Timed Context-Free Temporal Logics*

Laura Bozzelli Aniello Murano Adriano Peron
University of Napoli "Federico II", Napoli, Italy

The paper is focused on temporal logics for the description of the behaviour of real-time pushdown reactive systems. The paper is motivated to bridge tractable logics specialized for expressing separately dense-time real-time properties and context-free properties by ensuring decidability and tractability in the combined setting. To this end we introduce two real-time linear temporal logics for specifying quantitative timing context-free requirements in a pointwise semantics setting: *Event-Clock Nested Temporal Logic* (EC_NTL) and *Nested Metric Temporal Logic* (NMTL). The logic EC_NTL is an extension of both the logic CaRet (a context-free extension of standard LTL) and *Event-Clock Temporal Logic* (a tractable real-time logical framework related to the class of Event-Clock automata). We prove that satisfiability of EC_NTL and visibly model-checking of Visibly Pushdown Timed Automata (VPTA) against EC_NTL are decidable and EXPTIME-complete. The other proposed logic NMTL is a context-free extension of standard Metric Temporal Logic (MTL). It is well known that satisfiability of future MTL is undecidable when interpreted over infinite timed words but decidable over finite timed words. On the other hand, we show that by augmenting future MTL with future context-free temporal operators, the satisfiability problem turns out to be undecidable also for finite timed words. On the positive side, we devise a meaningful and decidable fragment of the logic NMTL which is expressively equivalent to EC_NTL and for which satisfiability and visibly model-checking of VPTA are EXPTIME-complete.

1 Introduction

Model checking is a well-established formal-method technique to automatically check for global correctness of reactive systems [8]. In this setting, temporal logics provide a fundamental framework for the description of the dynamic behavior of reactive systems.

In the last two decades, model checking of pushdown automata (PDA) has received a lot of attention [26, 16, 7, 13]. 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, algorithmic solutions have been proposed for interesting subclasses of context-free requirements [3, 7, 16]. A relevant example is that of the linear temporal logic CaRet [3], a context-free extension of standard LTL. CaRet formulas are interpreted on words over a *pushdown alphabet* which is partitioned into three disjoint sets of calls, returns, and internal symbols. A call denotes invocation of a procedure (i.e. a push stack-operation) and the *matching* return (if any) along a given word denotes the exit from this procedure (corresponding to a pop stack-operation). CaRet allows to specify LTL 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. An automata theoretic generalization of CaRet is the class of (nondeterministic) *Visibly Pushdown Automata* (VPA) [7], a subclass of PDA where the input symbols

*The work by Adriano Peron and Aniello Murano has been partially supported by the GNCS project *Formal methods for verification and synthesis of discrete and hybrid systems* and by Dept. project MODAL *MOdel-Driven Analysis of Critical Industrial Systems.*

A. Orlandini, M. Zimmermann (Eds.): 9th Symposium on Games, Automata, Logics and Formal Verification (GandALF'18)
EPTCS 277, 2018, pp. 235–249, doi:10.4204/EPTCS.277.17

over a pushdown alphabet control the admissible operations on the stack. VPA push onto the stack only when a call is read, pops 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 the less expressive class of regular languages [7]. In fact, VPL are closed under Boolean operations, and language inclusion, which is undecidable for context-free languages, is EXPTIME-complete for VPL.

Real-time pushdown model-checking. Recently, many works [1, 9, 10, 12, 17, 18, 25] 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*). All the clocks progress at the same speed and can be reset by transitions (thus, each clock keeps track of the elapsed time since the last reset). The emptiness problem for TA is decidable and PSPACE-complete [2]. However, since in TA, clocks can be reset nondeterministically and independently of each other, the resulting class of timed languages is not closed under complement and, moreover, 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. In fact, checking language inclusion for VPTA is undecidable even in the restricted case of specifications using at most one clock [18]. More robust approaches [24, 11, 14], although less expressive, are based on formalisms combining VPA and *Event-clock automata* (ECA) [5] such as the recently introduced class of *Event-Clock Nested Automata* (ECNA) [14]. ECA [5] are a well-known determinizable subclass of TA where the explicit reset of clocks is disallowed. In ECA, clocks have a predefined association with the input alphabet symbols and their values refer to the time distances from previous and next occurrences of input symbols. ECNA [14] combine ECA and VPA by providing an explicit mechanism to relate the use of a stack with that of event clocks. In particular, ECNA retain the closure and decidability properties of ECA and VPA being closed under Boolean operations and having a decidable (specifically, EXPTIME-complete) language-inclusion problem, and are strictly more expressive than other formalisms combining ECA and VPA [24, 11] such as the class of *Event-Clock Visibly Pushdown Automata* (ECVPA) [24]. In [11] a logical characterization of the class of ECVPA is provided by means of a non-elementarily decidable extension of standard MSO over words.

Our contribution. 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 logic EC_NTL is an extension of *Event-Clock Temporal Logic* (EC_TL) [23], the latter being a known decidable and tractable real-time logical framework related to the class of Event-clock automata. EC_TL extends LTL + past with timed temporal modalities which specify time constraints on the distances from the previous or next timestamp where a given subformula holds. The novel 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 allow to refer to abstract and caller paths. We address expressiveness and complexity issues for the logic EC_NTL. In particular, we establish 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, namely 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 interpreted over infinite timed words [21],

Table 1: Decidability results.

Logic	Satisfiability	Visibly model checking
EC_NTL	EXPTIME-complete	EXPTIME-complete
$\mathsf{NMITL}_{(0,\infty)}$	EXPTIME-complete	EXPTIME-complete
future MTL fin.	Decidable	
future MTL infin.	Undecidable	
future NMTL fin.	Undecidable	

and decidable [22] over finite timed words. We show that over finite timed words, the adding of the future abstract timed modalities to future MTL makes the satisfiability problem undecidable. 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 EXPTIME-complete. The overall picture of decidability results is reported in table 1 (new results in red).

Due to space limitations, some proofs are omitted: we refer for complete proofs to the complete version of the paper in [15].

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 write $|w| = \infty$ if w is infinite). For all $i, j \in \mathbb{N}$, with $i \leq j < |w|$, w_i is i-th letter of w, while $w[i, j]$ is the finite subword $w_i \cdots w_j$.

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

$$\sigma_w := \varepsilon \mid a \cdot \sigma_w \mid c \cdot \sigma_w \cdot r \cdot \sigma_w$$

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

Fix a word σ over Σ. For a call position i of σ, 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 of σ, 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 $i+1 < |\sigma|$ and $i+1$ 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:

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

We also consider the *global successor* $\mathsf{succ}(\mathsf{g},\sigma,i)$ of i along σ given by $i+1$ if $i+1 < |\sigma|$, and undefined otherwise. 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 $i_j = \mathsf{succ}(\mathsf{a},\sigma,i_{j-1})$ 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 $j_{h+1} = \mathsf{succ}(\mathsf{c},\sigma,j_h)$ for all $0 \leq h < n$. Note that all the positions of a *MAP* have the same caller (if any). Intuitively, 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.

For instance, consider the finite untimed word σ of length 10 depicted in Figure 1 where $\Sigma_{call} = \{c\}$, $\Sigma_{ret} = \{r\}$, and $\Sigma_{int} = \{\imath\}$. Note that 0 is the unique unmatched call position of σ: hence, the *MAP* visiting 0 consists of just position 0 and has no caller. The *MAP* visiting position 1 is the sequence $1,6,7,9,10$ and the associated caller is position 0. The *MAP* visiting position 2 is the sequence $2,3,5$ and the associated caller is position 1, and the *MAP* visiting position 4 consists of just position 4 whose caller path is $4,3,1,0$.

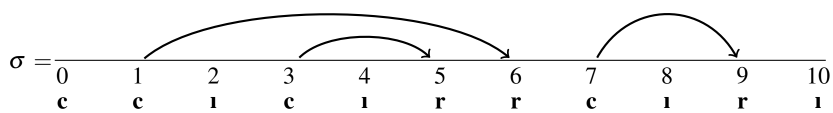

Figure 1: An untimed word over a pushdown alphabet

3 Event-clock nested automata

In this section, we recall the class of *Event-Clock Nested Automata* (ECNA) [14], a formalism that combines Event Clock Automata (ECA) [5] and Visibly Pushdown Automata (VPA) [7] by allowing a combined used of event clocks and visible operations on the stack.

Here, we adopt a propositional-based approach, where the pushdown alphabet is implicitly given. This is because in formal verification, one usually considers a finite set of atomic propositions which represent predicates over the states of the given system. Moreover, for verifying recursive programs, one fixes three additional propositions, here denoted by *call*, *ret*, and *int*: *call* denotes the invocation of a procedure, *ret* denotes the return from a procedure, and *int* denotes internal actions of the current procedure. Thus, we fix a finite set $\mathscr{P}$ of atomic propositions containing the special propositions *call*, *ret*, and *int*. The set $\mathscr{P}$ induces a pushdown alphabet $\Sigma_{\mathscr{P}} = \Sigma_{call} \cup \Sigma_{ret} \cup \Sigma_{int}$, where $\Sigma_{call} = \{P \subseteq \mathscr{P} \mid P \cap \{call, ret, int\} = \{call\}\}$, $\Sigma_{ret} = \{P \subseteq \mathscr{P} \mid P \cap \{call, ret, int\} = \{ret\}\}$, and $\Sigma_{int} = \{P \subseteq \mathscr{P} \mid P \cap \{call, ret, int\} = \{int\}\}$.

The set $C_{\mathscr{P}}$ of event clocks associated with $\mathscr{P}$ is given by $C_{\mathscr{P}} := \bigcup_{p \in \mathscr{P}} \{x_p^{\mathsf{g}}, y_p^{\mathsf{g}}, x_p^{\mathsf{a}}, y_p^{\mathsf{a}}, x_p^{\mathsf{c}}\}$. Thus, we associate with each proposition $p \in \mathscr{P}$, five event clocks: the *global recorder clock* x_p^{g} (resp., the *global predictor clock* y_p^{g}) recording the time elapsed since the last occurrence of p if any (resp., the time required to the next occurrence of p if any); the *abstract recorder clock* x_p^{a} (resp., the *abstract predictor clock* y_p^{a}) recording the time elapsed since the last occurrence of p if any (resp., the time required to the next occurrence of p) along the *MAP* visiting the current position; and the *caller (recorder) clock* x_p^{c} recording the time elapsed since the last occurrence of p if any along the caller path from the current position. Let $w = (\sigma,\tau)$ be a timed word over $\Sigma_{\mathscr{P}}$ and $0 \leq i < |w|$. We denote by $Pos(\mathsf{a},\sigma,i)$ the set of positions visited by the *MAP* of σ associated with position i, and by $Pos(\mathsf{c},\sigma,i)$ the set of positions visited by the caller path of σ from position i. For having a uniform notation, let $Pos(\mathsf{g},\sigma,i)$ be the full set of w-positions. The values of the clocks at a position i of the word w can be deterministically determined as follows.

Definition 1 (Determinisitic clock valuations)**.** *A* clock valuation *over $C_{\mathscr{P}}$ is a mapping $val : C_{\mathscr{P}} \mapsto \mathbb{R}_+ \cup \{\vdash\}$, assigning to each event clock a value in $\mathbb{R}_+ \cup \{\vdash\}$ ($\vdash$ is the* undefined *value). For a timed word $w = (\sigma,\tau)$ over Σ and $0 \leq i < |w|$, the* clock valuation val_i^w over $C_{\mathscr{P}}$*, specifying the values of the event clocks at position i along w, is defined as follows for each $p \in \mathscr{P}$, where $dir \in \{\mathsf{g},\mathsf{a}\}$ and $dir' \in \{\mathsf{g},\mathsf{a},\mathsf{c}\}$:*

$$val_i^w(x_p^{dir'}) = \begin{cases} \tau_i - \tau_j & \text{if there exists the unique } j < i : p \in \sigma_j,\ j \in Pos(dir',\sigma,i), \text{ and} \\ & \quad \forall k : (j < k < i \text{ and } k \in Pos(dir',\sigma,i)) \Rightarrow p \notin \sigma_k \\ \vdash & \text{otherwise} \end{cases}$$

$$val_i^w(y_p^{dir}) = \begin{cases} \tau_j - \tau_i & \text{if there exists the unique } j > i : p \in \sigma_j,\ j \in Pos(dir,\sigma,i), \text{ and} \\ & \quad \forall k : (i < k < j \text{ and } k \in Pos(dir,\sigma,i)) \Rightarrow p \notin \sigma_k \\ \vdash & \text{otherwise} \end{cases}$$

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).

A *clock constraint* over $C_{\mathscr{P}}$ is a conjunction of atomic formulas of the form $z \in I$, where $z \in C_{\mathscr{P}}$, and I is either an interval in $\mathbb{R}_+$ with bounds in $\mathbb{N} \cup \{\infty\}$, or the singleton $\{\vdash\}$. 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 $\Phi(C_{\mathscr{P}})$ the set of clock constraints over $C_{\mathscr{P}}$.

Definition 2. *An* ECNA *over $\Sigma_{\mathscr{P}} = \Sigma_{call} \cup \Sigma_{int} \cup \Sigma_{ret}$ is a tuple $\mathscr{A} = (\Sigma_{\mathscr{P}}, Q, Q_0, C_{\mathscr{P}}, \Gamma \cup \{\bot\}, \Delta, F)$, where Q is a finite set of (control) states, $Q_0 \subseteq Q$ is a set of initial states, $\Gamma \cup \{\bot\}$ is a finite stack alphabet, $\bot \notin \Gamma$ is the special* stack bottom symbol*, $F \subseteq Q$ is a set of accepting states, and $\Delta = \Delta_c \cup \Delta_r \cup \Delta_i$ is a transition relation, where:*

- $\Delta_c \subseteq Q \times \Sigma_{call} \times \Phi(C_{\mathscr{P}}) \times Q \times \Gamma$ *is the set of* push transitions*,*
- $\Delta_r \subseteq Q \times \Sigma_{ret} \times \Phi(C_{\mathscr{P}}) \times (\Gamma \cup \{\bot\}) \times Q$ *is the set of* pop transitions*,*
- $\Delta_i \subseteq Q \times \Sigma_{int} \times \Phi(C_{\mathscr{P}}) \times Q$ *is the set of* internal transitions*.*

We now describe how an ECNA $\mathscr{A}$ behaves over a timed word w. Assume that on reading the i-th position of w, the current state of $\mathscr{A}$ is q, and val_i^w is the event-clock valuation associated with w and position i. If $\mathscr{A}$ reads a call $c \in \Sigma_{call}$, it chooses a push transition of the form $(q,c,\theta,q',\gamma) \in \Delta_c$ and pushes the symbol $\gamma \neq \bot$ onto the stack. If $\mathscr{A}$ reads a return $r \in \Sigma_{ret}$, it chooses a pop transition of the form $(q,r,\theta,\gamma,q') \in \Delta_r$ such that γ is the symbol on the top of the stack, and pops γ from the stack (if $\gamma = \bot$, then γ is read but not removed). Finally, on reading an internal action $a \in \Sigma_{int}$, $\mathscr{A}$ chooses an internal transition of the form $(q,a,\theta,q') \in \Delta_i$, and, in this case, there is no operation on the stack. Moreover, in all the cases, the constraint θ of the chosen transition must be fulfilled by the valuation val_i^w and the control changes from q to q'.

Formally, a configuration of $\mathscr{A}$ is a pair (q,β), where $q \in Q$ and $\beta \in \Gamma^* \cdot \{\bot\}$ is a stack content. A run π of $\mathscr{A}$ over a timed word $w = (\sigma,\tau)$ is a sequence of configurations $\pi = (q_0,\beta_0),(q_1,\beta_1),\ldots$ of length $|w|+1$ ($\infty + 1$ stands for ∞) such that $q_0 \in Q_0$, $\beta_0 = \bot$ (initialization), and the following holds for all $0 \leq i < |w|$:

Push If $\sigma_i \in \Sigma_{call}$, then for some $(q_i,\sigma_i,\theta,q_{i+1},\gamma) \in \Delta_c$, $\beta_{i+1} = \gamma \cdot \beta_i$ and $val_i^w \models \theta$.

Pop If $\sigma_i \in \Sigma_{ret}$, then for some $(q_i,\sigma_i,\theta,\gamma,q_{i+1}) \in \Delta_r$, $val_i^w \models \theta$, and *either* $\gamma \neq \bot$ and $\beta_i = \gamma \cdot \beta_{i+1}$, *or* $\gamma = \beta_i = \beta_{i+1} = \bot$.

Internal If $\sigma_i \in \Sigma_{int}$, then for some $(q_i, \sigma_i, \theta, q_{i+1}) \in \Delta_i$, $\beta_{i+1} = \beta_i$ and $val_i^w \models \theta$.

The run π is *accepting* if *either* π is finite and $q_{|w|} \in F$, *or* π is infinite and there are infinitely many positions $i \geq 0$ such that $q_i \in F$. The *timed language* $\mathcal{L}_T(\mathcal{A})$ (resp., ω-*timed language* $\mathcal{L}_T^{\omega}(\mathcal{A})$) of $\mathcal{A}$ is the set of finite (resp., infinite) timed words w over $\Sigma_{\mathcal{P}}$ such that there is an accepting run of $\mathcal{A}$ on w. When considered as an acceptor of infinite timed words, an ECNA is called Büchi ECNA. In this case, for technical convenience, we also consider ECNA equipped with a *generalized Büchi acceptance condition* $\mathcal{F}$ consisting of a family of sets of accepting states. In such a setting, an infinite run π is accepting if for each Büchi component $F \in \mathcal{F}$, the run π visits infinitely often states in F.

In the following, we also consider the class of *Visibly Pushdown Timed Automata* (VPTA) [12, 18], a combination of VPA and standard Timed Automata [2]. The clocks in a VPTA can be reset when a transition is taken; hence, their values at a position of an input word depend in general on the behaviour of the automaton and not only, as for event clocks, on the word. The syntax and semantics of VPTA is shortly recalled in Appendix A of [15].

4 The Event-Clock Nested Temporal Logic

A known decidable timed temporal logical framework related to the class of Event-Clock automata (ECA) is the so called *Event-Clock Temporal Logic* (EC_TL) [23], an extension of standard LTL with past obtained by means of two indexed modal operators $\triangleleft$ and $\triangleright$ which express real-time constraints. On the other hand, for the class of VPA, a related logical framework is the temporal logic CaRet [3], a well-known context-free extension of LTL with past by means of non-regular versions of the LTL temporal operators. In this section, we introduce an extension of both EC_TL and CaRet, called *Event-Clock Nested Temporal Logic* (EC_NTL) which allows to specify non-regular context-free real-time properties.

For the given set $\mathcal{P}$ of atomic propositions containing the special propositions *call*, *ret*, and *int*, the syntax of EC_NTL formulas φ is as follows:

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

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 allow to 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 $\triangleright_I^{\mathsf{g}}$, $\triangleleft_I^{\mathsf{g}}$, $\triangleright_I^{\mathsf{a}}$, $\triangleleft_I^{\mathsf{a}}$, and $\triangleleft_I^{\mathsf{c}}$. The formula $\triangleright_I^{\mathsf{g}}\varphi$ requires that the delay t before the next position where φ holds satisfies $t \in I$; symmetrically, $\triangleleft_I^{\mathsf{g}}\varphi$ constraints the previous position where φ holds. The abstract versions $\triangleright_I^{\mathsf{a}}\varphi$ and $\triangleleft_I^{\mathsf{a}}\varphi$ 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 $\triangleleft_I^{\mathsf{c}}\varphi$ of $\triangleleft_I^{\mathsf{g}}\varphi$, 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 [23] is obtained from EC_NTL by disallowing the abstract and caller modalities. As pointed out in [23], the real-time operators $\triangleleft$ and $\triangleright$ generalize the semantics of event clock variables since they allows 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 $\triangleright_I^{dir}\varphi$ and $\triangleleft_I^{dir'}\varphi$ in the syntax with the clauses $\triangleright_I^{dir}p$ and $\triangleleft_I^{dir'}p$, 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 LTL eventually operator), and $\Box^{\mathsf{g}}\psi$ 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 $Const_\varphi$.

Given an EC_NTL formula φ, a timed word $w = (\sigma, \tau)$ over $\Sigma_{\mathscr{P}}$ 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):

$$
\begin{array}{lll}
(w,i) \models \bigcirc^{dir}\varphi & \Leftrightarrow & \text{there is } j > i \text{ such that } j = \mathsf{succ}(dir,\sigma,i) \text{ and } (w,j) \models \varphi \\
(w,i) \models \ominus^{dir'}\varphi & \Leftrightarrow & \text{there is } j < i \text{ such that } (w,j) \models \varphi \text{ and } \textit{either } (dir' \neq \mathsf{c} \text{ and} \\
& & i = \mathsf{succ}(dir',\sigma,j)), \text{ or } (dir' = \mathsf{c} \text{ and } j = \mathsf{succ}(\mathsf{c},\sigma,i)) \\
(w,i) \models \varphi_1 \mathsf{U}^{dir} \varphi_2 & \Leftrightarrow & \text{there is } j \geq i \text{ such that } j \in Pos(dir,\sigma,i), (w,j) \models \varphi_2 \text{ and} \\
& & (w,k) \models \varphi_1 \text{ for all } k \in [i, j-1] \cap Pos(dir,\sigma,i) \\
(w,i) \models \varphi_1 \mathsf{S}^{dir'} \varphi_2 & \Leftrightarrow & \text{there is } j \leq i \text{ such that } j \in Pos(dir',\sigma,i), (w,j) \models \varphi_2 \text{ and} \\
& & (w,k) \models \varphi_1 \text{ for all } k \in [j+1, i] \cap Pos(dir',\sigma,i)
\end{array}
$$

$$
\begin{array}{lll}
(w,i) \models \rhd_I^{dir}\varphi & \Leftrightarrow & \text{there is } j > i \text{ s.t. } j \in Pos(dir,\sigma,i), (w,j) \models \varphi, \tau_j - \tau_i \in I, \\
& & \text{and } (w,k) \not\models \varphi \text{ for all } k \in [i+1, j-1] \cap Pos(dir,\sigma,i) \\
(w,i) \models \lhd_I^{dir'}\varphi & \Leftrightarrow & \text{there is } j < i \text{ s.t. } j \in Pos(dir',\sigma,i), (w,j) \models \varphi, \tau_i - \tau_j \in I, \\
& & \text{and } (w,k) \not\models \varphi \text{ for all } k \in [j+1, i-1] \cap Pos(dir',\sigma,i)
\end{array}
$$

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 $\mathscr{L}_T(\varphi)$ (resp. ω-timed language $\mathscr{L}_T^\omega(\varphi)$) of φ is the set of finite (resp., infinite) timed words over $\Sigma_{\mathscr{P}}$ satisfying φ. We consider the following decision problems:

- *Satisfiability:* has a given EC_NTL formula a finite (resp., infinite) model?
- *Visibly model-checking:* given a VPTA $\mathscr{A}$ over $\Sigma_{\mathscr{P}}$ and an EC_NTL formula φ over $\mathscr{P}$, does $\mathscr{L}_T(\mathscr{A}) \subseteq \mathscr{L}_T(\varphi)$ (resp., $\mathscr{L}_T^\omega(\mathscr{A}) \subseteq \mathscr{L}_T^\omega(\varphi)$) hold?

The logic EC_NTL allows to express in a natural way real-time LTL-like properties over the non-regular patterns capturing the local computations of procedures or the stack contents at given positions. Here, 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: $\Box^{\mathsf{g}}\big((call \wedge p \wedge p_A) \rightarrow (\bigcirc^{\mathsf{a}} q \wedge \rhd^{\mathsf{a}}_{[0,k]} ret)\big)$
- *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: $\Box^{\mathsf{g}}\big((p \wedge c_A) \rightarrow \rhd^{\mathsf{a}}_{[0,k]} q\big)$
- *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): $\Box^{\mathsf{g}}\big((call \wedge p_A) \rightarrow \lhd^{\mathsf{c}}_{[0,k]}\, p_B\big)$

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 [23]. This result trivially generalizes to ECNA and EC_NTL (note that over timed words consisting only of internal actions, ECNA correspond to ECA, and the logic EC_NTL corresponds to EC_TL). In [14], it is shown that ECNA are strictly less expressive than VPTA. In Section 4.1, 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 [18]. Hence, we obtain the following result.

Theorem 1. *Over finite (resp., infinite) timed words, EC_NTL and ECNA are expressively incomparable, and EC_NTL is strictly less expressive than VPTA.*

We additionally investigate the expressiveness of the novel timed temporal modalities $\lhd^{\mathsf{a}}_I$, $\rhd^{\mathsf{a}}_I$, and $\lhd^{\mathsf{c}}_I$. It turns out that these modalities add expressive power.

Theorem 2. *Let $\mathscr{F}$ 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, $\mathscr{F}$ 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 $\mathscr{P} = \{call, ret\}$ and $\mathscr{L}_T$ 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. $\mathscr{L}_T$ can be easily expressed in EC_NTL. On the other hand, one can show that $\mathscr{L}_T$ is not definable in $\mathscr{F}$ (the detailed proof can be found in Appendix B of [15]) . □

4.1 Decision procedures for the logic EC_NTL

In this section, we provide an automata-theoretic approach for solving satisfiability and visibly model-checking for the logic EC_NTL which generalizes both the automatic-theoretic approach of CaRet [3] and the one for EC_TL [23]. We focus on infinite timed words (the approach for finite timed words is similar). Given an EC_NTL formula φ over $\mathscr{P}$, we construct in exponential time a generalized Büchi ECNA $\mathscr{A}_\varphi$ over an extension of the pushdown alphabet $\Sigma_{\mathscr{P}}$ accepting suitable encodings of the infinite models of φ.

Fix an EC_NTL formula φ over $\mathscr{P}$. For each infinite timed word $w = (\sigma, \tau)$ over $\Sigma_{\mathscr{P}}$ we associate to w an infinite timed word $\pi = (\sigma_e, \tau)$ over an extension of $\Sigma_{\mathscr{P}}$, 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 various 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 a position i along an infinite timed word w over $\Sigma_{\mathscr{P}}$ 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 position 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:

- $\top \in \mathsf{Cl}(\varphi)$, each proposition $p \in \mathscr{P} \cup \{p_\infty\}$, and formulas $\bigcirc^{\mathsf{a}} \top$ and $\ominus^{\mathsf{a}} \top$;
- 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_i^π specifying the values of the event clocks x_ψ, y_ψ, x_ψ^{a}, y_ψ^{a}, and x_ψ^{c} 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:

- 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 now 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 $\Sigma_{\mathscr{P}}$ (the projection of π over $\Sigma_{\mathscr{P}} \times \mathbb{R}_+$). 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* $\Sigma_{Cl(\varphi)}$*, 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* $\ominus^{dir}\psi \in Cl(\varphi)$*,* $\neg\ominus^{dir}\psi \in A_0$.
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* $Caller(A_{i+1}) = \{\ominus^{\mathsf{c}}\psi \in Cl(\varphi) \mid \psi \in A_i\}$ *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* $\lhd_I^{dir}\psi \in Cl(\varphi)$*,* $\lhd_I^{dir}\psi \in A_i$ *iff* $val_i^\pi(x_\psi^{dir}) \in I$*;*
   - *for all* $dir \in \{\mathsf{g}, \mathsf{a}\}$ *and* $\rhd_I^{dir}\psi \in Cl(\varphi)$*,* $\rhd_I^{dir}\psi \in A_i$ *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 (i) for infinitely many $i \geq 0$, $p_\infty \in A_i$; (ii) 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 (iii) 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, there is at a most one *infinite MAP* ν of σ, 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. Thus, we obtain the following characterization of the infinite models of φ, where $Proj_\varphi$ is 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 $\Sigma_{\mathcal{P}}$ given by $Proj(\pi) = (A_0 \cap \mathcal{P}, t_0)(A_1 \cap \mathcal{P}, t_1)\ldots$.

Proposition 1. Let $\pi = (A_0,t_0)(A_1,t_1)\ldots$ be a fair Hintikka sequence of φ. Then, for all $i \geq 0$ and $\psi \in \mathsf{Cl}(\varphi) \setminus \{p_\infty, \neg p_\infty\}$, $\psi \in A_i$ *iff* $(Proj_\varphi(\pi), i) \models \psi$. Moreover, the mapping $Proj_\varphi$ is a bijection between the set of fair Hintikka sequences of φ and the set of infinite timed words over $\Sigma_{\mathcal{P}}$. In particular, an infinite timed word over $\Sigma_{\mathcal{P}}$ is a model of φ iff the associated fair Hintikka sequence is initialized.

The detailed proof of Proposition 1 can be found in Appendix C of [15].

The notion of initialized fair Hintikka sequence can be easily captured by a generalized Büchi ECNA.

Theorem 3. *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, a 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. We first build 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. 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), $\mathcal{A}_\varphi$ simply uses the recorder clocks and predictor clocks: a transition having as source state an atom A has a clock constraint whose set of atomic constraints has the form

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

where $\widehat{I}$ is either $\{\vdash\}$ or a *maximal* interval over $\mathbb{R}_+$ disjunct from I. Finally, the generalized Büchi acceptance condition is exploited for checking that the input initialized Hintikka sequence is fair. The detailed construction of $\mathcal{A}_\varphi$ can be found in Appendix D of [15]. Note that $\mathcal{A}_\varphi$ has $2^{O(|\varphi|)}$ states and stack symbols, a set of constants $Const_\varphi$, 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 1, we obtain a generalized Büchi ECNA accepting the infinite models of φ. □

We can state now the main result of the section.

Theorem 4. *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 a set of constants $Const_\varphi$, 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 $\Sigma_{\mathscr{P}}$. By Theorem 3, one can construct a generalized Büchi ECNA $\mathscr{A}_\varphi$ over $\Sigma_{\mathsf{Cl}(\varphi)}$ having $2^{O(|\varphi|)}$ states and stack symbols, a set of constants $Const_\varphi$, and accepting the set of initialized fair Hintikka sequences of φ. By [14], one can construct a generalized Büchi VPTA $\mathscr{A}'_\varphi$ over $\Sigma_{\mathsf{Cl}(\varphi)}$ accepting $\mathscr{L}^\omega_T(\mathscr{A}_\varphi)$, having $2^{O(|\varphi|^2\cdot k)}$ states and stack symbols, $O(k)$ clocks, and a set of constants $Const_\varphi$, where k is the number of atomic constraints used by $\mathscr{A}_\varphi$. Note that $k = O(|\varphi|)$. Thus, by projecting the input symbols of the transition function of $\mathscr{A}'_\varphi$ over $\mathscr{P}$, we obtain a (generalized Büchi) VPTA satisfying the first part of Theorem 4.

For the upper bounds of the second part of Theorem 4, observe that by [12, 1] emptiness 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 $\mathscr{A}$ over $\Sigma_{\mathscr{P}}$ and an EC_NTL formula φ over $\Sigma_{\mathscr{P}}$, model-checking $\mathscr{A}$ against φ reduces to check emptiness of $\mathscr{L}^\omega_T(\mathscr{A}) \cap \mathscr{L}^\omega_T(\mathscr{A}'_{\neg\varphi})$, where $\mathscr{A}'_{\neg\varphi}$ is the generalized Büchi VPTA associated with $\neg\varphi$. Thus, since Büchi VPTA are polynomial-time closed under intersection, membership in EXPTIME for satisfiability and visibly model-checking of EC_NTL follow. The matching lower bounds follow from EXPTIME-completeness of satisfiability and visibly model-checking for the logic CaRet [3] which is subsumed by EC_NTL. □

5 Nested Metric Temporal Logic (NMTL)

Metric temporal logic (MTL) [19] 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 the CaRet modalities.

For the given set $\mathscr{P}$ of atomic propositions containing the special propositions *call*, *ret*, and *int*, the syntax of nested NMTL formulas φ is as follows:

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

where $p \in \mathscr{P}$, I is an interval in $\mathbb{R}_+$ with endpoints in $\mathbb{N} \cup \{\infty\}$, $dir \in \{\mathsf{g},\mathsf{a}\}$ and $dir' \in \{\mathsf{g},\mathsf{a},\mathsf{c}\}$. The operators $\widehat{\mathsf{U}}^{\mathsf{g}}_I$ and $\widehat{\mathsf{S}}^{\mathsf{g}}_I$ are the standard *timed until and timed since* MTL modalities, respectively, $\widehat{\mathsf{U}}^{\mathsf{a}}_I$ and $\widehat{\mathsf{S}}^{\mathsf{a}}_I$ are their non-regular abstract versions, and $\widehat{\mathsf{S}}^{\mathsf{c}}_I$ is the non-regular caller version of $\widehat{\mathsf{S}}^{\mathsf{g}}_I$. MTL with past is 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. For an NMTL formula φ, a timed word $w = (\sigma,\tau)$ over $\Sigma_{\mathscr{P}}$ and $0 \leq i < |w|$, the satisfaction relation $(w,i) \models \varphi$ is defined as follows (we omit the clauses for propositions and Boolean connectives):

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

In the following, we use some derived operators in NMTL:

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

Let $\mathscr{I}_{(0,\infty)}$ 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 J can be replaced by expressions of the form $\sim c$ for some $c \in \mathbb{N}$ and $\sim \in \{<,\leq,>,\geq\}$. We focus on the following two fragments of NMTL:

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

It is known that for the considered pointwise semantics, $\mathsf{MITL}_{(0,\infty)}$ [4] (the fragment of MTL allowing only intervals in $\mathscr{I}_{(0,\infty)}$) and $\mathsf{EC\_TL}$ are equally expressive [23]. Here, we easily generalize such a result to the nested extensions of $\mathsf{MITL}_{(0,\infty)}$ and $\mathsf{EC\_TL}$.

Lemma 1. *There exist effective linear-time translations from $\mathsf{EC\_NTL}$ into $\mathsf{NMITL}_{(0,\infty)}$, and vice versa.*

A proof of Lemma 1 can be found in Appendix E of [15]. By Lemma 1 and Theorem 4, we obtain the following result.

Theorem 5. *$\mathsf{EC\_NTL}$ and $\mathsf{NMITL}_{(0,\infty)}$ are expressively equivalent. Moreover, satisfiability and visibly model-checking for $\mathsf{NMITL}_{(0,\infty)}$ over finite (resp., infinite) timed words are* EXPTIME*-complete.*

In the considered pointwise semantics setting, it is well-known that satisfiability of MTL with past is undecidable [6, 21]. Undecidability already holds for future MTL interpreted over infinite timed words [21]. However, over finite timed words, satisfiability of future MTL is instead decidable [22]. Here, we show that over finite timed words, the addition of the future abstract timed modalities to future MTL makes the satisfiability problem undecidable.

Theorem 6. *Satisfiability of future NMTL interpreted over finite timed words is undecidable.*

We prove Theorem 6 by a reduction from the halting problem for Minsky 2-counter machines [20]. Fix such a machine M which is a tuple $M = (Lab, \mathsf{Inst}, \ell_{init}, \ell_{halt})$, where Lab is a finite set of labels (or program counters), $\ell_{init}, \ell_{halt} \in Lab$, and Inst is a mapping assigning to each label $\ell \in Lab \setminus \{\ell_{halt}\}$ an instruction for either (i) *increment*: $c_h := c_h + 1$; $\texttt{goto}\ \ell_r$, or (ii) *decrement*: $\texttt{if}\ c_h > 0\ \texttt{then}\ c_h := c_h - 1$; $\texttt{goto}\ \ell_s\ \texttt{else goto}\ \ell_t$, where $h \in \{1,2\}$, $\ell_s \neq \ell_t$, and $\ell_r, \ell_s, \ell_t \in Lab$.

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 [20]. We adopt the following notation, where $\ell \in Lab \setminus \{\ell_{halt}\}$:

- (i) if $\mathsf{Inst}(\ell)$ is an increment instruction of the form $c_h := c_h + 1$; $\texttt{goto}\ \ell_r$, define $c(\ell) := c_h$ and $succ(\ell) := \ell_r$; (ii) if $\mathsf{Inst}(\ell)$ is a decrement instruction of the form $\texttt{if}\ c_h > 0\ \texttt{then}\ c_h := c_h - 1$; $\texttt{goto}\ \ell_r\ \texttt{else goto}\ \ell_s$, 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 $\Sigma_{\mathscr{P}}$, where $\mathscr{P} = Lab \cup \{c_1, c_2\} \cup \{call, ret, int\}$. For a finite word $\sigma = a_1 \ldots a_n$ over $Lab \cup \{c_1, c_2\}$, we denote by σ^R the reverse of σ, and by $(call, \sigma)$ (resp., (ret, σ)) the finite word over $\Sigma_{\mathscr{P}}$ 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 $\Sigma_{\mathscr{P}}$ 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.

Formally, let $\mathscr{L}_{halt}$ be the set of finite words over $\Sigma_{\mathscr{P}}$ 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 $C \cdot C'$ of adjacent call-codes, the associated M-configurations, say (ℓ, n_1, n_2) and (ℓ', n'_1, n'_2), satisfy: $\ell \neq \ell_{halt}$ and (i) if $\mathsf{Inst}(\ell)$ is an increment instruction and $c(\ell) = c_h$, then $\ell' = succ(\ell)$ and $n'_h > 0$; (ii) if $\mathsf{Inst}(\ell)$ 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 $\ell \in Lab$.
- *Halting:* ℓ_{halt} occurs along σ.
- For each pair $C \cdot C'$ of adjacent call-codes in $(call, \sigma)$ with C' non-halting, the relative M-configurations (ℓ, n_1, n_2) and (ℓ', n'_1, n'_2) satisfy:[1] (i) *Increment requirement:* if $\mathsf{Inst}(\ell)$ is an increment instruction and $c(\ell) = c_h$, then $n'_h = n_h + 1$ and $n'_{3-h} = n_{3-h}$; (ii) *Decrement requirement:* if $\mathsf{Inst}(\ell)$ is a decrement instruction and $c(\ell) = c_h$, then $n'_{3-h} = n_{3-h}$, and, if $\ell' = dec(\ell)$, then $n'_h = n_h - 1$.

Evidently, M halts iff $\mathscr{L}_{halt} \neq \emptyset$. We construct in polynomial time a future NMTL formula φ_M over $\mathscr{P}$ such that the set of untimed components σ in the finite timed words (σ, τ) satisfying φ_M is exactly $\mathscr{L}_{halt}$. Hence, Theorem 6 directly follows. 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.

Formally, formula φ_M is given by $\varphi_M := \varphi_{WM} \vee \varphi_{\mathsf{LTL}} \vee \varphi_{Time}$ where 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 $\Sigma_{\mathscr{P}}$. By the formulas φ_{WM} and φ_{LTL}, we can assume that the untime part of w is of the form $(call, \sigma) \cdot (ret, \sigma^R)$ such that the call part $(call, \sigma)$ satisfies the consecution, initialization, and halting requirements. Then, 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 *Lab*-positions of two adjacent *call*-codes (resp., *ret*-codes) is 1. This can be expressed as follows:

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

- *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, the above requirement (*) ensures that $n'_h \geq n_h + 1$ and $n'_{3-h} \geq n_{3-h}$. In order to enforce 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.

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

[1]For technical convenience, we do not require that the counters in a configuration having as successor an halting configuration are correctly updated.

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

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.

Note that the unique abstract modality used in the reduction is $\bigcirc^{\mathsf{a}}$. This concludes the proof of Theorem 6.

6 Conclusions

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 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 the decidability properties 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 well-known decidable (and EXPSPACE-complete) fragment of MTL [4] which is strictly more expressive than $\mathsf{MITL}_{(0,\infty)}$ in the pointwise semantics setting [23].

References

[1] Parosh Aziz Abdulla, Mohamed Faouzi Atig & Jari Stenman (2012): *Dense-Timed Pushdown Automata*. In: *Proc. 27th LICS*, IEEE Computer Society, pp. 35–44, doi:10.1109/LICS.2012.15.

[2] Rajeev Alur & David L. Dill (1994): *A Theory of Timed Automata*. *Theoretical Computer Science* 126(2), pp. 183–235, doi:10.1016/0304-3975(94)90010-8.

[3] Rajeev Alur, Kousha Etessami & Parthasarathy Madhusudan (2004): *A Temporal Logic of Nested Calls and Returns*. In: *Proc. 10th TACAS*, *LNCS* 2988, Springer, pp. 467–481, doi:10.1007/978-3-540-24730-2_35.

[4] Rajeev Alur, Tomás Feder & Thomas A. Henzinger (1996): *The Benefits of Relaxing Punctuality*. *J. ACM* 43(1), pp. 116–146, doi:10.1145/227595.227602.

[5] Rajeev Alur, Limor Fix & Thomas A. Henzinger (1999): *Event-Clock Automata: A Determinizable Class of Timed Automata*. *Theoretical Computer Science* 211(1-2), pp. 253–273, doi:10.1016/S0304-3975(97)00173-4.

[6] Rajeev Alur & Thomas A. Henzinger (1993): *Real-Time Logics: Complexity and Expressiveness*. *Inf. Comput.* 104(1), pp. 35–77, doi:10.1006/inco.1993.1025.

[7] Rajeev Alur & Parthasarathy Madhusudan (2004): *Visibly Pushdown Languages*. In: *Proc. 36th STOC*, ACM, pp. 202–211, doi:10.1145/1007352.1007390.

[8] Christel Baier & Joost-Pieter Katoen (2008): *Principles of Model Checking*. The MIT Press.

[9] Massimo Benerecetti, Stefano Minopoli & Adriano Peron (2010): *Analysis of Timed Recursive State Machines*. In: *TIME 2010 - 17th International Symposium on Temporal Representation and Reasoning, Paris, France, 6-8 September 2010*, pp. 61–68, doi:10.1109/TIME.2010.10.

[10] Massimo Benerecetti & Adriano Peron (2016): *Timed recursive state machines: Expressiveness and complexity*. *Theoretical Computer Science* 625, pp. 85–124, doi:10.1016/j.tcs.2016.02.021.

[11] Devendra Bhave, Vrunda Dave, Shankara Narayanan Krishna, Ramchandra Phawade & Ashutosh Trivedi (2016): *A Logical Characterization for Dense-Time Visibly Pushdown Automata*. In: *Proc. 10th LATA*, *LNCS* 9618, Springer, pp. 89–101, doi:10.1007/978-3-319-30000-9_7.

[12] Ahmed Bouajjani, Rachid Echahed & Riadh Robbana (1994): *On the Automatic Verification of Systems with Continuous Variables and Unbounded Discrete Data Structures*. In: *Hybrid Systems II*, pp. 64–85, doi:10.1007/3-540-60472-3_4.

[13] L. Bozzelli, A. Murano & A. Peron (2010): *Pushdown Module Checking. Formal Methods in System Design* 36(1), pp. 65–95, doi:10.1007/s10703-010-0093-x.

[14] L. Bozzelli, A. Peron & A. Murano (2018): *Event-clock Nested Automata*. In: *Proc. 12th LATA*, LNCS 10792, Springer, pp. 80–92, doi:10.1007/978-3-319-77313-1_6.

[15] Laura Bozzelli, Aniello Murano & Adriano Peron (2018): *Timed contex-free temporal logics (extended version)*. *http://arxiv.org/abs/1808.04271*. Available at `http://arxiv.org/abs/1808.04271`.

[16] K. Chatterjee, D. Ma, R. Majumdar, T. Zhao, T.A. Henzinger & J. Palsberg (2003): *Stack Size Analysis for Interrupt-Driven Programs*. In: *Proc. 10th SAS*, LNCS 2694, Springer, pp. 109–126, doi:10.1007/3-540-44898-5_7.

[17] Lorenzo Clemente & Slawomir Lasota (2015): *Timed Pushdown Automata Revisited*. In: *Proc. 30th LICS*, IEEE Computer Society, pp. 738–749, doi:10.1109/LICS.2015.73.

[18] Michael Emmi & Rupak Majumdar (2006): *Decision Problems for the Verification of Real-Time Software*. In: *Proc. 9th HSCC*, *LNCS* 3927, Springer, pp. 200–211, doi:10.1007/11730637_17.

[19] Ron Koymans (1990): *Specifying Real-Time Properties with Metric Temporal Logic*. *Real-Time Systems* 2(4), pp. 255–299, doi:10.1007/BF01995674.

[20] M.L. Minsky (1967): *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs.

[21] Joël Ouaknine & James Worrell (2006): *On Metric Temporal Logic and Faulty Turing Machines*. In: *Proc. 9th FOSSACS*, LNCS 3921, Springer, pp. 217–230, doi:10.1007/11690634_15.

[22] Joël Ouaknine & James Worrell (2007): *On the decidability and complexity of Metric Temporal Logic over finite words*. *Logical Methods in Computer Science* 3(1), doi:10.2168/LMCS-3(1:8)2007.

[23] Jean-François Raskin & Pierre-Yves Schobbens (1999): *The Logic of Event Clocks - Decidability, Complexity and Expressiveness*. *Journal of Automata, Languages and Combinatorics* 4(3), pp. 247–286.

[24] Nguyen Van Tang & Mizuhito Ogawa (2009): *Event-Clock Visibly Pushdown Automata*. In: *Proc. 35th SOFSEM*, *LNCS* 5404, Springer, pp. 558–569, doi:10.1007/978-3-540-95891-8_50.

[25] Ashutosh Trivedi & Dominik Wojtczak (2010): *Recursive Timed Automata*. In: *Proc. 8th ATVA*, *LNCS* 6252, Springer, pp. 306–324, doi:10.1007/978-3-642-15643-4_23.

[26] I. Walukiewicz (1996): *Pushdown Processes: Games and Model Checking*. In: *CAV'96*, pp. 62–74, doi:10.1007/3-540-61474-5_58.