi'$, and the thesis holds.

It remains to show that Proj_φ is surjective. Let $w = (\sigma, \tau)$ be an infinite timed word over $\Sigma_{\mathcal{P}}$. For each $i \geq 0$, let A_i be the subset of $\mathsf{Cl}(\varphi)$ defined as follows:

- for all $\psi \in \mathsf{Cl}(\varphi) \setminus \{p_\infty, \neg p_\infty\}$, $\psi \in A_i$ if $(w, i) \models \psi$, and $\neg\psi \in A_i$ otherwise.
- $p_\infty \notin A_i$ iff i has a caller whose matching return exists.

Let $\pi = (A_0, \tau_0)(A_1, \tau_1) \ldots$. By construction, for all $i \geq 0$, $A_i \cap \mathcal{P} = \sigma_i$. Thus, it suffices to show that π is a fair Hintikka sequence of φ. By the semantics of EC_NTL, it easily follows that for all $i \geq 0$, A_i is an atom of φ, and π satisfies Properties 1–3 in the definition of Hintikka sequence of φ (Definition 3). Now, let us consider Property 4 in Definition 3 concerning the real-time formulas in $\mathsf{Cl}(\varphi)$. Let us focus on real-time formulas of the form $\rhd^{\mathsf{a}}_I \psi \in \mathsf{Cl}(\varphi)$ (the other cases being similar). We have that $\rhd^{\mathsf{a}}_I \psi \in A_i$ *if and only if* (by construction) $(w, i) \models \rhd^{\mathsf{a}}_I \psi$ *if and only if* (by the semantics of EC_NTL) there exists $j > i$ such that $j \in \mathit{Pos}(\mathsf{a}, \sigma, i)$, $(w, j) \models \psi$, $t_j - t_i \in I$, and for all $k \in \mathit{Pos}(\mathsf{a}, \sigma, i)$ such that $i < k < j$, $(w, k) \not\models \psi$ *if and only if* (by construction) there exists $j > i$ such that $j \in \mathit{Pos}(\mathsf{a}, \sigma, i)$, $\psi \in A_j$, $t_j - t_i \in I$, and for all $k \in \mathit{Pos}(\mathsf{a}, \sigma, i)$ such that $i < k < j$, $\psi \notin A_k$ *if and only if* (by definition of val^π_i) $\mathit{val}^\pi_i(y^{\mathsf{a}}_\psi) \in I$. Hence, Property 4 of Definition 3 holds, and π is an Hintikka sequence of φ.

We show now that π is fair. By construction and EC_NTL semantics, the fulfillment of the fairness constraint about the global until modalities easily follows from standard arguments. Now, let consider the non-local constraint on the proposition p_∞. We need to show that for infinitely many $i \geq 0$, $p_\infty \in A_i$. Since for an infinite word over a pushdown alphabet, either there is an infinite *MAP*, or there are an infinite number of unmatched call positions, or there are an infinite number of unmatched return positions, by construction, the result trivially follows. We consider now the fairness requirements on the abstract until modalities. Let $\psi_1 \mathsf{U}^{\mathsf{a}} \psi_2 \in \mathsf{Cl}(\varphi)$. We need to show that there are infinitely many $i \geq 0$ such that $p_\infty \in A_i$ and $\{\psi_2, \neg(\psi_1 \mathsf{U} \psi_2)\} \cap A_i \neq \emptyset$. By the above observation, one of the following holds:

- either the set H of unmatched call positions in σ is infinite, or the set K of unmatched return positions in σ is infinite: let us consider the second case (the first one being similar). By construction, for all $i \in K$, $p_\infty \in A_i$. Moreover, if $(w, i) \models \psi_1 \mathsf{U}^{\mathsf{a}} \psi_2$, then $(w, j) \models \psi_2$ for some position $j \geq i$ along the *MAP* associated with position i. Since $p_\infty \in A_i$, $p_\infty \in A_j$ as well. Hence, by construction, the result follows.
- σ has an infinite *MAP* ν. By construction, for all positions i visited by ν, $p_\infty \in A_i$. Thus, by the semantics of the abstract until modalities, it follows that there are infinitely many positions j along ν such that $\{\psi_2, \neg(\psi_1 \mathsf{U} \psi_2)\} \cap A_j \neq \emptyset$, and the result follows. $\square$

The final step is to show that the notion of initialized fair Hintikka sequence can be easily captured by a generalized Büchi ECNA. To this end we construct a generalized Büchi ECNA $\mathcal{A}_\varphi$ over $\Sigma_{\mathsf{Cl}(\varphi)}$ accepting the set of initialized fair Hintikka sequences of φ. The set of $\mathcal{A}_\varphi$ states is the set of atoms of φ, and a state A_0 is initial if $\varphi \in A_0$ and A_0 satisfies Property 1 (initial consistency) in Definition 3. In the transition function, we require that the input symbol coincides with the source state in such a way that in a run, the sequence of control states corresponds to the untimed part of the input. Moreover, by the transition function, the automaton checks that the input word is an Hintikka sequence. In particular, for the abstract next and abstract previous requirements (Property 3 in Definition 3), whenever the input symbol A is a call, the automaton pushes on the stack the atom A. In such a way, on reading the matching return A_r (if any) of the call A, the automaton pops A from the stack and can locally check that $AbsNext(A, A_r)$ holds. In order to ensure the real-time requirements (Property 4 in Definition 3), $\mathcal{A}_\varphi$ uses the recorder clocks and predictor clocks. Finally, the generalized Büchi acceptance condition is exploited for checking that the input initialized Hintikka sequence is fair.

Theorem 11. *Given an EC_NTL formula φ, one can construct in singly exponential time a generalized Büchi ECNA $\mathcal{A}_\varphi$ having $2^{O(|\varphi|)}$ states, $2^{O(|\varphi|)}$ stack symbols, set of constants $Const_\varphi$, and $O(|\varphi|)$ clocks. If φ is non-recursive, then $\mathcal{A}_\varphi$ accepts the infinite models of φ; otherwise, $\mathcal{A}_\varphi$ accepts the set of initialized fair Hintikka sequences of φ.*

Proof. Fix an EC_NTL formula φ. For each atom A of φ, we denote by Φ_A the set of clock constraints θ such that the set of atomic constraints of θ has the form

$$\bigcup_{\triangleleft_I^{dir}\psi \in A} \{x_\psi^{dir} \in I\} \cup \bigcup_{\neg\triangleleft_I^{dir}\psi \in A} \{x_\psi^{dir} \in \widehat{I}\} \cup \bigcup_{\triangleright_I^{dir}\psi \in A} \{y_\psi^{dir} \in I\} \cup \bigcup_{\neg\triangleright_I^{dir}\psi \in A} \{y_\psi^{dir} \in \widehat{I}\}$$

where $\widehat{I}$ is either $\{\vdash\}$ or a *maximal* interval over $\mathbb{R}_+$ disjunct from I. The generalized Büchi ECNA $\mathcal{A}_\varphi$ over $\Sigma_{\mathsf{Cl}(\varphi)}$ accepting the set of initialized fair Hintikka sequences of φ is $\mathcal{A}_\varphi = (\Sigma_{\mathsf{Cl}(\varphi)}, Q, Q_0, C_\varphi, Q \cup \{\bot\}, \Delta, \mathcal{F})$, where Q is the set of atoms of φ, C_φ is the set of event clocks associated with $\mathsf{Cl}(\varphi)$ and

- $A_0 \in Q_0$ iff $\varphi \in A_0$ and for all $\ominus^{dir} \psi \in \mathsf{Cl}(\varphi)$, $\neg \ominus^{dir} \psi \in A_0$ with $dir \in \{\mathsf{g}, \mathsf{a}, \mathsf{c}\}$.
- $\mathcal{F} = \{F_\infty\} \cup \{F_{\psi_1 \mathsf{U} \psi_2} \mid \psi_1 \mathsf{U} \psi_2 \in \mathsf{Cl}(\varphi)\} \cup \{F_{\psi_1 \mathsf{U}^{\mathsf{a}} \psi_2} \mid \psi_1 \mathsf{U}^{\mathsf{a}} \psi_2 \in \mathsf{Cl}(\varphi)\}$, where
  - F_∞ consists of the atoms A such that $p_\infty \in A$;
  - for all $\psi_1 \mathsf{U} \psi_2 \in \mathsf{Cl}(\varphi)$, $F_{\psi_1 \mathsf{U} \psi_2}$ consists of the atoms A s.t. $\{\psi_2, \neg(\psi_1 \mathsf{U} \psi_2)\} \cap A \neq \emptyset$;
  - for all $\psi_1 \mathsf{U}^{\mathsf{a}} \psi_2 \in \mathsf{Cl}(\varphi)$, $F_{\psi_1 \mathsf{U}^{\mathsf{a}} \psi_2}$ consists of the atoms A such that $p_\infty \in A$ and $\{\psi_2, \neg(\psi_1 \mathsf{U}^{\mathsf{a}} \psi_2)\} \cap A \neq \emptyset$.

The transition function $\Delta = \Delta_c \cup \Delta_r \cup \Delta_i$ is as follows:

- **Call transitions**: Δ_c consists of transitions of the form $(A_c, A_c, \theta, A', A_c)$ such that $\theta \in \Phi_{A_c}$, $call \in A_c$, $Next(A_c, A')$, and ($p_\infty \in A_c$ if $\bigcirc^{\mathsf{a}} \top \notin A_c$). Moreover, if $ret \notin A'$, then $Caller(A') = \{\ominus^{\mathsf{c}} \psi \in \mathsf{Cl}(\varphi) \mid \psi \in A_c\}$ and ($\bigcirc^{\mathsf{a}} \top \in A_c$ iff $p_\infty \notin A'$).
- **Pop transitions**: Δ_r consists of transitions of the form $(A_r, A_r, \theta, A_c^\perp, A')$ such that $\theta \in \Phi_{A_r}$, $ret \in A_r$, and $Next(A_r, A')$. Moreover, we require:
  - if $ret \notin A'$, then $AbsNext(A_r, A')$ and ($p_\infty \in A_r$ iff $p_\infty \in A'$);
  - if $ret \in A'$, then $\bigcirc^{\mathsf{a}} \top \notin A_r$; if $\ominus^{\mathsf{a}} \top \notin A'$, then $p_\infty \in A_r \cap A'$, and $Caller(A') = \emptyset$;
  - if $A_c^\perp = \bot$, then $\ominus^{\mathsf{a}} \top \notin A_r$; otherwise, $AbsNext(A_c^\perp, A_r)$ and ($p_\infty \in A_c^\perp$ iff $p_\infty \in A_r$) (note that in this case, since $\top \in A_c^\perp$, $\ominus^{\mathsf{a}} \top \in A_r$).

- **Internal transitions**: Δ_i consists of transitions of the form (A_i, A_i, θ, A') s.t. $\theta \in \Phi_{A_i}$, $int \in A_i$, and $Next(A_i, A')$. Moreover we require:
  - if $ret \notin A'$, then $AbsNext(A_i, A')$ and ($p_\infty \in A_i$ iff $p_\infty \in A'$);
  - if $ret \in A'$, then $\bigcirc^{\mathsf{a}} \top \notin A_i$; if $\ominus^{\mathsf{a}} \top \notin A'$, then $p_\infty \in A_i \cap A'$, and $Caller(A') = \emptyset$.

The conditions on the set of initial states reflect the initialization requirement and Property 1 in Definition 3, while the transition function reflects the requirements associated with Properties 2–4 of Definition 3. Finally, the generalized Büchi condition corresponds to the fairness requirement. The unique non-obvious feature is the requirement in Property 3 of Definition 3 that along an Hintikka sequence $(A_0, t_0)(A_1, t_1)\ldots$, for all call positions $i \geq 0$, $\bigcirc^{\mathsf{a}}\top \in A_i$ iff the matching return of i along π is defined. To prove the claim that this requirement is fulfilled by the timed words accepted by $\mathcal{A}_\varphi$ we reason by contradiction. Assume that there is an accepting run of $\mathcal{A}_\varphi$ over an infinite timed word $\pi = (A_0, t_0)(A_1, t_1)\ldots$ such that A_i is an atom for all $i \geq 0$ and for some call position i_c, one of the following holds: either the matching return of i_c is defined and $\bigcirc^{\mathsf{a}}\top \notin A_{i_c}$, or i_c is an unmatched call and $\bigcirc^{\mathsf{a}}\top \in A_{i_c}$. Let us first examine the first case. Let i_r be the matching return of i_c along π. The transition function of $\mathcal{A}_\varphi$ ensures that $\mathit{AbsNext}(A_{i_c}, A_{i_r})$. Hence, since $\top \in A_{i_r}$, it holds that $\bigcirc^{\mathsf{a}}\top \in A_{i_c}$, which is a contradiction. Thus, the first case cannot hold. Now, let us consider the second case. Since $\bigcirc^{\mathsf{a}}\top \in A_{i_c}$ and i_c is an unmatched call, the transition function ensures that $\neg p_\infty \in A_j$ for all $j > i_c$. On the other hand, the first component F_∞ of the generalized Büchi acceptance condition guarantees that for infinitely many i, $p_\infty \in A_i$. Thus, we have a contradiction and the claim is proved.

Hence, A_φ accepts the set of initialized fair Hintikka sequences of φ. Note that A_φ has $2^{O(|\varphi|)}$ states and stack symbols, set of constants Const_φ, and $O(|\varphi|)$ event clocks.

If φ is non-recursive, then the effective clocks are only associated with propositions in $\mathcal{P}$. Thus, by projecting the input symbols of the transition function of $\mathcal{A}_\varphi$ over $\mathcal{P}$, by Proposition 10, we obtain a generalized Büchi ECNA accepting the infinite models of φ. $\square$

Finally, we can state the main result of this section.

Theorem 12. *Given an EC_NTL formula φ over $\Sigma_{\mathcal{P}}$, one can construct in singly exponential time a VPTA, with $2^{O(|\varphi|^3)}$ states and stack symbols, $O(|\varphi|)$ clocks, and set of constants Const_φ, which accepts $\mathcal{L}_T(\varphi)$ (resp., $\mathcal{L}_T^\omega(\varphi)$). Moreover, satisfiability and visibly model-checking for EC_NTL over finite (resp., infinite) timed words are* Exptime*-complete.*

Proof. We focus on the case of infinite timed words. Fix an EC_NTL formula φ over $\Sigma_{\mathcal{P}}$. By Theorem 11, one can construct a generalized Büchi ECNA $\mathcal{A}_\varphi$ over $\Sigma_{\mathsf{Cl}(\varphi)}$ having $2^{O(|\varphi|)}$ states and stack symbols, set of constants Const_φ, and accepting the set of initialized fair Hintikka sequences of φ. By Theorem 5, one can construct a generalized Büchi VPTA $\mathcal{A}'_\varphi$ over $\Sigma_{\mathsf{Cl}(\varphi)}$ accepting $\mathcal{L}_T^\omega(\mathcal{A}_\varphi)$, having $2^{O(|\varphi|^2 \cdot k)}$ states and stack symbols, $O(k)$ clocks, and set of constants Const_φ, where k is the number of atomic constraints used by $\mathcal{A}_\varphi$. Note that $k = O(|\varphi|)$. Thus, by projecting the input symbols of the transition function of $\mathcal{A}'_\varphi$ over $\mathcal{P}$, we obtain a (generalized Büchi) VPTA satisfying the first part of Theorem 12.

For the upper bounds of the second part of Theorem 12, observe that by [16,1] the emptiness problem of generalized Büchi VPTA is solvable in time $O(n^4 \cdot 2^{O(m \cdot \log Km)})$, where n is the number of states, m is the number of clocks, and K is the largest constant used in the clock constraints of the automaton (hence, the time complexity is polynomial in the number of states). Now, given a Büchi VPTA $\mathcal{S}$ over $\Sigma_{\mathcal{P}}$ where all the states are accepting and an EC_NTL formula φ over $\Sigma_{\mathcal{P}}$, model-checking $\mathcal{S}$ against φ reduces to check emptiness of the language $\mathcal{L}_T^\omega(\mathcal{S}) \cap \mathcal{L}_T^\omega(\mathcal{A}'_{\neg\varphi})$, where $\mathcal{A}'_{\neg\varphi}$ is the generalized Büchi VPTA associated with $\neg\varphi$. Thus, since Büchi VPTA are polynomial-time closed under intersection, the membership in Exptime for satisfiability and visibly model-checking of EC_NTL follows. The matching lower bounds directly follow from Exptime-completeness of satisfiability and visibly model-checking for the logic CaRet [3] which is subsumed by EC_NTL. $\square$

6. Nested metric temporal logic (NMTL)

Metric temporal logic (MTL) [27] is a well-known timed linear-time temporal logic which extends LTL with time constraints on *until* modalities. In this section, we introduce an extension of MTL with past, we call *nested* MTL (NMTL, for short), by means of timed versions of CaRet modalities. For the given set $\mathcal{P}$ of atomic propositions containing the special propositions *call*, *ret*, and *int*, the syntax of nested NMTL formulas φ is as follows:

$$\varphi := \top \mid p \mid \varphi \vee \varphi \mid \neg\,\varphi \mid \varphi\, \widehat{\mathsf{U}}_I^{\mathit{dir}} \varphi \mid \varphi\, \widehat{\mathsf{S}}_I^{\mathit{dir}'} \varphi$$

where $p \in \mathcal{P}$, I is an interval in $\mathbb{R}_+$ with endpoints in $\mathbb{N} \cup \{\infty\}$, $\mathit{dir} \in \{\mathsf{g}, \mathsf{a}\}$ and $\mathit{dir}' \in \{\mathsf{g}, \mathsf{a}, \mathsf{c}\}$. The operators $\widehat{\mathsf{U}}_I^{\mathsf{g}}$ and $\widehat{\mathsf{S}}_I^{\mathsf{g}}$ are the standard *timed until* and *timed since* MTL modalities, respectively, $\widehat{\mathsf{U}}_I^{\mathsf{a}}$ and $\widehat{\mathsf{S}}_I^{\mathsf{a}}$ are their non-regular abstract versions, and $\widehat{\mathsf{S}}_I^{\mathsf{c}}$ is the non-regular caller version of $\widehat{\mathsf{S}}_I^{\mathsf{g}}$. MTL with past corresponds to the fragment of NMTL obtained by disallowing the timed abstract and caller modalities, while standard MTL or future MTL is the fragment of MTL with past where the global *timed since* modalities are disallowed.

Given an NMTL formula φ, a timed word $w = (\sigma, \tau)$ over $\Sigma_{\mathcal{P}}$ and a position $0 \leq i < |w|$, the satisfaction relation $(w, i) \models \varphi$ is defined as follows (we omit the clauses for the atomic propositions and Boolean connectives):

$$(w,i) \models \varphi_1 \widehat{\mathsf{U}}^{dir}_I \varphi_2 \;\Leftrightarrow\; \text{there is } j > i \text{ s.t. } j \in Pos(dir,\sigma,i),\ (w,j)\models\varphi_2,\ \tau_j - \tau_i \in I, \text{ and } (w,k)\models\varphi_1 \text{ for all } k \in [i+1, j-1] \cap Pos(dir,\sigma,i)$$

$$(w,i) \models \varphi_1 \widehat{\mathsf{S}}^{dir'}_I \varphi_2 \;\Leftrightarrow\; \text{there is } j < i \text{ s.t. } j \in Pos(dir',\sigma,i),\ (w,j)\models\varphi_2,\ \tau_i - \tau_j \in I, \text{ and } (w,k)\models\varphi_1 \text{ for all } k \in [j+1, i-1] \cap Pos(dir',\sigma,i).$$

In the following, we use some derived operators:

- for $dir \in \{\mathsf{g},\mathsf{a}\}$, $\widehat{\Diamond}^{dir}_I \varphi := \top \widehat{\mathsf{U}}^{dir}_I \varphi$ and $\widehat{\Box}^{dir}_I \varphi := \neg \widehat{\Diamond}^{dir}_I \neg\varphi$;
- for $dir \in \{\mathsf{g},\mathsf{a},\mathsf{c}\}$, $\widehat{\Diamond\!\!\!-}^{dir}_I \varphi := \top \widehat{\mathsf{S}}^{dir}_I \varphi$ and $\widehat{\boxminus}^{dir}_I \varphi := \neg \widehat{\Diamond\!\!\!-}^{dir}_I \neg\varphi$.

Let $\mathcal{I}_{(0,\infty)}$ be the set of *nonsingular* intervals J in $\mathbb{R}_+$ with endpoints in $\mathbb{N}\cup\{\infty\}$ such that either J is unbounded, or J is left-closed with left endpoint 0. Such intervals can be replaced by expressions of the form $\sim c$ for some $c \in \mathbb{N}$ and $\sim \in \{<,\leq,>,\geq\}$. For a generic interval I with left endpoint $c_L \in \mathbb{N}$ and right endpoint $c_R \in \mathbb{N}\cup\{\infty\}$, we denote by $L(I)$ the unbounded interval having c_L as left endpoint and such that $c_L \in L(I)$ iff $c_L \in I$, and by $R(I)$ the left-closed interval having as endpoints 0 and c_R and such that $c_R \in R(I)$ iff $c_R \in I$. Note that $L(I), R(I) \in \mathcal{I}_{(0,\infty)}$. We focus on the following two fragments of NMTL:

- $\mathsf{NMITL}_{(0,\infty)}$: obtained by allowing only intervals in $\mathcal{I}_{(0,\infty)}$;
- *Future* NMTL: obtained by disallowing the variants of *timed since* modalities.

Example 5. We express the real-time LTL-like properties proposed in Example 4. Note that all the formulas are $\mathsf{NMITL}_{(0,\infty)}$ formulas.

- *Real-time total correctness.* If the pre-condition p holds when the procedure A is invoked, then the procedure must return within k time units and q must hold upon return (p_A marks calls to procedure A): $\widehat{\Box}^{\mathsf{g}}_{[0,\infty]}\big((call \wedge p \wedge p_A) \rightarrow (\neg\top\ \widehat{\mathsf{U}}^{\mathsf{a}}_{[0,k]}\ q \wedge ret)\big)$.
- *Local bounded-time response properties.* In the local computation of a procedure A, every request p is followed by a response q within k time units (c_A denotes that the control is inside procedure A): $\widehat{\Box}^{\mathsf{g}}_{[0,\infty]}\big((p \wedge c_A) \rightarrow \widehat{\Diamond}^{\mathsf{a}}_{[0,k]}\ q\big)$.
- *Real-time properties over the stack content.* A procedure A is invoked only if procedure B belongs to the call stack and within k time units since the activation of B (the calls to A and B are marked by p_A and p_B, respectively): $\widehat{\Box}^{\mathsf{g}}_{[0,\infty]}\big((call \wedge p_A) \rightarrow \widehat{\Diamond\!\!\!-}^{\mathsf{c}}_{[0,k]}\ p_B\big)$.

It is known that for the considered pointwise semantics, $\mathsf{MITL}_{(0,\infty)}$ [4] (the fragment of MTL allowing only intervals in $\mathcal{I}_{(0,\infty)}$) and EC_TL are equally expressive [33]. In the following, we show that the result generalizes to the nested extensions of $\mathsf{MITL}_{(0,\infty)}$ and EC_TL.

Moreover, it is well-known that satisfiability of MTL with past is undecidable [6,30]. Undecidability already holds for future MTL interpreted over infinite timed words [30]. However, over finite timed words, satisfiability of future MTL instead is decidable [31]. In Subsection 6.1, we show that the addition of the future abstract timed modalities to future MTL makes the satisfiability problem undecidable also over finite timed words.

Given two formulas φ_1 and φ_2 in NMTL + EC_NTL (i.e., the extension of NMTL with the temporal modalities of EC_NTL), φ_1 and φ_2 are *globally equivalent*, denoted $\varphi_1 \equiv \varphi_2$, if for each timed word w over $\Sigma_{\mathcal{P}}$ and $0 \leq i < |w|$, $(w,i)\models\varphi_1$ iff $(w,i)\models\varphi_2$. We first show that EC_NTL is subsumed by $\mathsf{NMITL}_{(0,\infty)}$. For this, we consider the following global equivalences, which easily follow from the semantics of EC_NTL and $\mathsf{NMITL}_{(0,\infty)}$and allow the expression of the temporal modalities of EC_NTL in terms of the temporal modalities of $\mathsf{NMITL}_{(0,\infty)}$.

Proposition 4. *For all formulas φ_1 and φ_2 in NMTL + EC_NTL, the following holds, for $dir \in \{\mathsf{g},\mathsf{a}\}$, $dir' \in \{\mathsf{g},\mathsf{a},\mathsf{c}\}$, and $\sim \in \{<,\leq,>,\geq\}$:*

- $\bigcirc^{dir}\varphi_1 \equiv \bot\, \widehat{U}^{dir}_{\geq 0}\, \varphi_1$ *and* $\circleddash^{dir'}\varphi_1 \equiv \bot\, \widehat{S}^{dir'}_{\geq 0}\, \varphi_1$*;*
- $\varphi_1 U^{dir} \varphi_2 \equiv \varphi_2 \vee (\varphi_1 \wedge (\varphi_1 \widehat{U}^{dir}_{\geq 0} \varphi_2))$ *and* $\varphi_1 S^{dir'} \varphi_2 \equiv \varphi_2 \vee (\varphi_1 \wedge (\varphi_1 \widehat{S}^{dir'}_{\geq 0} \varphi_2))$*;*
- $\rhd^{dir}_{\sim c}\varphi_1 \equiv \neg\varphi_1 \widehat{U}^{dir}_{\sim c}\, \varphi_1$ *and* $\lhd^{dir'}_{\sim c}\varphi_1 \equiv \neg\varphi_1 \widehat{S}^{dir'}_{\sim c}\, \varphi_1$*;*
- $\rhd^{dir}_{I}\varphi_1 \equiv \rhd^{dir}_{L(I)}\varphi_1 \wedge \rhd^{dir}_{R(I)}\varphi_1$ *and* $\lhd^{dir'}_{I}\varphi_1 \equiv \lhd^{dir'}_{L(I)}\varphi_1 \wedge \lhd^{dir'}_{R(I)}\varphi_1$.

On the opposite side, for the expressibility of $\mathsf{NMITL}_{(0,\infty)}$ into EC_NTL, we consider the following global equivalences which allow us to express the temporal modalities of $\mathsf{NMITL}_{(0,\infty)}$ in terms of the temporal modalities of EC_NTL.

Proposition 5. *For all formulas φ_1 and φ_2 in NMTL + EC_NTL, the following holds, where $c \in \mathbb{N}$, $dir \in \{\mathsf{g},\mathsf{a}\}$, $dir' \in \{\mathsf{g},\mathsf{a},\mathsf{c}\}$, $\prec \in \{<,\leq\}$, $\succ \in \{>,\geq\}$, $\geq^{-1}$ is $<$, and $>^{-1}$ is $\leq$:*

1. $\varphi_1 \widehat{U}^{dir}_{\prec c} \varphi_2 \equiv \bigcirc^{dir}(\varphi_1 U^{dir} \varphi_2) \wedge \rhd^{dir}_{\prec c} \varphi_2$;
2. $\varphi_1 \widehat{S}^{dir'}_{\prec c} \varphi_2 \equiv \ominus^{dir'}(\varphi_1 S^{dir'} \varphi_2) \wedge \lhd^{dir'}_{\prec c} \varphi_2$;
3. $\varphi_1 \widehat{U}^{dir}_{\succ c} \varphi_2 \equiv \widehat{\Box}^{dir}_{\succ^{-1} c}(\varphi_1 \wedge \bigcirc^{dir}(\varphi_1 U^{dir} \varphi_2)) \wedge \bigcirc^{dir}(\varphi_1 U^{dir} \varphi_2)$;
4. $\varphi_1 \widehat{S}^{dir'}_{\succ c} \varphi_2 \equiv \widehat{\boxminus}^{dir'}_{\succ^{-1} c}(\varphi_1 \wedge \ominus^{dir'}(\varphi_1 S^{dir'} \varphi_2)) \wedge \ominus^{dir'}(\varphi_1 S^{dir'} \varphi_2)$.

Proof. The global equivalences in items 1 and 2 easily follow from the semantics of $\mathsf{NMITL}_{(0,\infty)}$ and $\mathsf{EC\_NTL}$. Now, let us consider items 3 and 4. We focus on the *abstract until* modalities and assume that $\succ$ is $>$ (the other cases being similar). Let $w = (\sigma, \tau)$ be a timed word over $\Sigma_{\mathcal{P}}$ and $0 \leq i < |w|$. We need to show that $(w,i) \models \varphi_1 \widehat{\mathsf{U}}^{\mathsf{a}}_{>c} \varphi_2 \Leftrightarrow (w,i) \models \theta$, where $\theta = \widehat{\Box}^{\mathsf{a}}_{\leq c}(\varphi_1 \wedge \bigcirc^{\mathsf{a}}(\varphi_1 \mathsf{U}^{\mathsf{a}} \varphi_2)) \wedge \bigcirc^{\mathsf{a}}(\varphi_1 \mathsf{U}^{\mathsf{a}} \varphi_2)$. We consider the left implication $\Leftarrow$ (the right implication $\Rightarrow$ being simpler). Assume that $(w,i) \models \theta$. Let $P_{\leq c}$ be the set of positions $j \in Pos(\sigma, \mathsf{a}, i)$ such that $j > i$ and $\tau_j - \tau_i \leq c$. There are two cases:

- $P_{\leq c}$ is empty: since $(w,i) \models \bigcirc^{\mathsf{a}}(\varphi_1 \mathsf{U}^{\mathsf{a}} \varphi_2)$, there is $j \in Pos(\sigma, \mathsf{a}, i)$ such that $j > i$, $(w,j) \models \varphi_2$ and $(w,h) \models \varphi_1$ for all $h \in Pos(\sigma, \mathsf{a}, i) \cap [i+1, j-1]$. Since $P_{\leq c} = \emptyset$, we have that $\tau_j - \tau_i > c$. Hence, $(w,i) \models \varphi_1 \widehat{\mathsf{U}}^{\mathsf{a}}_{>c} \varphi_2$.
- $P_{\leq c}$ is *not* empty: let j be the greatest position of $P_{\leq c}$ (note that such a position exists). Since $(w,i) \models \widehat{\Box}^{\mathsf{a}}_{\leq c}(\varphi_1 \wedge \bigcirc^{\mathsf{a}}(\varphi_1 \mathsf{U}^{\mathsf{a}} \varphi_2))$, we have that $(w,h) \models \varphi_1$ for all $h \in Pos(\sigma, \mathsf{a}, i) \cap [i+1, j]$ and there exists $\ell > j$ such that $\ell \in Pos(\sigma, \mathsf{a}, i)$, $(w,\ell) \models \varphi_2$ and $(w,k) \models \varphi_1$ for all $h \in Pos(\sigma, \mathsf{a}, i) \cap [j+1, \ell-1]$. Since $\ell \notin P_{\leq c}$, we have that $\tau_\ell - \tau_i > c$. It follows that $(w,i) \models \varphi_1 \widehat{\mathsf{U}}^{\mathsf{a}}_{>c} \varphi_2$, proving the assertion. $\square$

Propositions 4 and 5 provide linear-time translations from $\mathsf{EC\_NTL}$ into $\mathsf{NMITL}_{(0,\infty)}$, and vice versa, which preserve global equivalence. Therefore, as a corollary of Theorem 12, we can fix the complexity of decision problems for $\mathsf{NMITL}_{(0,\infty)}$.

Theorem 13. *EC_NTL and $NMITL_{(0,\infty)}$ are expressively equivalent. Satisfiability and visibly model-checking for $NMITL_{(0,\infty)}$ over finite (resp., infinite) timed words are* EXPTIME*-complete.*

6.1. Undecidability of future NMTL over finite timed words

Finally, in this section we show that future abstract timed modalities add expressiveness to future MTL over finite words leading to the undecidability of the satisfiability problem.

Theorem 14. *Satisfiability of future NMTL over finite timed words is undecidable.*

We prove undecidability by a reduction from the halting problem for Minsky 2-counter machines [29]. We recall that a Minsky 2-counter machine M which is a tuple $M = (\mu, \mathsf{Inst}, \ell_{init}, \ell_{halt})$, where μ is a finite set of labels (or program counters), $\ell_{init}, \ell_{halt} \in \mu$, and Inst is a mapping assigning to each label $\ell \in \mu \setminus \{\ell_{halt}\}$ an instruction for either

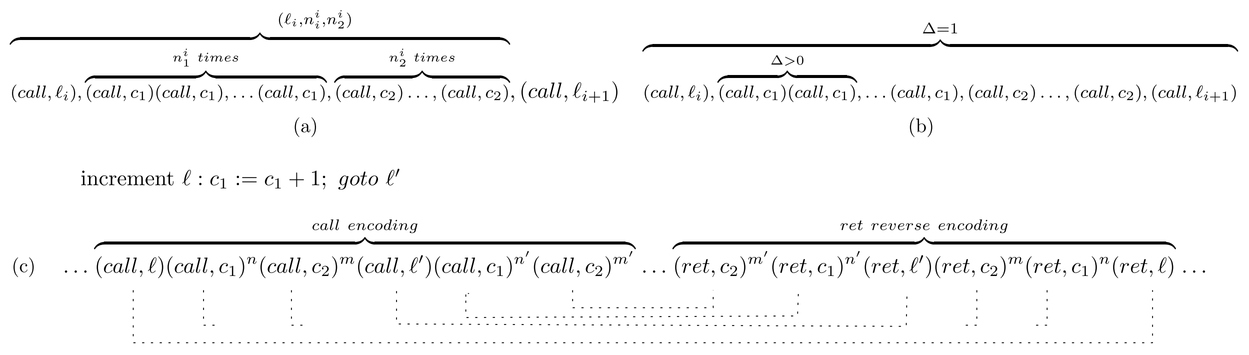
- *increment*: $c_h := c_h + 1$; `goto` ℓ_r, or
- *decrement*: `if` $c_h > 0$ `then` $c_h := c_h - 1$; `goto` ℓ_s `else goto` ℓ_t

where $h \in \{1,2\}$, $\ell_s \neq \ell_t$, and $\ell_r, \ell_s, \ell_t \in \mu$.

The machine M induces a transition relation $\longrightarrow$ over configurations of the form (ℓ, n_1, n_2), where ℓ is a label of an instruction to be executed and $n_1, n_2 \in \mathbb{N}$ represent current values of counters c_1 and c_2, respectively. A computation of M is a finite sequence $C_1 \ldots C_k$ of configurations such that $C_i \longrightarrow C_{i+1}$ for all $i \in [1, k-1]$. The machine M *halts* if there is a computation starting at $(\ell_{init}, 0, 0)$ and leading to configuration (ℓ_{halt}, n_1, n_2) for some $n_1, n_2 \in \mathbb{N}$. The halting problem is to decide whether a given machine M halts. The problem is undecidable [29]. We adopt the following notation, for $\ell \in \mu \setminus \{\ell_{halt}\}$:

- if $\mathsf{Inst}(\ell)$ is an increment instruction of the form $c_h := c_h + 1$; `goto` ℓ_r, we define $c(\ell) := c_h$ and $succ(\ell) := \ell_r$;
- if $\mathsf{Inst}(\ell)$ is a decrement instruction of the form `if` $c_h > 0$ `then` $c_h := c_h - 1$; `goto` ℓ_r `else goto` ℓ_s, we define $c(\ell) := c_h$, $dec(\ell) := \ell_r$, and $zero(\ell) := \ell_s$.

We encode the computations of M by using finite words over the pushdown alphabet $\Sigma_{\mathcal{P}}$, where $\mathcal{P} = \mu \cup \{c_1, c_2\} \cup \{call, ret, int\}$. For a finite word $\sigma = a_1 \ldots a_n$ over $\mu \cup \{c_1, c_2\}$, we denote by σ^R the reverse of σ, and by $(call, \sigma)$ (resp., (ret, σ)) the finite word over $\Sigma_{\mathcal{P}}$ given by $\{a_1, call\} \ldots \{a_n, call\}$ (resp., $\{a_1, ret\} \ldots \{a_n, ret\}$). We associate to each M-configuration (ℓ, n_1, n_2) two distinct encodings: the *call-code* which is the finite word over $\Sigma_{\mathcal{P}}$ given by $(call, \ell c_1^{n_1} c_2^{n_2})$, and the *ret-code* which is given by $(ret, (\ell c_1^{n_1} c_2^{n_2})^R)$ intuitively corresponding to the matched-return version of the call-code. A computation π of M is then represented by the well-matched word $(call, \sigma_\pi) \cdot (ret, (\sigma_\pi)^R)$, where σ_π is obtained by concatenating the call-codes of the individual configurations along π. An example of encoding for configurations and computations is given in Fig. 5.



(ℓ_i, n_i^i, n_2^i)
n_1^i times
n_2^i times
$(call, \ell_i), (call, c_1)(call, c_1), \ldots (call, c_1), (call, c_2) \ldots, (call, c_2), (call, \ell_{i+1})$
(a)
$\Delta = 1$
$\Delta > 0$
$(call, \ell_i), (call, c_1)(call, c_1), \ldots (call, c_1), (call, c_2) \ldots, (call, c_2), (call, \ell_{i+1})$
(b)
increment $\ell : c_1 := c_1 + 1;$ goto ℓ'
call encoding
ret reverse encoding
(c) $\ldots (call, \ell)(call, c_1)^n (call, c_2)^m (call, \ell')(call, c_1)^{n'} (call, c_2)^{m'} \ldots (ret, c_2)^{m'} (ret, c_1)^{n'} (ret, \ell')(ret, c_2)^m (ret, c_1)^n (ret, \ell) \ldots$

Fig. 5. The encoding of a computation of a Minsky 2-counter machine: (a) the untimed word encoding a computation; (b) the timing constraints of encoding (timestamps are omitted, Δ is the time distance); (c) the matching of the call encoding and reverse encoding of the configuration change for the execution of an increment instruction $\ell : c_1 := c_1 + 1;$ *goto* ℓ'. A future MTL formula checks that $n' \geq n + 1$ and $m \geq m'$ on the call encoding and a future MTL formula checks that $n' \leq n + 1$ and $m \leq m'$ on the ret encoding.

Formally, let $\mathcal{L}_{halt}$ be the set of finite words over $\Sigma_{\mathcal{P}}$ of the form $(call, \sigma) \cdot (ret, \sigma^R)$ (*well-matching requirement*) such that the call part $(call, \sigma)$ satisfies:

- *Consecution:* $(call, \sigma)$ is a sequence of call-codes, and for each pair (C, C') of adjacent call-codes, with C preceding C', the associated M-configurations, say (ℓ, n_1, n_2) and (ℓ', n'_1, n'_2), satisfy: $\ell \neq \ell_{halt}$ and
  - if $\mathsf{Inst}(\ell)$ is an increment instruction and $c(\ell) = c_h$, then $\ell' = succ(\ell)$ and $n'_h > 0$; if instead $\mathsf{Inst}(\ell)$ is a decrement instruction and $c(\ell) = c_h$, then *either* $\ell' = zero(\ell)$ and $n_h = n'_h = 0$, *or* $\ell' = dec(\ell)$ and $n_h > 0$.
- *Initialization:* σ has a prefix of the form $\ell_{init} \cdot \ell$ for some $\ell \in \mu$.
- *Halting:* ℓ_{halt} occurs along σ.
- For each pair (C, C') of adjacent call-codes in $(call, \sigma)$, with C preceding C' and s.t. C' is not halting, the associated M-configurations, say (ℓ, n_1, n_2) and (ℓ', n'_1, n'_2), satisfy (for technical convenience, we do not require that the counters in a configuration having as successor an halting configuration are correctly updated):
  - *Increment requirement:* if $\mathsf{Inst}(\ell)$ is an increment instruction and $c(\ell) = c_h$, then $n'_h = n_h + 1$ and $n'_{3-h} = n_{3-h}$;
  - *Decrement requirement:* if $\mathsf{Inst}(\ell)$ is a decrement instruction and $c(\ell) = c_h$, then $n'_{3-h} = n_{3-h}$: moreover, if $\ell' = dec(\ell)$, then $n'_h = n_h - 1$.

Evidently, M halts iff $\mathcal{L}_{halt} \neq \emptyset$. We construct in polynomial time a future NMTL formula φ_M over $\mathcal{P}$ such that the set of untimed components σ in the finite timed words (σ, τ) satisfying φ_M is exactly $\mathcal{L}_{halt}$ proving the assertion of Theorem 14. In the construction of φ_M, we exploit the future LTL modalities and the abstract next modality $\bigcirc^{\mathsf{a}}$ which can be expressed in future NMTL. The formula φ_M is $\varphi_M := \varphi_{WM} \wedge \varphi_{\mathsf{LTL}} \wedge \varphi_{Time}$ where φ_{WM}, φ_{LTL} and φ_{Time} are defined as follows. The formula φ_{WM} is a future CaRet formula ensuring the well-matching requirement:

$$\varphi_{WM} := call \wedge \bigcirc^{\mathsf{a}}(\neg \bigcirc^{\mathsf{g}} \top) \wedge \Box^{\mathsf{g}} \neg int \wedge \neg \Diamond^{\mathsf{g}} (ret \wedge \Diamond^{\mathsf{g}} call)$$

The conjunct φ_{LTL} is a standard future LTL formula ensuring the consecution, initialization, and halting requirements. The definition of φ_{LTL} is straightforward and we omit the details of the construction. Finally, we illustrate the construction of the conjunct φ_{Time} which is a future MTL formula enforcing the increment and decrement requirements by means of time constraints. Let w be a finite timed word over $\Sigma_{\mathcal{P}}$. By formulas φ_{WM} and φ_{LTL}, we can assume that the untimed part of w has the form $(call, \sigma) \cdot (ret, \sigma^R)$ and that the call part $(call, \sigma)$ satisfies the consecution, initialization, and halting requirements. The formula φ_{Time} ensures the following additional requirements:

- *Strict time monotonicity:* the time distance between distinct positions is always greater than zero. This can be expressed by the formula $\Box^{\mathsf{g}}(\neg \widehat{\Diamond}^{\mathsf{g}}_{[0,0]} \top)$.
- *1-Time distance between adjacent labels:* the time distance between the μ-positions of two adjacent *call*-codes (resp., *ret*-codes) is 1. This can be expressed as follows:

$$\bigwedge_{t \in \{call, ret\}} \Box^{\mathsf{g}} \Big([t \wedge \bigvee_{\ell \in \mu} \ell \wedge \widehat{\Diamond}^{\mathsf{g}}(t \wedge \bigvee_{\ell \in \mu} \ell)] \rightarrow \widehat{\Diamond}^{\mathsf{g}}_{[1,1]}(t \wedge \bigvee_{\ell \in \mu} \ell) \Big).$$

- *Increment and decrement requirements:* fix a *call*-code C along the call part immediately followed by some non-halting call-code C'. Let (ℓ, n_1, n_2) (resp., (ℓ', n'_1, n'_2)) be the configuration encoded by C (resp., C'), and $c(\ell) = c_h$ (for some

$h = 1, 2$). Note that $\ell \neq \ell_{halt}$. First, assume that $\mathsf{Inst}(\ell)$ is an increment instruction. We need to enforce that $n'_h = n_h + 1$ and $n'_{3-h} = n_{3-h}$. For this, we first require that

(*) for every *call*-code C with label ℓ, every c_{3-h}-position has a future call c_{3-h}-position at (time) distance 1, and every c_h-position has a future call c_h-position j at distance 1 such that $j+1$ is still a call c_h-position.

By the strict time monotonicity and the 1-Time distance between adjacent labels, requirement (*) ensures that $n'_h \geq n_h + 1$ and $n'_{3-h} \geq n_{3-h}$. To guarantee that $n'_h \leq n_h + 1$ and $n'_{3-h} \leq n_{3-h}$, we crucially exploit the return part (ret, σ^R) corresponding to the reverse of the call part $(call, \sigma)$. In particular, along the return part, the reverse of C' is immediately followed by the reverse of C. Thus, we additionally require that

(**) for every *non-first ret*-code R which is immediately followed by a *ret*-code with label ℓ, each c_{3-h}-position has a future c_{3-h}-position at distance 1, and each non-first c_h-position of R has a future c_h-position at distance 1.

Requirements (*) and (**) can be expressed by the following two formulas.

$$\Box^{\mathsf{g}}\Big((call \wedge \ell) \to \widehat{\Box}^{\mathsf{g}}_{[0,1]}[(c_{3-h} \to \widehat{\Diamond}^{\mathsf{g}}_{[1,1]} c_{3-h}) \wedge (c_h \to \widehat{\Diamond}^{\mathsf{g}}_{[1,1]}(c_h \wedge \bigcirc^{\mathsf{g}} c_h))]\Big)$$

$$\bigwedge_{\ell' \in \mu} \Box^{\mathsf{g}}\Big((ret \wedge \ell' \wedge \widehat{\Diamond}^{\mathsf{g}}_{[2,2]}\ell) \longrightarrow \widehat{\Box}^{\mathsf{g}}_{[0,1]}\big([c_{3-h} \to \widehat{\Diamond}^{\mathsf{g}}_{[1,1]} c_{3-h}] \wedge [(c_h \wedge \bigcirc^{\mathsf{g}} c_h) \to \bigcirc^{\mathsf{g}}\widehat{\Diamond}^{\mathsf{g}}_{[1,1]} c_h]\big)\Big)$$

Now, assume that $\mathsf{Inst}(\ell)$ is a decrement instruction. We need to enforce that $n'_{3-h} = n_{3-h}$, and whenever $\ell' = dec(\ell)$, then $n'_h = n_h - 1$. This can be ensured by requirements similar to Requirements (*) and (**), and we omit the details.

This concludes the proof of Theorem 14.

7. Conclusions

In this paper we have introduced and studied ECNA, a robust subclass of VPTA allowing the expression of meaningful non-regular timed properties of recursive systems. We proved that ECNA extends the expressivity of other subclass of robusts VPTA though preserving the same complexity cost. The closure under Boolean operations, and the decidability of languages inclusion and visibly model-checking makes ECNA amenable to specification and verification purposes. For this reason we have investigated the possibility of introducing real-time context-free features also in the context of linear-time temporal logics. In particular we have introduced two timed linear-time temporal logics for specifying real-time context-free requirements in a pointwise semantics setting: Event-Clock Nested Temporal Logic (EC_NTL) and Nested Metric Temporal Logic (NMTL). We have shown that while EC_NTL, which is the natural counterpart of ECNA, is decidable and tractable, NMTL is undecidable even for its future fragment interpreted over finite timed words. Moreover, we have established that the $\mathsf{MITL}_{(0,\infty)}$-like fragment $\mathsf{NMITL}_{(0,\infty)}$ of NMTL is decidable and tractable. As future research, we shall investigate decidability and complexity issues for the more general fragment of NMTL obtained by disallowing singular intervals. Such a fragment represents the NMTL counterpart of Metric Interval Temporal Logic (MITL), a decidable (and EXPSPACE-complete) fragment of MTL [4] which is strictly more expressive than $\mathsf{MITL}_{(0,\infty)}$ [33].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Removal of abstract recorder clocks in nested VPTA

Theorem 1 *(Removal of abstract recorder clocks). Given a generalized Büchi nested VPTA $\mathcal{A}$ with set of event clocks C and an abstract recorder clock $x^{\mathsf{a}}_b \in C$, one can construct in singly exponential time a generalized Büchi nested VPTA $\mathcal{A}_{x^{\mathsf{a}}_b}$ with set of event clocks $C \setminus \{x^{\mathsf{a}}_b\}$ such that $\mathcal{L}_T(\mathcal{A}_{x^{\mathsf{a}}_b}) = \mathcal{L}_T(\mathcal{A})$ and $K_{\mathcal{A}_{x^{\mathsf{a}}_b}} = K_{\mathcal{A}}$. Moreover, $\mathcal{A}_{x^{\mathsf{a}}_b}$ has $O(n \cdot 2^{O(p)})$ states and $m + O(p)$ clocks, where n is the number of $\mathcal{A}$-states, m is the number of standard $\mathcal{A}$-clocks, and p is the number of* event-clock *atomic constraints on x^{a}_b used by $\mathcal{A}$.*

In the following, we provide the proof of Theorem 1. Fix a generalized Büchi nested VPTA $\mathcal{A} = (\Sigma, Q, Q_0, C \cup C_{st}, \Gamma \cup \{\top\}, \Delta, \mathcal{F})$ such that $x^{\mathsf{a}}_b \in C$. We can assume that for each transition δ of $\mathcal{A}$, there is exactly one atomic constraint $x^{\mathsf{a}}_b \in I$ on x^{a}_b used as conjunct in the clock constraint of δ. We construct a generalized Büchi nested VPTA $\mathcal{A}_{x^{\mathsf{a}}_b}$ equivalent to $\mathcal{A}$ whose set of event clocks is $C \setminus \{x^{\mathsf{a}}_b\}$, and whose set of standard clocks is $C_{st} \cup C_{new}$, where C_{new} consists of the fresh standard clocks $z_{\succ \ell}$ (resp., $z_{\prec u}$) for each lower-bound constraint $x^{\mathsf{a}}_b \succ \ell$ (resp., upper-bound constraint $x^{\mathsf{a}}_b \prec u$) of $\mathcal{A}$ involving x^{a}_b.

We first explain the basic ideas of the translation. Note that a global recorder clock x_b^{g} can be trivially converted in a standard clock by resetting it whenever b occurs along the input word. This approach is not correct for the abstract recorder clock x_b^{a}, since along a *MAP* ν, there may be consecutive positions i_c and i_r such that i_c is a call with matching return i_r, and b may occur along positions in $[i_c+1, i_r-1]$ which are associated with *MAP* distinct from ν. Thus, as in the case of the abstract predictor clock y_b^{a}, we replace x_b^{a} with the set C_{new} of fresh standard clocks defined above. For a given infinite word σ over Σ, a *MAP* ν of σ and a position i of ν, we denote by $\mathit{infix}_b(\nu, i)$ the infix of ν defined as follows: if there exists the smallest b-position $j > i$ visited by ν, then $\mathit{infix}_b(\nu, i)$ is the infix of ν between the next position of i along ν and the position j; otherwise, $\mathit{infix}_b(\nu, i)$ is the suffix of ν starting from the next position of i along ν (note that in this case $\mathit{infix}_b(\nu, i)$ is empty if i is the last position of ν). The main idea of the construction is that when b occurs at the current position i of the input word, the simulating automaton $\mathcal{A}_{x_b^{\mathsf{a}}}$ guesses the set of lower-bound and upper-bound constraints on x_b^{a} which will be used by $\mathcal{A}$ along the portion $\mathit{infix}_b(\nu, i)$ of the current *MAP*.

First, let us consider lower-bound constraints $x_b^{\mathsf{a}} \succ \ell$. Assume that b occurs at position i of the input word for the first time and that i is not the last position of the current *MAP* ν (hence, $\mathit{infix}_b(\nu, i)$ is not empty). Then, $\mathcal{A}_{x_b^{\mathsf{a}}}$ guesses the set of lower-bound constraints $x_b^{\mathsf{a}} \succ \ell$ which will be used by $\mathcal{A}$ along $\mathit{infix}_b(\nu, i)$. For each of such guessed constraints $x_b^{\mathsf{a}} \succ \ell$, the associated new clock $z_{\succ\ell}$ is reset; moreover, if i is not a call (resp., i is a call), $\mathcal{A}_{x_b^{\mathsf{a}}}$ carries the obligation $\succ\ell$ in its control state (resp., pushes the obligation $\succ\ell$ onto the stack). On visiting the positions j in $\mathit{infix}_b(\nu, i)$, $\mathcal{A}_{x_b^{\mathsf{a}}}$ checks that the guess is correct by verifying that for the current lower-bound constraint $x_b^{\mathsf{a}} \succ \ell'$ used by $\mathcal{A}$, $\succ\ell'$ is in the current set of obligations, and constraint $z_{\succ\ell'} \succ \ell'$ holds. Moreover, at position j, $\mathcal{A}_{x_b^{\mathsf{a}}}$ guesses whether the constraint $x_b^{\mathsf{a}} \succ \ell'$ will be again used along $\mathit{infix}_b(\nu, i)$, or not. In the first case, the obligation $\succ\ell'$ is kept, otherwise, it is discarded. The crucial observation is that:

If a call $i_c \geq i$ occurs along ν before the last position (if any) of $\mathit{infix}_b(\nu, i)$, we know that the next position of i_c along ν is the matching return i_r of i_c, i_r is visited by $\mathit{infix}_b(\nu, i)$, and all the *MAP* visiting positions $h \in [i_c+1, i_r-1]$ are finite and terminate at positions $k < i_r$. Thus, the fulfillment of a lower-bound constraint $x_b^{\mathsf{a}} \succ \ell$ asserted at a position of such *MAP* always implies the fulfillment of the same constraint when asserted at a position $j \geq i_r$ of $\mathit{infix}_b(\nu, i)$. Thus, at the time of a guess (i.e., when a b occurs) along a *MAP* visiting positions in $[i_c+1, i_r-1]$, the clocks $z_{\succ\ell}$ associated with the guessed lower-bound constraints $x_b^{\mathsf{a}} \succ \ell$ can be safely reset.

At each position i, $\mathcal{A}_{x_b^{\mathsf{a}}}$ keeps track in its control state of the lower-bound obligations for the part $\mathit{infix}_b(\nu, i)$ of the current *MAP* ν. Whenever a call i_c occurs, the guessed lower-bound obligations for the matching return i_r of i_c are pushed on the stack in order to be recovered at position i_r. Moreover, if i_c+1 is not a return (i.e., $i_r \neq i_c+1$), then $\mathcal{A}_{x_b^{\mathsf{a}}}$ moves to a control state where the set of lower-bound obligations is empty (consistently with the fact that i_c+1 is the initial position of the *MAP* visiting i_c+1).

The case for upper-bound constraints $x_b^{\mathsf{a}} \prec u$ is symmetric. Whenever b occurs at a position i of the input word which is not the last position of the current *MAP* ν and $\mathcal{A}_{x_b^{\mathsf{a}}}$ guesses that the constraint $x_b^{\mathsf{a}} \prec u$ will be used by $\mathcal{A}$ along the infix $\mathit{infix}_b(\nu, i)$, then, assuming that no obligation is currently associated to the constraint $x_b^{\mathsf{a}} \prec u$, $\mathcal{A}_{x_b^{\mathsf{a}}}$ resets the standard clock $z_{\prec u}$ and carries the fresh obligation (*first*,$\prec u$) in its control state (resp., pushes the obligation (*first*,$\prec u$) onto the stack) if i is not a call (resp., i is a call). When at a position j of the infix $\mathit{infix}_b(\nu, i)$, $\mathcal{A}$ uses the constraint $x_b^{\mathsf{a}} \prec u$, $\mathcal{A}_{x_b^{\mathsf{a}}}$ checks that (*first*, $\prec u$) is in the current set of obligations, and that the constraint $z_{\prec u} \prec u$ holds. The obligation (*first*,$\prec u$) is removed or confirmed, depending on whether $\mathcal{A}_{x_b^{\mathsf{a}}}$ guesses that $x_b^{\mathsf{a}} \prec u$ will be again used by $\mathcal{A}$ along $\mathit{infix}_b(\nu, i)$ or not. Assume now that a call position $i_c \geq i$ occurs along ν before the last position (if any) of $\mathit{infix}_b(\nu, i)$, and let i_r be the matching return of i_c. The important observation is that the fulfillment of an upper-bound constraint $x_b^{\mathsf{a}} \prec u$ asserted at a position $j \geq i_r$ of $\mathit{infix}_b(\nu, i)$ always implies the fulfillment of the same constraint when asserted at a position h of a *MAP* ν' visiting positions in $[i_c+1, i_r-1]$ such that h is preceded along ν' by a position where b occurs. Thus, if the constraint $x_b^{\mathsf{a}} \prec u$ is guessed to hold at a position $j \geq i_r$ of $\mathit{infix}_b(\nu, i)$, for the guesses on the constraint $x_b^{\mathsf{a}} \prec u$ done by $\mathcal{A}_{x_b^{\mathsf{a}}}$ along the positions in $[i_c+1, i_r-1]$, the clock $z_{\prec u}$ is not reset at the times of the guesses (i.e., when b occurs along the positions in $[i_c+1, i_r-1]$).

Whenever a call i_c occurs, the updated set O of upper-bound and lower-bounds obligations is pushed onto the stack in order to be recovered at the matching return i_r of i_c. Moreover, if i_c+1 is not a return (i.e., $i_r \neq i_c+1$), then $\mathcal{A}_{x_b^{\mathsf{a}}}$ moves to a control state where, while the set of lower-bound obligations is empty, the set of upper-bound obligations is obtained from O by replacing each upper-bound obligation (f,$\prec u$), where $f \in \{\mathit{live}, \mathit{first}\}$ with the live obligation (*live*,$\prec u$). A live obligation (*live*,$\prec u$) asserted at the initial position i_c+1 of the *MAP* ν visiting i_c+1 (note that ν terminates at position i_r-1) is used by $\mathcal{A}_{x_b^{\mathsf{a}}}$ to remind that the clock $z_{\prec u}$ cannot be reset along ν when b occurs. Intuitively, live upper-bound obligations are propagated from the caller *MAP* to the called *MAP*. Note that fresh upper-bound obligations (*first*, $\prec u$) always refer to guesses done along the current *MAP* and differently from the live upper-bound obligations, may be removed, when along the current *MAP*, they are checked.

There are other technical issues to be handled. As for the construction associated to the automaton $\mathcal{A}_{y_b^{\mathsf{a}}}$ for an abstract predictor clock y_b^{a}, the automaton $\mathcal{A}_{x_b^{\mathsf{a}}}$ uses the special proposition p_∞, and keeps track in its control state of the guessed type (call, return, or internal symbol) of the next input symbol in order to check whether the current input position is the last one of the current *MAP*. Moreover, we have to ensure that the lower-bound obligations $\succ\ell$ (resp., the fresh upper-bound obligations (*first*, $\prec u$)) at the current position i are eventually checked, i.e., for the current *MAP* ν, $\mathit{infix}_b(\nu, i)$ eventually visits a position j where the constraint $x_b^{\mathsf{a}} \succ \ell$ (resp., $x_b^{\mathsf{a}} \prec u$) is used. For this, $\mathcal{A}_{x_b^{\mathsf{a}}}$ keeps track in its control state of the guessed interval constraint $x_b^{\mathsf{a}} \in I$ used by $\mathcal{A}$ on reading the next input symbol, and whether the guessed next input symbol is b. Moreover, for each lower-bound obligation $\succ\ell$ (resp., fresh upper-bound obligations $(\mathit{first}, \prec u)$), $\mathcal{A}_{x_b^{\mathsf{a}}}$ exploits a Büchi component ensuring that along an infinite *MAP* ν, *either* there are infinitely many occurrences of b-positions, *or* there are infinitely many occurrences of positions where an interval constraint $x_b^{\mathsf{a}} \in I$ consistent with $x_b^{\mathsf{a}} \succ \ell$ (resp., $x_b^{\mathsf{a}} \prec u$) is used, *or* there are infinitely many positions in ν where the set of obligations does not contain $\succ\ell$ (resp., $(\mathit{first}, \prec u)$).

We now provide the formal definition of $\mathcal{A}_{x_b^{\mathsf{a}}}$. To this end, we need additional notation. An *obligation set* O (for the fixed recorder event x_b^{a}) is a set consisting of lower-bound obligations $\succ\ell$ and upper-bound obligations $(f, \prec u)$, where $f \in \{\mathit{live}, \mathit{first}\}$, such that $x_b^{\mathsf{a}} \succ \ell$ and $x_b^{\mathsf{a}} \prec u$ are associated to interval constraints $x_b^{\mathsf{a}} \in I$ of $\mathcal{A}$, and $(f, \prec u), (f', \prec u) \in O$ implies $f = f'$. For an obligation set O, $\mathit{live}(O)$ consists of the live upper-bound obligations of O. Given an obligation set O and an interval constraint $x_b^{\mathsf{a}} \in I$ of $\mathcal{A}$, we say that $x_b^{\mathsf{a}} \in I$ is *consistent with* O if one of the following holds:

- $I = \{\vdash\}$ and $O = \mathit{live}(O)$;
- $x_b^{\mathsf{a}} \in I \equiv x_b^{\mathsf{a}} \succ \ell \wedge x_b^{\mathsf{a}} \prec u$, $\succ\ell \in O$ and $(f, \prec u) \in O$ for some $f \in \{\mathit{first}, \mathit{live}\}$.

Let $\Phi(x_b^{\mathsf{a}})$ be the set of interval constraints of the form $x_b^{\mathsf{a}} \in I$ used by $\mathcal{A}$. A *check set* H is a subset of $\{\mathit{call}, \mathit{ret}, \mathit{int}, p_\infty, b\} \cup \Phi(x_b^{\mathsf{a}})$ such that $H \cap \{\mathit{call}, \mathit{ret}, \mathit{int}\}$ and $H \cap \Phi(x_b^{\mathsf{a}})$ are singletons. We say that H and an obligation set O are *consistent* if the unique interval constraint in H is consistent with O. For an interval constraint $x_b^{\mathsf{a}} \in I$ used by $\mathcal{A}$, let $\mathit{con}(I)$ be the constraint over C_{new} defined as follows: $\mathit{con}(I) = \top$ if $I = \{\vdash\}$, and $\mathit{con}(I) = z_{\succ\ell} \succ \ell \wedge z_{\prec u} \prec u$ if $x_b^{\mathsf{a}} \in I \equiv x_b^{\mathsf{a}} \succ \ell \wedge x_b^{\mathsf{a}} \prec u$. The nested VPTA $\mathcal{A}_{x_b^{\mathsf{a}}}$ is given by

$$\mathcal{A}_{x_b^{\mathsf{a}}} = (\Sigma, Q', Q_0', C \setminus \{x_b^{\mathsf{a}}\} \cup C_{st} \cup C_{new}, (\Gamma \times \Gamma') \cup \{\mathit{bad}, \top\}, \Delta', \mathcal{F}')$$

where the set Q' of states consists of triples of the form (q, O, H) such that q is a state of $\mathcal{A}$, O is an obligation set, H is a check set, and H and O are consistent. The set Q_0' of initial states consists of states of the form $(q_0, \emptyset, H)$ such that $q_0 \in Q_0$ (initially there are no obligations). Note that for an initial state $(q_0, \emptyset, H)$, $(x_b^{\mathsf{a}} \in \{\vdash\}) \in H$ (H and the obligation set $\emptyset$ are consistent). The stack alphabet is $(\Gamma \times \Gamma') \cup \{\mathit{bad}, \top\}$, where Γ' is the set of pairs (O, H) such that O is an obligation set, H is a check set, and H and O are consistent.

We now define the transition function Δ'. To this end, we first define a predicate *AbsP* over tuples of the form $((O, H), a, x_b^{\mathsf{a}} \in I, \mathit{Res}, (O', H'))$ where $(O, H), (O', H')$ are pairs of obligation sets and check sets, $a \in \Sigma$, $x_b^{\mathsf{a}} \in I$ is a constraint of $\mathcal{A}$, and $\mathit{Res} \subseteq C_{new}$. Intuitively, O (resp., H) represents the obligation set (resp., check set) at the current position i of the input, a is the input symbol associated with position i, $x_b^{\mathsf{a}} \in I$ is the constraint on x_b^{a} used by $\mathcal{A}$ at position i, *Res* is the set of new standard clocks reset by $\mathcal{A}_{x_b^{\mathsf{a}}}$ on reading a, and O' (resp., H') represents the obligation set (resp., check set) at the position j following i along the *MAP* visiting i (if i is a call, then j is the matching-return of i). Formally, $\mathit{AbsP}((O, H), a, x_b^{\mathsf{a}} \in I, \mathit{Res}, (O', H'))$ is true iff the following holds:

1. ($p_\infty \in H$ iff $p_\infty \in H'$), $a \in \Sigma_{call}$ (resp., $a \in \Sigma_{ret}$, resp. $a \in \Sigma_{int}$) implies $\mathit{call} \in H$ (resp., $\mathit{ret} \in H$, resp., $\mathit{int} \in H$);
2. $(x_b^{\mathsf{a}} \in I) \in H$, and H and O are consistent (resp., H' and O' are consistent);
3. Case $b = a$: $b \in H$, $(x_b^{\mathsf{a}} \in \{\vdash\}) \notin H'$ and *Res* is a subset of C_{new} such that $z_{\prec u} \in \mathit{Res}$ implies $(\mathit{live}, \prec u) \notin O$. Moreover, $O' = \mathit{live}(O) \cup O''$, where O'' is obtained from *Res* by adding for each clock $z_{\succ\ell} \in \mathit{Res}$ (resp., $z_{\prec u} \in \mathit{Res}$), the obligation $\succ\ell$ (resp., the fresh obligation $(\mathit{first}, \prec u)$);
4. Case $b \neq a$: $b \notin H$, $\mathit{Res} = \emptyset$. If $I = \{\vdash\}$, then $O' = O = \mathit{live}(O)$ and $(x_b^{\mathsf{a}} \in \{\vdash\}) \in H'$. Otherwise, let $x_b^{\mathsf{a}} \in I \equiv x_b^{\mathsf{a}} \succ \ell \wedge x_b^{\mathsf{a}} \prec u$. Then, $(x_b^{\mathsf{a}} \notin \{\vdash\}) \in H'$, and O' is any obligation set obtained from O by *optionally* removing the obligation $\succ\ell$ (by Condition 2, $\succ\ell \in O$), and/or by *optionally* removing the obligation $(\mathit{first}, \prec u)$ if $(\mathit{first}, \prec u) \in O$.

Condition 1 requires that the Boolean value of proposition p_∞ is invariant along the positions of a *MAP*, and the current check set is consistent with the type (call, return, or internal symbol) of the current input symbol. Condition 2 requires that the current check set is consistent with the constraint $x_b^{\mathsf{a}} \in I$ currently used by $\mathcal{A}$. Conditions 3 and 4 provide the rules for updating the obligations on moving to the abstract next position along the current *MAP* and for resetting new clocks on reading the current input symbol a. Note that if $I = \{\vdash\}$ and $b \neq a$, then the current obligation set must contain only live upper-bound obligations, and $(x_b^{\mathsf{a}} \in \{\vdash\}) \in H'$.

Given a state (q, O, H) of $\mathcal{A}_{x_b^{\mathsf{a}}}$, we say that (q, O, H) is *terminal* if the following holds: if $x_b^{\mathsf{a}} \in I$ is the unique constraint associated with the check set H and $x_b^{\mathsf{a}} \in I \equiv x_b^{\mathsf{a}} \succ \ell \wedge x_b^{\mathsf{a}} \prec u$, then $O \setminus \{\succ\ell, (\mathit{first}, \prec u)\} = \mathit{live}(O)$. Intuitively, terminal states are associated with input positions i such that i is the last position of the related *MAP*.
The transition function Δ' of $\mathcal{A}_{x_b^{\mathsf{a}}}$ is defined as follows. Recall that we can assume that each clock constraint of $\mathcal{A}$ is of the form $\theta \wedge x_b^{\mathsf{a}} \in I$, where θ does not contain occurrences of x_b^{a}.

Push transitions: for each push transition $q \xrightarrow{a, \theta \wedge x_b^{\mathsf{a}} \in I, \mathit{Res}, \mathit{push}(\gamma)} q'$ of $\mathcal{A}$, we have the push transitions $(q, O, H) \xrightarrow{a, \theta \wedge \mathit{con}(I), \mathit{Res} \cup \mathit{Res}', \mathit{push}(\gamma')} (q', O', H')$ such that $b \in H$ iff $a = b$, and

1. Case $\gamma' \neq \mathit{bad}$. Then, $\gamma' = (\gamma, O_{ret}, H_{ret})$ and

- *AbsP*$((O,H), a, x^{\mathsf{a}}_b \in I, Res', (O_{ret}, H_{ret}))$. Moreover, if $ret \in H'$ then $H_{ret} = H'$ and $O' = O_{ret}$; otherwise, $p_\infty \notin H'$ and O' consists of the live obligations $(live, \prec u)$ such that $(f, \prec u) \in O_{ret}$ for some $f \in \{live, first\}$.

2. Case $\gamma' = bad$: $call \in H$, $(x^{\mathsf{a}}_b \in I) \in H$, state (q, O, H) is terminal, $p_\infty \in H$, $p_\infty \in H'$, $ret \notin H'$, $O' = \emptyset$, and $Res' = \emptyset$.

Note that if $I \neq \{\vdash\}$, then the constraint $x^{\mathsf{a}}_b \in I$ is checked by the constraint $con(I)$ (recall that if $I = \{\vdash\}$, then $con(I) = \top$). The push transitions of point 1 consider the case where $\mathcal{A}_{x^{\mathsf{a}}_b}$ guesses that the current call position i_c has a matching return i_r. The set of obligations and the check state for the next abstract position i_r along the current *MAP* are pushed on the stack for recovering at the matching-return i_r. Moreover, if $\mathcal{A}_{x^{\mathsf{a}}_b}$ guesses that the next position $i_c + 1$ is not i_r (i.e., $ret \notin H'$), then all the upper-bound obligations in O_{ret} are propagated as live obligations at the next position $i_c + 1$ (note that the *MAP* visiting $i_c + 1$ starts at $i_c + 1$, terminates at $i_r - 1$, and does not satisfy proposition p_∞). The push transitions of point 2 consider the case where $\mathcal{A}_{x^{\mathsf{a}}_b}$ guesses that the current call position i_c has no matching return i_r, i.e., i_c is the last position of the current *MAP*. In this case, $\mathcal{A}_{x^{\mathsf{a}}_b}$ pushes the symbol *bad* on the stack and the transition relation is consistently updated.

Internal transitions: for each internal transition $q \xrightarrow{a,\theta\wedge x^{\mathsf{a}}_b \in I, Res} q'$ of $\mathcal{A}$, we add the internal transitions $(q, O, H) \xrightarrow{a,\theta\wedge con(I), Res\cup Res'} (q', O', H')$ such that $b \in H$ iff $a = b$, and

1. Case $ret \in H'$: $int \in H$, $(x^{\mathsf{a}}_b \in I) \in H$, state (q, O, H) is terminal, and $Res' = \emptyset$.
2. Case $ret \notin H'$: *AbsP*$((O,H), a, x^{\mathsf{a}}_b \in I, Res', (O', H'))$.

In the former case, $\mathcal{A}_{x^{\mathsf{a}}_b}$ guesses that the current internal position i is the last one of the current *MAP* ($ret \in H'$); in the later, the *MAP* visits the next non-return position $i + 1$.

Pop transitions: for each pop transition $q \xrightarrow{a,\theta\wedge x^{\mathsf{a}}_b \in I, Res, pop(\gamma)} q' \in \Delta_r$, we have the pop transitions $(q, O, H) \xrightarrow{a,\theta\wedge con(I), Res\cup Res', pop(\gamma')} (q', O', H')$ such that $b \in H$ iff $a = b$, and

1. Case $\gamma \neq \top$: $ret \in H$, $\gamma' = (\gamma, (O, H))$, and $(x^{\mathsf{a}}_b \in I) \in H$. If $ret \notin H'$, then *AbsP*$((O,H), a, x^{\mathsf{a}}_b \in I, Res', (O', H'))$; otherwise, (q, O, H) is a terminal state and $Res' = \emptyset$.
2. Case $\gamma = \top$: $ret \in H$, $I = \{\vdash\}$, $\gamma' = \top$, $p_\infty \in H$, $p_\infty \in H'$, and $O = \emptyset$. If $ret \notin H'$, then *AbsP*$((O,H), a, x^{\mathsf{a}}_b \in I, Res', (O', H'))$; otherwise, $Res' = \emptyset$ and $O' = \emptyset$.

If $\gamma \neq \top$, then the current return position has a matched-call. Otherwise, the current position is also the initial position of the associated *MAP*.

Finally, the generalized Büchi condition $\mathcal{F}'$ of $\mathcal{A}_{x^{\mathsf{a}}_b}$ is defined as follows. For each Büchi component F of $\mathcal{A}$, $\mathcal{A}_{x^{\mathsf{a}}_b}$ has the Büchi component consisting of the states (q, O, H) such that $q \in F$. Moreover, $\mathcal{A}_{x^{\mathsf{a}}_b}$ has an additional component consisting of the states (q, O, H) such that $p_\infty \in H$. Such a component ensures that the guesses about the matched calls are correct. Finally, for each lower-bound constraint $x^{\mathsf{a}}_b \succ \ell$ (resp., upper-bound constraint $x^{\mathsf{a}}_b \prec u$) of $\mathcal{A}$, $\mathcal{A}_{x^{\mathsf{a}}_b}$ has a Büchi component consisting of the states (q, O, H) such that

- $p_\infty \in H$, and *either* $b \in H$, *or* the unique constraint in H is equivalent to $x^{\mathsf{a}}_b \succ \ell \wedge x^{\mathsf{a}}_b \prec u'$ for some upper-bound u', *or* $\succ\ell \notin O$;
- (resp., $p_\infty \in H$, and *either* $b \in H$, *or* the unique constraint in H is equivalent to $x^{\mathsf{a}}_b \succ \ell' \wedge x^{\mathsf{a}}_b \prec u$ for some lower-bound ℓ', *or* $(first, \prec u) \notin O$).

The Büchi component above ensures that along an infinite *MAP* ν, *either* there are infinitely many occurrences of b-positions, *or* there are infinitely many occurrences of positions where a constraint $x^{\mathsf{a}}_b \in I$ consistent with $x^{\mathsf{a}}_b \succ \ell$ (resp., $x^{\mathsf{a}}_b \prec u$) is used, *or* there are infinitely many positions in ν where the set of obligations does not contain $\succ\ell$ (resp., $(first, \prec u)$).

Appendix B. Removal of caller event-clocks in nested VPTA

Theorem 2 *(Removal of caller event-clocks). Given a generalized Büchi nested VPTA $\mathcal{A}$ with set of event clocks C and a caller event-clock $x^{\mathsf{c}}_b \in C$, one can construct in singly exponential time a generalized Büchi nested VPTA $\mathcal{A}_{x^{\mathsf{c}}_b}$ with set of event clocks $C \setminus \{x^{\mathsf{c}}_b\}$ such that $\mathcal{L}_T(\mathcal{A}_{x^{\mathsf{c}}_b}) = \mathcal{L}_T(\mathcal{A})$ and $K_{\mathcal{A}_{x^{\mathsf{c}}_b}} = K_{\mathcal{A}}$. Moreover, $\mathcal{A}_{x^{\mathsf{c}}_b}$ has $O(n \cdot 2^{O(p)})$ states and $m + O(p)$ clocks, where n is the number of $\mathcal{A}$-states, m is the number of standard $\mathcal{A}$-clocks, and p is the number of* event-clock *atomic constraints on x^{c}_b used by $\mathcal{A}$.*

Fix a generalized Büchi nested VPTA $\mathcal{A} = (\Sigma, Q, Q_0, C \cup C_{st}, \Gamma \cup \{\top\}, \Delta, \mathcal{F})$ such that $x^{\mathsf{c}}_b \in C$. We construct a generalized Büchi nested VPTA $\mathcal{A}_{x^{\mathsf{c}}_b}$ equivalent to $\mathcal{A}$ whose set of event clocks is $C \setminus \{x^{\mathsf{c}}_b\}$, and whose set of standard clocks is $C_{st} \cup C_{new}$, where C_{new} consists of the fresh standard clocks $z_{\succ\ell}$ (resp., $z_{\prec u}$) for each lower-bound constraint $x^{\mathsf{c}}_b \succ \ell$ (resp., upper-bound constraint $x^{\mathsf{c}}_b \prec u$) of $\mathcal{A}$ involving x^{c}_b. Since a caller path from a position j consists only of call positions except position j (if $j \notin \Sigma_{call}$), we assume that $b \in \Sigma_{call}$ (the case where $b \notin \Sigma_{call}$ is straightforward).

The main idea of the construction is that whenever b occurs at a call position i_c of the input word, the simulating automaton $\mathcal{A}_{x^{\mathsf{c}}_b}$ guesses the set of lower-bound and upper-bound constraints on x^{c}_b that will be used by $\mathcal{A}$ along the *MAP* ν having i_c as caller. Note that such a *MAP* is empty if $i_c + 1$ is a return, and starts at position $i_c + 1$ otherwise.

First, let us consider lower-bound constraints $x^{\mathsf{c}}_b \succ \ell$. Assume that b occurs at a call position i_c of the input word and $i_c + 1$ is not a return. Let ν be the *MAP* starting at position $i_c + 1$. Then, $\mathcal{A}_{x^{\mathsf{c}}_b}$ guesses the set of lower-bound constraints $x^{\mathsf{c}}_b \succ \ell$ that will be used by $\mathcal{A}$ along ν. For each of such guessed constraints $x^{\mathsf{c}}_b \succ \ell$, $\mathcal{A}_{x^{\mathsf{c}}_b}$ resets the associated new clock $z_{\succ\ell}$, and moves to the next position by carrying in the control state the new set of lower-bound obligations $\succ\ell$. On visiting the positions j of ν, $\mathcal{A}_{x^{\mathsf{c}}_b}$ checks that the guess is correct by verifying that for the current lower-bound constraint $x^{\mathsf{c}}_b \succ \ell'$ used by $\mathcal{A}$, $\succ\ell'$ is in the current set of obligations, and constraint $z_{\succ\ell'} \succ \ell'$ holds. Moreover, at position j, $\mathcal{A}_{x^{\mathsf{c}}_b}$ guesses whether the constraint $x^{\mathsf{c}}_b \succ \ell'$ will be again used along ν, or not. In the first case, the obligation $\succ\ell'$ is kept, otherwise, it is discarded. If a new call n_c occurs along ν before the last position of ν, then all the *caller paths* starting from the positions $h \in [n_c + 1, n_r - 1]$, where n_r is the matching return of n_c (i.e., n_r is the position following n_c along ν), visit positions i_c and n_c ($n_c > i_c$). Thus, the fulfillment of a lower-bound constraint $x^{\mathsf{c}}_b \succ \ell$ asserted at a position $h \in [n_c + 1, n_r - 1]$ always implies the fulfillment of the same constraint when asserted at a position $j \geq i_r$ of ν. Therefore, if b occurs at the new call-position n_c, the clocks $z_{\succ\ell}$ associated with the guessed lower-bound constraints $x^{\mathsf{c}}_b \succ \ell$ used by $\mathcal{A}$ along the *MAP* having n_c as caller (such a *MAP* starts at position $n_c + 1$ and leads to position $n_r - 1$) can be safely reset.
Overall, at each position i, $\mathcal{A}_{x^{\mathsf{c}}_b}$ keeps track in its control state whether the caller path from i visits a b-position preceding i, or not. In the first case, $\mathcal{A}_{x^{\mathsf{c}}_b}$ also keeps track in its control state of the set of obligations associated with the guessed lower-bound constraints on x^{c}_b which will be used by $\mathcal{A}$ in the suffix of the current *MAP* from position i. In the second case, there are no obligations. Whenever a matched call $i_c \geq i$ occurs along ν, the guessed lower-bound obligations (if any) for the matching return i_r of i_c are pushed on the stack in order to be recovered at position i_r. Moreover, if $i_c + 1$ is not a return (i.e., $i_r \neq i_c + 1$), and either we are in the first case or i_c is a b-position, then $\mathcal{A}_{x^{\mathsf{c}}_b}$ guesses the set L of lower-bound constraints which will be used by $\mathcal{A}$ in the finite *MAP* starting at position $i_c + 1$, and moves to the next position by carrying in its control state the obligations associated with L. Additionally, if i_c is a b-position, then for each $x^{\mathsf{c}}_b \succ \ell \in L$, the associated new clock $z_{\succ\ell}$ is reset.

The situation for upper-bound constraints $x^{\mathsf{c}}_b \prec u$ is dual. In this case, as in the proof of Theorem 1, we distinguish between fresh upper-bound obligations (*first*,$\prec u$) and live upper-bound obligations (*live*,$\prec u$). Fresh upper-bound obligations (*first*, $\prec u$) always refer to guesses done along the current *MAP* and differently from the live upper-bound obligations, may be removed, when along the current *MAP*, they are checked. Live upper-bound obligations (*live*,$\prec u$) are propagated from the caller *MAP* to the called *MAP*. They are used by $\mathcal{A}_{x^{\mathsf{c}}_b}$ to remember that at a matched b-call position i_c along the current *MAP* with matching return $i_r > i_c + 1$, if the upper-bound constraint $x^{\mathsf{c}}_b \prec u$ is guessed to be used by $\mathcal{A}$ along the finite *MAP* ν' having as caller i_c (ν' starts at $i_c + 1$ and ends at $i_r - 1$), and the guessed set of obligations for the matching return i_r already contains an obligation (f,$\prec u$), then the clock $z_{\prec u}$ must not be reset. This is safe since the fulfillment of an upper-bound constraint $x^{\mathsf{c}}_b \prec u$ asserted at a position $j \geq i_r$ along ν always implies the fulfillment of the same constraint when asserted at a position h of the *MAP* ν'.

The formal definition of $\mathcal{A}_{x^{\mathsf{c}}_b}$ is similar to that of the nested VPTA $\mathcal{A}_{x^{\mathsf{a}}_b}$ exploited in the proof of Theorem 1. Thus, here, we omit the details of the construction.

Appendix C. Proof of Theorem 8

We focus on the case of finite timed words (the case of infinite timed words is similar). Let $\mathcal{F}$ be the fragment of EC_NTL obtained by disallowing the timed non-regular modalities $\lhd^{\mathsf{a}}_I$, $\lhd^{\mathsf{c}}_I$, and $\rhd^{\mathsf{a}}_I$. Theorem 8 (for finite timed words) directly follows from the following result.

Proposition 6. *Let $\mathcal{P} = \{call, ret\}$ and $\mathcal{L}_T$ be the timed language consisting of the finite timed words of the form (σ, τ) such that σ is a well-matched word of the form $\{call\}^n \cdot \{ret\}^n$ for some $n > 0$, and there is a call position i_c of σ such that $\tau_{i_r} - \tau_{i_c} = 1$, where i_r is the matching-return of i_c in σ. Then, $\mathcal{L}_T$ can be expressed in EC_NTL but not in $\mathcal{F}$.*

Proof. The language $\mathcal{L}_T$ is definable by the following EC_NTL formula

$$call \wedge \bigcirc^{\mathsf{a}}(\neg \bigcirc^{\mathsf{g}} \top) \wedge \Box^{\mathsf{g}}\neg int \wedge \neg \Diamond^{\mathsf{g}}\,(ret \wedge \Diamond^{\mathsf{g}} call) \wedge \Diamond^{\mathsf{g}}(call \wedge \rhd^{\mathsf{a}}_{[1,1]} \top)$$

We show that no formula in $\mathcal{F}$ can capture the language $\mathcal{L}_T$. For a formula φ of $\mathcal{F}$, let $d(\varphi)$ be the nesting depth of the *unary* temporal modalities in φ. For all $H \geq 1$, let w^H_{good} and w^H_{bad} be the well-matched timed words over $\Sigma_{\mathcal{P}}$ of length $4H+2$ and $4H$, respectively, defined as

- $w^H_{good} = (\{call\}, \frac{1}{2H+1}) \ldots (\{call\}, \frac{2H+1}{2H+1}) \cdot (\{ret\}, 1 + \frac{1}{2H+1}) \ldots (\{ret\}, 1 + \frac{2H+1}{2H+1})$;

- w^H_{bad} is obtained from w^H_{good} by removing the call-position H and its matching-return position $3H+1$.

By construction, position H of w^H_{good} is the unique call-position i_c of w^H_{good} such that the time distance between the matching-return of i_c and i_c is exactly 1. Hence, for all $H \geq 1$, $w^H_{good} \in \mathcal{L}_T$ and $w^H_{bad} \notin \mathcal{L}_T$. We prove that for all $H \geq 1$ and formula φ in $\mathcal{F}$ such that $d(\varphi) < H$, w^H_{good} is a model of φ iff w^H_{bad} is a model of φ. Hence, $\mathcal{L}_T$ is not expressible in $\mathcal{F}$ and the result follows. For this, we first prove the following claim.

Claim 1: Let $H \geq 1$, $0 \leq k \leq H$, and $\varphi \in \mathcal{F}$ with $d(\varphi) \leq H-k$. Then, the following holds:

1. for all $i, j \in [H-k, H+k]$, $(w^H_{good}, i) \models \varphi$ iff $(w^H_{good}, j) \models \varphi$;
2. for all $i, j \in [3H+1-k, 3H+1+k]$, $(w^H_{good}, i) \models \varphi$ iff $(w^H_{good}, j) \models \varphi$.

Proof of Claim 1: Let $H \geq 1$, $0 \leq k \leq H$, and $\varphi \in \mathcal{F}$ with $d(\varphi) \leq H-k$. We prove the implication $(w^H_{good}, i) \models \varphi \rightarrow (w^H_{good}, j) \models \varphi$ in Properties 1 and 2 (the converse implication being similar). The proof is by induction on the structure of the formula and the nesting depth $d(\varphi)$. By construction, for all $\ell \in [H-k, H+k]$ (resp., $\ell \in [3H+1-k, 3H+1+k]$), ℓ is a call position (resp., return position) of w^H_{good}. Hence, the base case holds, while the cases where the root modality of φ is a Boolean connective directly follow from the induction hypothesis. For the other cases, we focus on Property 1 (Property 2 being similar). Thus, let $i, j \in [H-k, H+k]$. For a call-position $\ell \in [0, 2H]$ in w^H_{good}, let $ret(\ell)$ be the matching-return position. Note that $ret(\ell) = 4H+1-\ell$. Since $\varphi \in \mathcal{F}$, we consider the following cases:

- $\varphi = \varphi_1 \mathsf{U}^{\mathsf{g}} \varphi_2$. Assume that $(w^H_{good}, i) \models \varphi$. Hence, there is $\ell \in [i, 4H+1]$ such that $(w^H_{good}, \ell) \models \varphi_2$ and $(w^H_{good}, \ell') \models \varphi_1$ for all $\ell' \in [i, \ell-1]$. We distinguish two cases:
  - $\ell > j$. By the induction hypothesis, either $\ell = i$ and $(w^H_{good}, j) \models \varphi_2$, or $\ell > i$ and for all positions p between i and j, $(w^H_{good}, p) \models \varphi_1$. It follows that $(w^H_{good}, j) \models \varphi$.
  - $\ell \leq j$. Hence, $\ell \in [i, j]$. By the induction hypothesis, $(w^H_{good}, j) \models \varphi_2$, and the result follows.

- $\varphi = \varphi_1 \mathsf{S}^{\mathsf{g}} \varphi_2$: this case is similar to the previous one.
- $\varphi = \varphi_1 \mathsf{U}^{\mathsf{a}} \varphi_2$. Assume that $(w^H_{good}, i) \models \varphi$. Since position i is a call, by construction, either $(w^H_{good}, i) \models \varphi_2$, or $(w^H_{good}, i) \models \varphi_1$ and $(w^H_{good}, ret(i)) \models \varphi_2$. Since $ret(i), ret(j) \in [3H+1-k, 3H+1+k]$, by the induction hypothesis on Properties 1 and 2, either $(w^H_{good}, j) \models \varphi_2$, or $(w^H_{good}, j) \models \varphi_1$ and $(w^H_{good}, ret(j)) \models \varphi_2$. Hence, $(w^H_{good}, j) \models \varphi$.
- $\varphi = \varphi_1 \mathsf{S}^{\mathsf{a}} \varphi_2$: this case is similar to the previous one.
- $\varphi = \varphi_1 \mathsf{S}^{\mathsf{c}} \varphi_2$: since $i \in [H-k, H+k]$, by construction, $(w^H_{good}, i) \models \varphi_1 \mathsf{S}^{\mathsf{c}} \varphi_2$ iff $(w^H_{good}, i) \models \varphi_1 \mathsf{S}^{\mathsf{g}} \varphi_2$, and the result follows from the case for modality S^{g}.
- $\varphi = \bigcirc^{\mathsf{g}} \varphi_1$. Let $(w^H_{good}, i) \models \varphi$. Hence, $(w^H_{good}, i+1) \models \varphi_1$. Since $d(\varphi) \geq 1$ and $d(\varphi) \leq H-k$, we have that $k+1 \leq H$, $d(\varphi_1) \leq H-(k+1)$, and $i+1, j+1 \in [H-(k+1), H+(k+1)]$. Thus, by induction hypothesis on $d(\varphi_1)$, $(w^H_{good}, j) \models \varphi$.
- $\varphi = \ominus^{\mathsf{g}} \varphi_1$: this case is similar to the previous one.
- $\varphi = \bigcirc^{\mathsf{a}} \varphi_1$: let $(w^H_{good}, i) \models \varphi$. Since position i is a call, by construction, $(w^H_{good}, ret(i)) \models \varphi_1$. Since $ret(i), ret(j) \in [3H+1-k, 3H+1+k]$, by the induction hypothesis on Property 2, it follows that $(w^H_{good}, ret(j)) \models \varphi_1$. Hence, $(w^H_{good}, j) \models \varphi$.
- $\varphi = \ominus^{\mathsf{a}} \varphi_1$: this case is similar to the previous one.
- $\varphi = \ominus^{\mathsf{c}} \varphi_1$: since $i \in [H-k, H+k]$, by construction, $(w^H_{good}, i) \models \ominus^{\mathsf{c}} \varphi_1$ iff $(w^H_{good}, i) \models \ominus^{\mathsf{g}} \varphi_1$, and the result follows from the case for modality $\ominus^{\mathsf{g}}$.
- $\varphi = \rhd^{\mathsf{g}}_I \varphi_1$: for all positions $\ell \in [0, 4H+1]$, let τ_ℓ be the timestamp of w^H_{good} at position ℓ. Moreover, if $\ell \in [0, 2H]$, let $m(\ell) := 2H+1+\ell$. By construction, $\tau_{m(\ell)} - \tau_\ell = 1$. Assume that $(w^H_{good}, i) \models \varphi$. Hence, there is $\ell \in [i+1, 4H+1]$ such that $(w^H_{good}, \ell) \models \varphi_1$, $\tau_\ell - \tau_i \in I$ and $(w^H_{good}, \ell') \not\models \varphi_1$ for all $\ell' \in [i+1, \ell-1]$. By construction, one of the following cases occurs:
  - $\tau_\ell - \tau_i = 1$: by construction, $\ell = m(i)$. Hence, $\ell \in [3H+1-k, 3H+1+k]$. We show that this case cannot occur. Since $d(\varphi) \geq 1$ and $d(\varphi) \leq H-k$, we have that $k+1 \leq H$, $d(\varphi_1) \leq H-(k+1)$, and $\ell, \ell-1 \in [3H+1-(k+1), 3H+1+(k+1)]$. Thus, by the induction hypothesis on $d(\varphi_1)$, $(w^H_{good}, \ell) \models \varphi_1$ iff $(w^H_{good}, \ell-1) \models \varphi_1$. By hypothesis, $(w^H_{good}, \ell) \models \varphi_1$ and $(w^H_{good}, \ell-1) \not\models \varphi_1$, a contradiction.
  - $1 < \tau_\ell - \tau_i < 2$: hence, $\ell > m(i) > i$ and $(w^H_{good}, m(i)) \not\models \varphi_1$. Since, $m(i) \in [3H+1-k, 3H+1+k]$, by the induction hypothesis, it follows that $\ell > 3H+1+k \geq m(j)$ which entails that $1 < \tau_\ell - \tau_j < 2$. It follows that $\tau_\ell - \tau_j \in I$, and

by the induction hypothesis on $d(\varphi_1)$, we easily obtain that for all the positions p between i and j, $(w^H_{good}, p) \not\models \varphi_1$. It follows that $(w^H_{good}, j) \models \rhd^{\mathsf{g}}_I \varphi_1$.

  - $0 < \tau_\ell - \tau_i < 1$ and ℓ is a return-position: hence, $i < \ell < m(i)$. By the induction hypothesis on $d(\varphi_1)$, we deduce that $\ell \notin [3H+1-k, 3H+1+k]$ (otherwise, $(w^H_{good}, \ell-1) \models \varphi_1$). It follows that $j < \ell < m(j)$ which entails that $0 < \tau_\ell - \tau_j < 1$. Hence, $\tau_\ell - \tau_j \in I$, and by induction hypothesis on $d(\varphi_1)$, $(w^H_{good}, j) \models \rhd^{\mathsf{g}}_I \varphi_1$ holds.
  - $0 < \tau_\ell - \tau_i < 1$ and ℓ is a call-position: if $\ell \in [H-(k+1), H+(k+1)]$, then by the induction hypothesis on $d(\varphi_1)$, we have that $(w^H_{good}, j+1) \models \varphi_1$, and since $0 < \tau_{j+1} - \tau_j < 1$, we obtain that $(w^H_{good}, j) \models \rhd^{\mathsf{g}}_I \varphi_1$. On the other hand, if $\ell > H+(k+1)$, by the induction hypothesis, we deduce that for all positions p between i and j, $(w^H_{good}, p) \not\models \varphi_1$. Thus, since by construction $0 < \tau_\ell - \tau_j < 1$, we conclude that $(w^H_{good}, j) \models \rhd^{\mathsf{g}}_I \varphi_1$.

- $\varphi = \lhd^{\mathsf{g}}_I \varphi_1$: this case is similar to the previous one (proving Claim 1). $\square$

Let $H \geq 1$. For each position i of w^H_{bad} (note that $i \in [0, 4H-1]$), we denote by $H(i)$ the associated position in w^H_{good}, i.e. the unique position j of w^H_{good} such that $w^H_{bad}(i) = w^H_{good}(j)$. By Claim 1, we deduce the following Claim 2. Since $H(0) = 0$, Claim 2 entails the result, i.e. for all $H \geq 1$ and formulas φ in $\mathcal{F}$ such that $d(\varphi) < H$, $(w^H_{good}, 0) \models \varphi$ iff $(w^H_{bad}, 0) \models \varphi$.

Claim 2: Let $H \geq 1$ and $\varphi \in \mathcal{F}$ with $d(\varphi) < H$. Then, for all $i \in [0, 4H-1]$, $(w^H_{bad}, i) \models \varphi$ iff $(w^H_{good}, H(i)) \models \varphi$

Proof of Claim 2: Let $H \geq 1$ and $\varphi \in \mathcal{F}$ with $d(\varphi) < H$. We prove by structural induction on φ that for all $i \in [0, 4H-1]$, $(w^H_{bad}, i) \models \varphi$ iff $(w^H_{good}, H(i)) \models \varphi$. By construction, for all $i \in [0, 4H-1]$, $w^H_{bad}(i) = w^H_{good}(H(i))$. Hence, the base case holds, while the cases where the root modality of φ is a Boolean connective directly follow from the induction hypothesis. Since $\varphi \in \mathcal{F}$, it remains to consider the following cases:

- $\varphi = \varphi_1 \mathsf{U}^{\mathsf{g}} \varphi_2$. Assume that $(w^H_{good}, H(i)) \models \varphi$. Hence, there is $\ell \in [H(i), 4H+1]$ such that $(w^H_{good}, \ell) \models \varphi_2$ and $(w^H_{good}, \ell') \models \varphi_1$ for all $\ell' \in [H(i), \ell-1]$. Assume that $\ell \neq H(p)$ for all positions p of w^H_{bad} (the other case being simpler). Hence, $\ell \in \{H, 3H+1\}$. Let $\wp \in [0, 4H-1]$ such that $H(\wp) = \ell - 1$. Since $d(\varphi) < H$, by Claim 1, $(w^H_{good}, \ell-1) \models \varphi_2$. Thus, since $i \leq \wp$ and $H(p) \in [H(i), H(\wp)-1]$ for all $p \in [i, \wp-1]$, by the induction hypothesis, it follows that $(w^H_{bad}, i) \models \varphi$. The converse implication $(w^H_{bad}, i) \models \varphi \Rightarrow (w^H_{good}, H(i)) \models \varphi$ is similar.
- $\varphi = \varphi_1 \mathsf{S}^{\mathsf{g}} \varphi_2$: this case is similar to the previous one.
- $\varphi = \varphi_1 \mathsf{U}^{\mathsf{a}} \varphi_2$ or $\varphi = \varphi_1 \mathsf{S}^{\mathsf{a}} \varphi_2$. By construction, for all $i \in [0, 4H-1]$, the *MAP* of w^H_{bad} visiting position i consists of the positions i and $mt(i)$, where $mt(i)$ is the matching-return of i if i is a call, and the matching-call of i otherwise. Moreover, the *MAP* of w^H_{good} visiting position $H(i)$ consists of the positions $H(i)$ and $H(mt(i))$. Hence, the result for the abstract until and since modalities, directly follows from the induction hypothesis.
- $\varphi = \varphi_1 \mathsf{S}^{\mathsf{c}} \varphi_2$: let $i \in [0, 4H-1]$. By construction, $(w^H_{bad}, i) \models \varphi_1 \mathsf{S}^{\mathsf{c}} \varphi_2$ iff either (i) i is a call and $(w^H_{bad}, i) \models \varphi_1 \mathsf{S}^{\mathsf{g}} \varphi_2$, or (ii) i is a return, and either $(w^H_{bad}, i) \models \varphi_2$, or $(w^H_{bad}, i_c) \models \varphi_1 \mathsf{S}^{\mathsf{g}} \varphi_2$, where i_c is the caller of i. Hence, the case for modality S^{c} easily reduces to the case of modality S^{g}.
- $\varphi = \bigcirc^{\mathsf{g}} \varphi_1$. Assume that $(w^H_{good}, H(i)) \models \varphi$. Hence, $H(i) < 4H+1$ and $(w^H_{good}, H(i)+1) \models \varphi_1$. By construction, *either* $H(i)+1 = H(i+1)$, *or* $H(i)+1 \in \{H, 3H+1\}$ and $H(i+1) = (H(i)+1)+1$. In the first case, by the induction hypothesis, we obtain that $(w^H_{good}, i+1) \models \varphi_1$. In the second case, by applying Claim 1, we deduce that $(w^H_{good}, H(i)+2) \models \varphi_1$, hence, by the induction hypothesis, $(w^H_{good}, i+1) \models \varphi_1$ holds as well. The converse implication $(w^H_{bad}, i) \models \varphi \Rightarrow (w^H_{good}, H(i)) \models \varphi$ is similar.
- $\varphi = \ominus^{\mathsf{g}} \varphi_1$: this case is similar to the previous one.
- $\varphi = \bigcirc^{\mathsf{a}} \varphi_1$ or $\varphi = \ominus^{\mathsf{a}} \varphi_1$: similar to the case of the abstract until and since modalities.
- $\varphi = \ominus^{\mathsf{c}} \varphi_1$: let $i \in [0, 4H-1]$. By construction, $(w^H_{bad}, i) \models \ominus^{\mathsf{c}} \varphi_1$ iff either (i) i is a call and $(w^H_{bad}, i) \models \ominus^{\mathsf{g}} \varphi_1$, or (ii) i is a return and $(w^H_{bad}, i_c) \models \ominus^{\mathsf{g}} \varphi_1$, where i_c is the matched-call of i. Hence, the case for modality $\ominus^{\mathsf{c}}$ reduces to the case of modality $\ominus^{\mathsf{g}}$.
- $\varphi = \rhd^{\mathsf{g}}_I \varphi_1$: for all positions ℓ of w^H_{good} (resp., w^H_{bad}), let τ^{good}_ℓ (resp., τ^{bad}_ℓ) be the timestamp of w^H_{good} (resp., w^H_{bad}) at position ℓ. Let $i \in [0, 4H-1]$. We prove the implication $(w^H_{good}, H(i)) \models \varphi \Rightarrow (w^H_{bad}, i) \models \varphi$ (the converse implication being similar). Let $(w^H_{good}, H(i)) \models \varphi$. Hence, there is $\ell \in [H(i)+1, 4H+1]$ such that $(w^H_{good}, \ell) \models \varphi_1$, $\tau^{good}_\ell - \tau^{good}_{H(i)} \in I$ and $(w^H_{good}, \ell') \not\models \varphi_1$ for all $\ell' \in [H(i)+1, \ell-1]$. We have two cases:

  - $\ell > H(i)+1$: by hypothesis, $(w^H_{good}, \ell-1) \not\models \varphi_1$ and $(w^H_{good}, \ell) \models \varphi_1$. We first show that $\ell = H(j)$ for some $j \in [0, 4H-1]$. We assume the contrary and derive a contradiction. Hence, $\ell \in \{H, 3H+1\}$. Since $d(\varphi) < H$, by Claim 1, we deduce that $(w^H_{good}, \ell-1) \models \varphi_1$, a contradiction. Hence, $\ell = H(j)$ for some $j \in [0, 4H-1]$. By construction, $\tau^{good}_{H(j)} - \tau^{good}_{H(i)} = \tau^{bad}_j - \tau^{bad}_i$. Thus, by the induction hypothesis, we obtain that $(w^H_{bad}, i) \models \varphi$, and the result follows.

  - $\ell = H(i) + 1$. Hence, by construction, $0 < \tau_{\ell}^{good} - \tau_{H(i)}^{good} < 1$. If $H(i) + 1 = H(i+1)$, then being $0 < \tau_{i+1}^{bad} - \tau_{i}^{bad} < 1$, the result directly follows from the induction hypothesis. Otherwise, $\ell \in \{H, 3H+1\}$ and $H(i+1) = \ell + 1$. By applying Claim 1 and the induction hypothesis, we obtain that $(w_{bad}^{H}, i+1) \models \varphi_1$. Moreover, by construction, $0 < \tau_{i+1}^{bad} - \tau_{i}^{bad} < 1$. Hence, the result follows.

- $\varphi = \lhd_I^{\mathsf{g}} \varphi_1$: this case is similar to the previous one. $\square$

References

[1] P.A. Abdulla, M.F. Atig, J. Stenman, Dense-timed pushdown automata, in: Proc. 27th LICS, IEEE Computer Society, 2012, pp. 35–44.
[2] R. Alur, D.L. Dill, A theory of timed automata, Theor. Comput. Sci. 126 (2) (1994) 183–235.
[3] R. Alur, K. Etessami, P. Madhusudan, A temporal logic of nested calls and returns, in: Proc. 10th TACAS, in: LNCS, vol. 2988, Springer, 2004, pp. 467–481.
[4] R. Alur, T. Feder, T.A. Henzinger, The benefits of relaxing punctuality, J. ACM 43 (1) (1996) 116–146.
[5] R. Alur, L. Fix, T.A. Henzinger, Event-clock automata: a determinizable class of timed automata, Theor. Comput. Sci. 211 (1–2) (1999) 253–273.
[6] R. Alur, T.A. Henzinger, Real-time logics: complexity and expressiveness, Inf. Comput. 104 (1) (1993) 35–77.
[7] R. Alur, P. Madhusudan, Visibly pushdown languages, in: Proc. 36th STOC, ACM, 2004, pp. 202–211.
[8] R. Alur, P. Madhusudan, Adding nesting structure to words, J. ACM 56 (3) (2009) 16:1–16:43.
[9] É. André, D. Lime, O.H. Roux, On the expressiveness of parametric timed automata, in: Formal Modeling and Analysis of Timed Systems - 14th International Conference, FORMATS 2016, Quebec, QC, Canada, August 24-26, 2016, Proceedings, 2016, pp. 19–34.
[10] C. Baier, J.-P. Katoen, Principles of Model Checking, The MIT Press, 2008.
[11] M. Benerecetti, S. Minopoli, A. Peron, Analysis of timed recursive state machines, in: TIME 2010 - 17th International Symposium on Temporal Representation and Reasoning, Paris, France, 6-8 September 2010, 2010, pp. 61–68.
[12] M. Benerecetti, A. Peron, Timed recursive state machines: expressiveness and complexity, Theor. Comput. Sci. 625 (2016) 85–124.
[13] N. Benes, P. Bezdek, K.G. Larsen, J. Srba, Language emptiness of continuous-time parametric timed automata, in: Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II, 2015, pp. 69–81.
[14] B. Bérard, S. Haddad, A. Jovanovic, D. Lime, Interrupt timed automata with auxiliary clocks and parameters, Fundam. Inform. 143 (3–4) (2016) 235–259.
[15] D. Bhave, V. Dave, S.N. Krishna, R. Phawade, A. Trivedi, A logical characterization for dense-time visibly pushdown automata, in: Proc. 10th LATA, in: LNCS, vol. 9618, Springer, 2016, pp. 89–101.
[16] A. Bouajjani, R. Echahed, R. Robbana, On the automatic verification of systems with continuous variables and unbounded discrete data structures, in: Hybrid Systems II, in: LNCS, vol. 999, Springer, 1994, pp. 64–85.
[17] A. Bouajjani, J. Esparza, O. Maler, Reachability analysis of pushdown automata: application to model-checking, in: CONCUR'97, 1997, pp. 135–150.
[18] L. Bozzelli, A. Murano, A. Peron, Pushdown module checking, Form. Methods Syst. Des. 36 (1) (2010) 65–95.
[19] L. Bozzelli, A. Murano, A. Peron, Timed context-free temporal logics, in: Proceedings Ninth International Symposium on Games, Automata, Logics, and Formal Verification, GandALF 2018, Saarbrücken, Germany, 26-28th September 2018, 2018, pp. 235–249.
[20] L. Bozzelli, A. Peron, A. Murano, Event-clock nested automata, in: Proc. 12th LATA, in: LNCS, vol. 10792, Springer, 2018, pp. 80–92.
[21] D. Carotenuto, A. Murano, A. Peron, 2-Visibly pushdown automata, in: Developments in Language Theory, 11th International Conference, DLT 2007, Turku, Finland, July 3-6, 2007, Proceedings, 2007, pp. 132–144.
[22] D. Carotenuto, A. Murano, A. Peron, Ordered multi-stack visibly pushdown automata, Theor. Comput. Sci. 656 (2016) 1–26.
[23] K. Chatterjee, D. Ma, R. Majumdar, T. Zhao, T. Henzinger, J. Palsberg, Stack size analysis for interrupt-driven programs, in: Proc. 10th SAS, in: LNCS, vol. 2694, Springer, 2003, pp. 109–126.
[24] E. Clarke, E. Emerson, Design and synthesis of synchronization skeletons using branching time temporal logic, in: LP'81, in: LNCS, vol. 131, 1981, pp. 52–71.
[25] L. Clemente, S. Lasota, Timed pushdown automata revisited, in: Proc. 30th LICS, IEEE Computer Society, 2015, pp. 738–749.
[26] M. Emmi, R. Majumdar, Decision problems for the verification of real-time software, in: Proc. 9th HSCC, in: LNCS, vol. 3927, 2006, pp. 200–211.
[27] R. Koymans, Specifying real-time properties with metric temporal logic, Real-Time Syst. 2 (4) (1990) 255–299.
[28] O. Kupferman, N. Piterman, M.Y. Vardi, Pushdown specifications, in: Proc. 9th LPAR, in: LNCS, vol. 2514, Springer, 2002, pp. 262–277.
[29] M. Minsky, Computation: Finite and Infinite Machines, Prentice-Hall, Englewood Cliffs, 1967.
[30] J. Ouaknine, J. Worrell, On metric temporal logic and faulty Turing machines, in: Proc. 9th FOSSACS, in: LNCS, vol. 3921, Springer, 2006, pp. 217–230.
[31] J. Ouaknine, J. Worrell, On the decidability and complexity of metric temporal logic over finite words, Log. Methods Comput. Sci. 3 (1) (2007).
[32] J. Queille, J. Sifakis, Specification and verification of concurrent programs in Cesar, in: SP'81, in: LNCS, vol. 137, Springer, 1981, pp. 337–351.
[33] J. Raskin, P. Schobbens, The logic of event clocks - decidability, complexity and expressiveness, J. Autom. Lang. Comb. 4 (3) (1999) 247–286.
[34] N.V. Tang, M. Ogawa, Event-clock visibly pushdown automata, in: Proc. 35th SOFSEM, in: LNCS, vol. 5404, Springer, 2009, pp. 558–569.
[35] A. Trivedi, D. Wojtczak, Recursive timed automata, in: Proc. 8th ATVA, in: LNCS, vol. 6252, Springer, 2010, pp. 306–324.
[36] I. Walukiewicz, Pushdown processes: games and model checking, in: CAV'96, 1996, pp. 62–74.