The Cost of Punctuality

Patricia Bouyer[1,2,*] Nicolas Markey[1] Joël Ouaknine[2] James Worrell[2]

[1] LSV, CNRS & ENS Cachan, France
{bouyer,markey}@lsv.ens-cachan.fr

[2] Oxford University, UK
{joel,jbw}@comlab.ox.ac.uk

Abstract

In an influential paper titled "The Benefits of Relaxing Punctuality" [2], Alur, Feder, and Henzinger introduced Metric Interval Temporal Logic (MITL) as a fragment of the real-time logic Metric Temporal Logic (MTL) in which exact or punctual timing constraints are banned. Their main result showed that model checking and satisfiability for MITL are both EXPSPACE-Complete.

Until recently, it was widely believed that admitting even the simplest punctual specifications in any linear-time temporal logic would automatically lead to undecidability. Although this was recently disproved, until now no punctual fragment of MTL was known to have even primitive recursive complexity (with certain decidable fragments having provably non-primitive recursive complexity).

In this paper we identify a 'co-flat' subset of MTL that is capable of expressing a large class of punctual specifications and for which model checking (although not satisfiability) has no complexity cost over MITL. Our logic is moreover qualitatively different from MITL in that it can express properties that are not timed-regular. Correspondingly, our decision procedures do not involve translating formulas into finite-state automata, but rather into certain kinds of reversal-bounded Turing machines. Using this translation we show that the model checking problem for our logic is EXPSPACE-Complete

1 Introduction

In the formal study of real-time systems, it has long been accepted that there is an unavoidable and substantial trade-off between the *expressiveness* of a specification formalism and the *feasability* of the associated verification task. This tension figures most prominently in the case of Metric Temporal Logic (MTL), a timed extension of Linear Temporal Logic (LTL), in which the temporal operators are constrained by time intervals.

*Partly supported by a Marie Curie fellowship.

MTL was introduced almost two decades ago by Koymans [14], and has since been extensively studied. Unfortunately, the model-checking and satisfiability problems for MTL over dense time are undecidable [3, 20], an extreme case of infeasibility. Researchers were therefore led to seek relaxations of the framework in a search for tractability. Alur and Henzinger, for example, proved that model checking MTL over discrete time was EXPSPACE-Complete [4]. Since untimed LTL model checking is already PSPACE-Complete, their result clearly sat towards the desirable end of the feasibility spectrum. The price they paid, however, was to renounce the density of time.

Accommodating time density, unfortunately, appeared to be problematic: it was widely held at the time that any formalism in which exact or punctual timing constraints could be expressed would automatically be undecidable. Such constraints correspond to allowing singleton intervals in MTL temporal operators, and enable one to specify, for example, that a particular event is to be followed exactly one time unit later by another one. In their seminal paper titled "The Benefits of Relaxing Punctuality" [2], Alur, Feder, and Henzinger therefore considered a fragment of MTL, called Metric Interval Temporal Logic (MITL), which syntactically bans punctual timing constraints. Their main achievement was to show that the model-checking and satisfiability problems for MITL are EXPSPACE-Complete. The proof they gave, in which MITL formulas are first transformed into timed automata, was quite complicated. Nevertheless, this work was influential as it firmly established MITL as the most important fragment of Metric Temporal Logic over dense-time having a feasible model-checking problem. In recent years, new or simplified proofs of the EXPSPACE-Completeness of MITL have appeared in the literature (e.g., [23, 13, 16]).

Recently, it was discovered that punctuality and dense time do not after all necessarily lead to undecidability, although the complexity of the various decidable fragments studied was either non-primitive recursive or non-elementary [19, 21, 22]. From a feasibility point of view, such improvements, while significant, remained unsatisfactory.

The aim of the present paper was therefore to investigate more thoroughly the intrinsic cost of allowing punctuality in a dense-time setting. Our main reasults concern two new 'punctual' fragments of Metric Temporal Logic, Bounded-MTL and coFlat-MTL.

Bounded-MTL is derived from MTL by requiring all time-constraining intervals to have finite length. As a result, the truth or falsity of a Bounded-MTL formula on a given timed word is determined by an initial segment of the word whose duration depends solely on the formula. We are then able to show that the model-checking and satisfiability problems for Bounded-MTL over dense time is EXPSPACE-Complete.

Bounded-MTL is therefore a punctual fragment of Metric Temporal Logic having precisely the same complexity as MITL. The two fragments, however, differ in important respects. A first observation is that, at a syntactic level, MITL restricts MTL in banning constraining intervals that are 'too small', whereas Bounded-MTL prohibits intervals that are 'too large'. Semantically, Bounded-MTL thus cannot express invariant properties, required to hold forever, contrary to MITL. In that respect, the expressiveness of Bounded-MTL is quite restricted.

Thankfully, it is possible to incorporate invariance into a substantially larger fragment of MTL. The principal contribution of this paper concerns the logic coFlat-MTL, which subsumes LTL, Bounded-MTL, and is closed under invariance. Our main result is that model checking this highly expressive punctual fragment of MTL is EXPSPACE-Complete. Perhaps surprisingly, satisfiability of coFlat-MTL, on the other hand, turns out to be undecidable.

Our proof of EXPSPACE membership proceeds by translating Flat-MTL formulas into alternating timed automata, and in turn simulating runs of these using special kinds of reversal-bounded Turing machines, for which termination can be shown to be in EXPSPACE. By contrast, MITL formulas are analysed by translation into timed automata, and, unlike Bounded-MTL and coFlat-MTL, can therefore only give rise to timed-regular languages.

MITL and coFlat-MTL have incomparable expressiveness. However, it can be argued that coFlat-MTL comprises virtually all the specifications that one could reasonably be interested to model check in practice. One might therefore view the dense-time logic coFlat-MTL as the first significant fragment of MTL to combine high expressiveness and punctuality together with model-checking feasibility.

2 Channel Automata

Before presenting our real-time modelling framework, we introduce a class of discrete machines that ultimately underly our model-checking algorithm for coFlat-MTL.

A *channel automaton* is a finite-state automaton equipped with a single unbounded fifo channel (or queue). The transitions of the automaton either write messages to the tail of the channel or read messages from the head of the channel. This model is easily seen to be Turing powerful [8]. In this paper we consider a class of channel automata with two extra primitives: *global renaming* and *occurrence testing*. The former allows a transition to simultaneously rename all the letters on the channel according to some renaming relation, including the possibility of deleting letters. The latter allows a transition to be guarded by the predicate that some letter not appear on the channel.

Given an alphabet Σ, let Σ_ε denote $\Sigma \cup \{\varepsilon\}$, where ε represents the empty word.

Definition 1. *A* Channel Automaton with Renaming and Occurrence Testing *(CAROT) is a tuple $\mathcal{C} = (S, s_0, \Sigma, \Delta, F)$, where S is a finite set of control states, $s_0 \in S$ is the initial control state, $F \subseteq S$ is a set of accepting control states, Σ is a finite channel alphabet and $\Delta \subseteq S \times Op \times S$ is the set of transition rules, with $Op = \{\sigma!, \sigma? \mid \sigma \in \Sigma\} \cup \{zero(\sigma) \mid \sigma \in \Sigma\} \cup \{R \mid R \subseteq \Sigma \times \Sigma_\varepsilon\}$ the set of operations. Given a rule $\tau \in \Delta$, we denote the corresponding operation $op(\tau)$. Intuitively, $zero(\sigma) \in Op$ guards against the occurrence of σ in the channel, and $R \in Op$ is interpreted as a global renaming (where renaming to ε corresponds to deletion).*

A *global state* of $\mathcal{C}$ is a pair $\gamma = (s, x)$, where $s \in S$ is the control state and $x \in \Sigma^*$ is the channel contents. The rules in Δ induce a transition relation on the set of global states according to the following table, where, given $x = x_1 \ldots x_n \in \Sigma^*$ and $R \subseteq \Sigma \times \Sigma_\varepsilon$, $R(x) \stackrel{\text{def}}{=} \{y_1 \cdots y_n \in \Sigma^* : x_i \; R \; y_i\}$.

Rule	Transition
$(s, \sigma!, t)$	$(s, x) \to (t, x \cdot \sigma)$
$(s, \sigma?, t)$	$(s, \sigma \cdot x) \to (t, x)$
$(s, zero(\sigma), t)$	$(s, x) \to (t, x)$, if $\sigma \notin x$
(s, R, t)	$(s, x) \to (t, y)$, if $y \in R(x)$

Assume that Σ always contains a special symbol $\triangleright$, called the *end-of-channel marker*. A computation of $\mathcal{C}$ is a (finite or infinite) sequence of transitions $\gamma_0 \to \gamma_1 \to \gamma_2 \to \cdots$ with $\gamma_0 = (s_0, \triangleright)$. A finite computation is *accepting* if it ends in an accepting state γ_n.

To aid our analysis of computations, we make the following (harmless) assumption about $\mathcal{C}$. We suppose that given *consecutive* rules (s, op_1, t) and (t, op_2, u), $op_1 = \triangleright?$ iff $op_2 = \triangleright!$: roughly speaking, this ensures that there is always a unique copy of $\triangleright$ on the channel. This restriction allows us to use the end-of-channel marker to measure the number of *cycles* of the channel during a computation. Intuitively a segment of the computation during which $\triangleright$ moves from the tail of the channel to the head of the channel involves a complete cycle of the channel. Formally we define

s	$\boxed{b!}$	s	$\boxed{b!}$	s	R	t	$\triangleright?$	u			$\boxed{\triangleright!}$	v						
	$\boxed{b?}$	v	$\boxed{c?}$	s	$\boxed{a!}$	s	$\boxed{b!}$	s	R	t	$\boxed{\triangleright?}$	u	$\boxed{\triangleright!}$	v				
					$\boxed{a?}$	v	$\boxed{c?}$	s	$\boxed{b!}$	s	R	t	$\boxed{\triangleright?}$	u	$\boxed{\triangleright!}$	v		
									$\boxed{c?}$	s	R	u			$\boxed{\triangleright?}$	u	$\triangleright!$	v

Fig. 1. Computation table

s	$b!$	s	$b!$	s	R	t	$\triangleright?$	u	u	u	$\triangleright!$	v	v	v	v	v	v	v
v	$b?$	v	$c?$	s	$a!$	s	$b!$	s	R	t	$\triangleright?$	u	$\triangleright!$	v	v	v	v	v
v	v	v	v	v	$a?$	v	$c?$	s	$b!$	s	R	t	$\triangleright?$	u	$\triangleright!$	v	v	v
v	v	v	v	v	v	v	v	v	$c?$	s	R	u	u	u	$\triangleright?$	u	$\triangleright!$	v

Fig. 2. Computation table with sliding window

cycles(ϱ) to be the number of transitions in ϱ with operand $\triangleright!$. This measure is similar to the notion of head reversals for Turing machines.[1]

Definition 2. *The* cycle-bounded reachability problem *for CAROTs is as follows:*
Instance*: A CAROT $\mathcal{C}$ and a cycle bound N.*
Question*: Does $\mathcal{C}$ have an accepting computation ϱ with cycles$(\varrho) \leqslant N$?*

In the channel automaton $\mathcal{C}$ below, let R be the relation that nondeterministically renames b to either b or c.

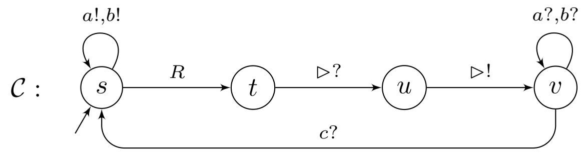

Figure 1 represents a computation of $\mathcal{C}$ in tabular form. Each row of the table represents a cycle of the channel, and, reading left-to-right, it records the sequence of transitions during that cycle. The most important property of the table is that the spacing is arranged so that an operation that reads a message is placed directly below the operation that originally wrote the message, necessarily in the previous cycle of the channel.[2] In Figure 1 matching pairs of reads and writes are indicated by rectangular boxes. Because of global renaming, the corresponding read and write events need not refer to the same element of Σ. For instance, in Figure 1, a write-event $b!$ is sometimes aligned with a read-event $c?$.

The length of the computation table (*i.e.*, the number of columns) is at least the maximum length of the channel during the corresponding computation. It is easy to see that this can be exponential in the value of the cycle bound. (Consider a machine that repeatedly reads one copy of σ and writes two copies of σ.) However we describe a procedure to guess the existence of a computation table using only polynomial space in the value of the cycle bound. The first step is to fill in the blank spaces in the table by repeating the immediately preceding control state; for example, starting from Figure 1 we obtain the table in Figure 2.

A nondeterministic procedure for guessing and verifying such a table involves storing only part of the table in memory at any one time. Imagine a sliding window of dimension $3 \times h$, where h is the table height (*i.e.*, the number of rows). The window represents the part of the table in view at any time; it starts at the left end and is moved one place to the right with each phase of the procedure. Given a particular view, a phase of the procedure checks that the transitions therein are consistent with the control structure of $\mathcal{C}$. For instance, in Figure 2, while viewing the leftmost window it is checked that $(s, b!, s), (v, b?, v) \in \Delta$. Somewhat more subtly the corresponding read and write events in the current view must be consistent with the zero testing and renaming in the rest of the table. For instance, in Figure 2, again in the left-most window, the justification of the vertically aligned $b!$ and $b?$ operations is that between the occurrence of these two operations b was neither renamed nor zero-tested.

In general, what is required is to store in memory the cumulative effect of the zero tests and renaming in the part of the table not currently in view. To this end, we associate with each rule $\tau \in \Delta$ a relation R_τ on Σ_ε according to the value of $op(\tau)$. The table below shows this association, where *Id* is the identity relation on Σ_ε.

$op(\tau)$	$\sigma!, \sigma?$	$zero(\sigma)$	R
R_τ	*Id*	$Id - \{(\sigma, \sigma)\}$	$R \cup \{\varepsilon, \varepsilon\}$

Now suppose that on row i of the computation table the sequence of transition rules is $\tau_1, \tau_2, \ldots, \tau_n$, and that τ_j is the transition currently in view. Then the sliding-window procedure stores in memory a pair of relations $Left_i$ and $Right_i$ on Σ_ε, where $Left_i = R_{\tau_j} \circ \cdots \circ R_{\tau_1}$ and $Right_i = R_{\tau_n} \circ \cdots \circ R_{\tau_{j+1}}$. Note that $Right_i$ must be guessed since it refers to the part of the table to the right of the current view, which has not been seen yet. The correctness criterion on the current view is that if $\sigma, \sigma' \in \Sigma$ are vertically aligned, with $\sigma!$ on row i and $\sigma'?$ on row $i+1$, then $\sigma\ (Left_{i+1} \circ Right_i)\ \sigma'$. Finally, observe that it is straightforward to verify the consistency of the guessed value of $Right_i$ from one view to the next.

Theorem 3. *The cycle-bounded reachability problem for CAROTs is solvable in polynomial space in the size of the channel automaton and the value of the cycle bound.*

[1] Formally, it can be shown that an N-cycle-bounded CAROT can simulate an N-reversal-bounded single-tape Turing machine, and *vice-versa*.

[2] To accommodate global deletion in such a table (i.e., renaming to ε), we postulate a self-loop $(s, \varepsilon?, s)$ for each control state s of $\mathcal{C}$.

3 Metric Temporal Logic

In this section we formally define the syntax and semantics of Metric Temporal Logic. Following [10, 11, 12, 25, 4, 5], among others, we interpret the logic over timed words: ω-sequences of events with associated timestamps.[3]

Definition 4. *The syntax of* Metric Temporal Logic *(*MTL*) [14] is defined by the following grammar:*

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

where σ ranges over a finite set of events Σ and I is an interval of $\mathbb{R}^+$ with bounds in $\mathbb{N} \cup \{\infty\}$.

MTL formulas are interpreted over timed words: a timed word w is an infinite sequence $(\sigma_i, t_i)_{i\in\mathbb{N}}$ where $\sigma_i \in \Sigma$ and $t_i \in \mathbb{R}^+$ for each i, and such that the sequence $(t_i)_{i\in\mathbb{N}}$ is strictly increasing and diverges to infinity.

Definition 5. *Let $w = (\sigma_i, t_i)_{i\in\mathbb{N}}$ be an infinite timed word, and $k \in \mathbb{N}$. The (pointwise) semantics of* MTL *is defined recursively as follows (we omit Boolean operations):*

$$\begin{aligned} w, k \models \sigma \; &\Leftrightarrow \; \sigma_k = \sigma \\ w, k \models \varphi\, \mathbf{U}_I\, \psi \; &\Leftrightarrow \; \exists i > 0.\ w, k+i \models \psi,\ t_{k+i} - t_k \in I \\ &\qquad \textit{and } \forall 0 < j < i,\ w, k+j \models \varphi \\ w, k \models \varphi\, \widetilde{\mathbf{U}}_I\, \psi \; &\Leftrightarrow \; w, k \models \neg\Big((\neg\varphi)\, \mathbf{U}_I\, (\neg\psi)\Big). \end{aligned}$$

If $w, 0 \models \varphi$, we write $w \models \varphi$.

Additional operators, such as $\mathbf{t}$ (true), $\mathbf{f}$ (false), $\Rightarrow$, $\Leftrightarrow$, $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{X}$, are defined in the usual way: $\mathbf{F}_I\, \varphi \equiv \mathbf{t}\, \mathbf{U}_I\, \varphi$, $\mathbf{G}_I\, \varphi \equiv \mathbf{f}\, \widetilde{\mathbf{U}}_I\, \varphi$, and $\mathbf{X}_I\, \varphi \equiv \mathbf{f}\, \mathbf{U}_I\, \varphi$. We also use pseudo-arithmetic expressions to denote intervals. For example, '$= 1$' denotes the singleton $\{1\}$.

Let us point out that the main results of this paper also hold under a weakly monotonic semantics for time (in which the timestamps are merely nondecreasing), as well as under a non-strict semantics for temporal operators (in which the present time point is included).

3.1 Satisfiability and model checking

We consider the following two fundamental questions for MTL and various fragments thereof:

- The *satisfiability problem*, asking whether a given MTL formula φ is satisfiable, *i.e.*, whether $w \models \varphi$ for some infinite timed word w over Σ;
- The *model-checking problem*, asking whether a given timed automaton $\mathcal{A}$ satisfies a given MTL formula φ, *i.e.*, whether all timed words accepted by $\mathcal{A}$ satisfy φ (see [1] for details). We write $\mathcal{A} \models \varphi$ when the answer is positive.

[3] This is the so-called *pointwise* semantics. Another semantics, interval-based, is interpreted over continuous signals. See e.g. [11, 23] for details. As noted in [11], the known complexity results for MITL hold both in the interval-based and in the pointwise semantics.

Among others, we identify the following syntactic fragments of MTL. *Linear Temporal Logic* (LTL) can be considered as the fragment of MTL in which modalities are not constrained (*i.e.*, where $\mathbb{R}^+$ is the only constraining interval). *Metric Interval Temporal Logic* (MITL) is the fragment of MTL where punctuality is not allowed (*i.e.*, where interval constraints are not singletons). Bounded-MTL is the fragment of MTL in which all interval constraints have finite length.

MITL was introduced in [2], motivated by the role played by punctuality in the undecidability proof for MTL. The main result of [2] was that model checking and satisfiability for MITL are EXPSPACE-Complete. As we will see, these problems are also EXPSPACE-Complete for Bounded-MTL. This is somewhat surprising in view of the following example.

Example 6. *Let the* variability *of a timed word be the maximum number of events that occur in any one time unit. We exhibit a family of* Bounded-MTL *formulas $\{\varphi_n\}_{n\in\mathbb{N}}$ such that the size of φ_n is linear in n, but the variability of any timed word satisfying φ_n is at least 2^{2^n}, i.e.,* doubly exponential *in n. We define $\varphi_n \equiv a \wedge \varphi_D \wedge \mathbf{G}_{[0,2^n]}\, \varphi_D$, where $\varphi_D \equiv (a \to \mathbf{F}_{=1}\, (a \wedge \mathbf{X}_{\leqslant 1} b)) \wedge (b \to \mathbf{F}_{=1}\, (a \wedge \mathbf{X}_{\leqslant 1} b))$. If $\varrho \models \varphi_n$, then the variability of ϱ must (at least) double every time unit over the first 2^n time units.*

Observe that while Bounded-MTL permits punctual formulas, it disallows unconstrained modalities. In particular, Bounded-MTL is not suitable to express invariance—the most basic type of temporal specification—and it does not subsume LTL (either syntactically or semantically). Intuitively, Bounded-MTL is only suitable for expressing time-bounded specifications. To remedy this deficiency we introduce Flat-MTL as the fragment of MTL generated by the grammar:

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

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

Notice immediately that Flat-MTL subsumes both LTL and Bounded-MTL, however it is not closed under negation. In fact, the most natural way to state our main results is in terms of the dual logic, which we call coFlat-MTL. This consists of the duals (*i.e.*, the negations) of Flat-MTL formulas. Correspondingly, the syntactic restriction determining coFlat-MTL as a subset of MTL is dual to that determining Flat-MTL: we require that, if I is unbounded, then formulas appearing on the right of $\mathbf{U}_I$ and on the left of $\widetilde{\mathbf{U}}_I$ be LTL formulas.

Like Flat-MTL, coFlat-MTL includes both LTL and Bounded-MTL. However, crucially, it is also closed under $\mathbf{G}_I$ for unbounded I, since $\mathbf{G}_I\,\varphi \equiv \mathbf{f}\,\widetilde{\mathbf{U}}_I\,\varphi$. Thus we have the slogan:

$$\textsf{Bounded-MTL} + \text{Invariance} \subseteq \textsf{coFlat-MTL}\,.$$

This means that one can express a much more useful class of specifications in coFlat-MTL than in Bounded-MTL. In particular, a wide variety of safety specifications can be expressed in the form $\mathbf{G}\,\varphi$, where φ is in Bounded-MTL.

The main result of this paper is that the model-checking problem for coFlat-MTL is EXPSPACE-Complete. This last problem can be understood as a slight generalisation of the satisfiability problem for the dual logic Flat-MTL.

Example 7. *The formula* $\mathbf{G}\,(req \Rightarrow \mathbf{F}_{[0,1]}\,(acq \wedge \mathbf{F}_{=1}\,rel))$ *says that every time the lock is requested, it is acquired within one time unit, and released after exactly one further time unit. This formula is in* coFlat-MTL*, but is not in* Bounded-MTL *(due to the unconstrained* $\mathbf{G}$ *) and is not in* MITL *(due to the punctual* $\mathbf{F}_{=1}$ *).*

Given a timed automaton $\mathcal{A}$*, to find a violation of the above formula one must search for a run of* $\mathcal{A}$ *such that after some request event, every acquire event in the subsequent time unit fails to be followed after exactly one time unit by a release event. Intuitively, over a dense-time semantics, this task seems to require 'remembering' a potentially unbounded amount of information. Thus our EXPSPACE-Completeness result for model checking* coFlat-MTL *may appear surprising.*

For comparison with previous work we describe one more fragment of MTL, called Safety-MTL [19, 21]. This is determined by the restriction that the Until modality only be constrained by bounded intervals. Like coFlat-MTL, Safety-MTL includes Bounded-MTL and is closed under $\mathbf{G}$, but, unlike coFlat-MTL, satisfiability is decidable for Safety-MTL whereas model checking is non-elementary.

We summarise the relationships between the various logics introduced above in the following diagram (where $\hookrightarrow$ indicates a syntactic inclusion):

$$\begin{tikzcd}
 & \textsf{Safety-MTL} \arrow[dr, hook] & \\
\textsf{Bounded-MTL} \arrow[ur, hook] \arrow[r, hook] & \textsf{coFlat-MTL} \arrow[r, hook] & \textsf{MTL} \\
\textsf{LTL} \arrow[ur, hook] \arrow[r, hook] & \textsf{MITL} \arrow[ur, hook] &
\end{tikzcd}$$

	Model Checking	**Satisfiability**
LTL	PSPACE-C.	PSPACE-C.
MITL	EXPSPACE-C.	EXPSPACE-C.
Bounded-MTL	EXPSPACE-C.	EXPSPACE-C.
Safety-MTL	Non-Prim.-Rec.	Non-Elem.
coFlat-MTL	EXPSPACE-C.	Undec.
MTL	Undec.	Undec.

Table 1. Complexity of fragments of MTL (interpreted over infinite timed words)

3.2 Main results

Table 1 summarizes the complexity of the fragments of MTL defined above. Dark gray boxes correspond to results stated and proved elsewhere, whereas light gray boxes correspond to results that can be deduced straightforwardly from other papers. The undecidability of MTL is proved in [20], while MITL and Safety-MTL have been defined and studied respectively in [2] and in [19, 21].

In this paper, we state the following results:

- The model-checking problem for coFlat-MTL is in EXPSPACE (see sections 4, 5), which immediately implies the same result for Bounded-MTL.
- The model-checking and satisfiability problems for Bounded-MTL are EXPSPACE-Hard, which immediately implies that coFlat-MTL model checking is also EXPSPACE-Hard.

For lack of space, we refer to [7] for full details.

In addition, it is worth noticing that the undecidability proof of [20] for the satisfiability of MTL over infinite words can also be used to prove that the satisfiability problem for coFlat-MTL (and thus the model-checking problem for Flat-MTL) is undecidable. The result that the satisfiability problem for Safety-MTL is non-elementary is a consequence of [6].

The proof that the model checking of coFlat-MTL is in EXPSPACE can be sketched as follows: (i) if φ is the formula that we want to verify, we first construct an alternating timed automaton (ATA) which recognizes all models of $\neg\varphi$ (Section 4.1); (ii) we then prove properties of that ATA (Section 4.2); (iii) we construct a CAROT which will simulate joint executions of the automaton we want to model check and the above-mentioned ATA (Section 5).

4 Alternating Timed Automata

In this section, we recall the definition of one-clock alternating timed automata (ATA), a natural timed analog of alternating automata [15, 19]. ATA generalise classical (Alur-Dill) timed automata [1], and, unlike the latter, are closed under complement. However language-emptiness is, in general, undecidable for ATA.

Let L be a finite set of locations and x a clock variable. We define $\Phi(L, x)$ as the set of formulas defined by the grammar '$\varphi ::= \mathbf{t} \mid \mathbf{f} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \ell \mid x \sim c \mid x.\varphi$' where $\ell \in L$, $\sim \in \{<, \leqslant, =, \geqslant, >\}$ and $c \in \mathbb{N}$.

Definition 8. *A (one-clock) alternating timed automaton $\mathcal{A}$ is a tuple $(L, \Sigma, \delta_0, \delta, F)$ where L is a finite set of locations, Σ is a finite set of actions, $\delta_0 \in \Phi(L, x)$ is an initial condition, $\delta\colon L \times \Sigma \to \Phi(L, x)$ is the transition relation, and $F \subseteq L$ is a set of accepting locations.*

Given a timed alternating automaton $\mathcal{A}$, let M be the maximum constant mentioned in the clock contraints in $\mathcal{A}$. Define the set of clock values Val to be $[0, M] \cup \{\bot\}$. Here $\bot$ represents any clock value strictly greater than M.[4] The set of *states* of $\mathcal{A}$ is $L \times \mathsf{Val}$. A set of states C and a clock value $v \in \mathsf{Val}$ defines a Boolean valuation on the set of formulas $\Phi(L, x)$ as follows (we omit obvious cases):

$$C \models_v x \sim c \;\Leftrightarrow\; v \sim c \qquad C \models_v \ell \;\Leftrightarrow\; (\ell, v) \in C$$
$$C \models_v x.\varphi \;\Leftrightarrow\; C \models_0 \varphi$$

We say that C is a *minimal model* of $\varphi \in \Phi(L, x)$ with respect to the clock value v if $C \models_v \varphi$ and if there is no proper subset $C' \subseteq C$ such that $C' \models_v \varphi$.

Let S be a set of letters, and S^* be the set of finite words over S. A *tree* τ over S is a subset of S^* such that (i) for every word $s_0\, s_1\, \cdots\, s_p \in \tau$, we also have $s_0\, s_1\, \cdots\, s_{p-1} \in \tau$; (ii) if $s_0, \cdots s_p$ and $s'_0, \cdots s'_q$ are two words of τ, then $s_0 = s'_0$. The word s_0 is the *root* of τ. An element of a tree is called a *node*. Let $\nu = s_0\, \cdots\, s_p$ be a node. The depth of ν is p, its label is s_p, and its successors have set of labels $\mathsf{succ}(\nu) = \{s' \in \Sigma \mid s_0\, \cdots\, s_p\, s' \in \tau\}$. A *branch* of a tree is a maximal (finite or infinite) sequence of nodes $(\nu_i)_i$ such that ν_i is a prefix of ν_{i+1} for each i. A *forest* is a finite set of trees.

Definition 9. *An* execution *of an ATA over a timed word $w = (\sigma_i, t_i)_{i \in \mathbb{N}}$ is a forest $\{\tau_1, ..., \tau_k\}$ over the set $L \times \mathsf{Val}$, such that (i) any root (ℓ, v) is such that $v = 0$, and the set of roots satisfies the initial condition δ_0 under valuation 0; (ii) for each node ν of the forest of depth p and label (ℓ, v), letting $v' = v + t_p - t_{p-1}$ (where $t_{-1} = 0$), we have that $\mathsf{succ}(\nu)$ is a minimal model of $\delta(\ell, \sigma_p)$ with respect to v'.*

We use a Büchi acceptance condition: an execution forest is accepting *if every infinite branch in the forest contains infinitely many nodes whose labels are in $F \times \mathsf{Val}$.*

[4] Such clock values are indistinguishable by clock constraints in $\mathcal{A}$, so this identification is harmless. We require $\bot$ to satisfy the obvious arithmetic properties, e.g., $\bot + t = \bot$ for all $t \in \mathbb{R}^+$.

Observe that any execution forest is finitely branching, due to the minimality assumption on $\mathsf{succ}(\nu)$.

We say that an execution is *memoryless* if for any two identically labelled nodes ν_1 and ν_2 of same depth, the respective subtrees rooted at ν_1 and ν_2 are identical. There is no loss of generality in restricting to memoryless executions. If $\mathcal{A}$ has an execution on a timed word w then, by a result of Emerson and Jutla [9] on the memoryless determinacy of parity games, it can also be shown that $\mathcal{A}$ has a memoryless execution on w. Details can be found in [7].

4.1 Translating Formulas into Automata

Following the construction given in [19], from any MTL formula φ, we can derive an ATA $\mathcal{B}_\varphi$ that recognizes exactly the set of infinite timed words satisfying φ. Rather than recall the precise definition of $\mathcal{B}_\varphi$ here, we axiomatise its key properties for later use. We refer the reader to [7] for full details.

The set of locations of $\mathcal{B}_\varphi$ is the set $\mathrm{Sub}(\varphi)$ of modal subformulas of φ. Moreover, the transition function δ of $\mathcal{B}_\varphi$ satisfies the following three axioms.
(1) *Linearity:* if $\psi, \psi' \in \mathrm{Sub}(\varphi)$ are such that ψ' appears in $\delta(\psi, \sigma)$, then ψ' is a subformula of ψ. In particular, the only loops in $\mathcal{B}_\varphi$ are self loops.
(2) *Locality:* every location appearing in $\delta(\psi, \sigma)$ other than ψ appears under the scope of a reset '$x.$'. Furthermore, ψ never occurs in $\delta(\psi, \sigma)$ under the scope of a reset.

If $\varphi \in \mathsf{Flat\text{-}MTL}$, then $\mathcal{B}_\varphi$ also satisfies the following *flatness* property:
(3) *Flatness:* If $\psi \in \mathrm{Sub}(\varphi)$ is an LTL formula, then $\delta(\psi, \sigma)$ contains no clock constraints. Otherwise, $\delta(\psi, \sigma)$ has the form $((x \leqslant c) \wedge \varphi_1) \vee \varphi_2 \vee \varphi_3$, where $\varphi_1, \varphi_2, \varphi_3 \in \Phi(\mathrm{Sub}(\varphi), x)$ are such that φ_2 only mentions locations in $\{\psi\} \cup \mathsf{LTL}$ and φ_3 doesn't mention ψ. This last condition can be read as follows: after a certain amount of time, location ψ cannot make a simultaneous transition to itself and another location in $\mathrm{Sub}(\varphi) \setminus \mathsf{LTL}$.

4.2 Ranking Flat-MTL

In this section, we analyse the structure of the execution forests of those ATA arising from $\mathsf{Flat\text{-}MTL}$ formulas. Roughly speaking, the main result of this section, Theorem 12, says that the segments of such an execution forest in which the automaton clocks are *active* have a short total duration. Here we say that a clock (value) is inactive if it is greater than the maximum clock constant M of the automaton, otherwise we say that it is active.

In the rest of this section let $\mathcal{B}_\varphi$ denote an ATA arising from a $\mathsf{Flat\text{-}MTL}$ formula φ, and let M be the maximum clock constant of $\mathcal{B}_\varphi$. Recall that the set of locations of $\mathcal{B}_\varphi$ is the set $\mathrm{Sub}(\varphi)$ of modal subformulas of φ. By extension, we say that a state (ψ, v) of $\mathcal{B}_\varphi$ is inactive if ψ is an LTL formula or if $v > M$.

Given an execution forest of $\mathcal{B}_\varphi$, its i-th configuration is the set of states labelling the nodes at depth i. An execution forest of an ATA thus generates a sequence of configurations $\varrho\colon C_0 \to C_1 \to \ldots \to C_k \to \ldots$. Next we define a rank function on configurations based on the distinction between active and inactive clocks.

Let $<$ be a linear order on $\mathrm{Sub}(\varphi)$ such that $\varphi_1 < \varphi_2$ whenever φ_1 is a subformula of φ_2 (one such can always be chosen). Furthermore, let Γ denote the set $\mathrm{Sub}(\varphi) \times \{\bot, \top\}$ ordered lexicographically, where $\bot < \top$. We think of $\top$ as representing an active clock, whereas (following the notation introduced in Section 4) $\bot$ denotes an inactive clock.

Definition 10. *Given a configuration C of $\mathcal{B}_\varphi$, let the non-LTL formulas occurring in C be written $\{\varphi_i\}_{i=1}^k$, where $\varphi_k > \varphi_{k-1} > \ldots > \varphi_1$. If none of the φ_i is paired with an active clock in C, then we define $\mathsf{rank}(C)$ to be the word $(\varphi_k, \bot) \ldots (\varphi_2, \bot)(\varphi_1, \bot)$. Otherwise, let φ_j be the maximum among all formulas appearing in C that are paired with an active clock, and define $\mathsf{rank}(C)$ to be the word $(\varphi_k, \bot) \ldots (\varphi_{j+1}, \bot)(\varphi_j, \top)$. We order the ranks of configurations according to the lexicographic order on Γ^*, denoted $\preceq$.*

Example 11. *Let the maximum clock constant in φ be $M = 3$ and let $C = \{(\varphi_1, 2.4), (\varphi_1, \bot), (\varphi_2, 0.8), (\varphi_2, \bot), (\varphi_3, \bot), (\varphi_4, \bot)\}$ be a configuration of $\mathcal{B}_\varphi$, where $\varphi_4 > \varphi_3 > \varphi_2 > \varphi_1$. Then $\mathsf{rank}(C) = (\varphi_4, \bot)(\varphi_3, \bot)(\varphi_2, \top)$, that is, we record the maximum active state and all inactive states above it.*

Let $\varrho\colon C_0 \to C_1 \to \ldots \to C_k \to \ldots$ be a sequence of configurations in a run of $\mathcal{B}_\varphi$ on the timed word $(\sigma_i, t_i)_{i\in\mathbb{N}}$. Given an interval $I \subseteq \mathbb{N}$, write $\varrho[I]$ for the subsequence of ϱ consisting of those C_i with $i \in I$. Furthermore, define the *active duration* of $\varrho[I]$, denoted $\mathsf{duration}(\varrho[I])$, to be 0 if none of the C_i, $i \in I$, contains an active clock, and $t_{\sup(I)} - t_{\inf(I)} + M$ otherwise, where $t_\infty = \infty$. Intuitively, $\mathsf{duration}(\varrho[I])$ gives an upper bound for the amount of time that an active clock is present in the run segment $\varrho[I]$. (In case $I = \{i, \ldots, j\}$ is finite, this segment includes the time delay between positions j and $j+1$ in ϱ, hence the extra term M in the expression for $\mathsf{duration}(\varrho[I])$.)

Theorem 12. *Let $\varrho\colon C_0 \to C_1 \to \ldots \to C_k \to \ldots$ be the sequence of configurations of a memoryless run of $\mathcal{B}_\varphi$. Then there is a partition $\mathcal{I}$ of $\mathbb{N}$ into at most $|\varphi| \cdot 2^{|\varphi|}$ intervals, where for each interval I in $\mathcal{I}$, $\varrho[I]$ has active duration at most $2M+1$, the last interval is (unbounded and) fully inactive, and $|\varphi|$ is the number of non-LTL modal subformulas of φ.*

Proof. Define an equivalence $\equiv$ on $\mathbb{N}$ by $n \equiv m$ iff $\mathsf{rank}(C_n) = \mathsf{rank}(C_m)$. Now Lemma 13 (below) says that rank is non-increasing along ϱ; it follows that the equivalence classes of $\equiv$ are intervals. Furthermore, the index of the equivalence relation is bounded by the number of ranks, which is easily seen to be no more than $|\varphi| \cdot 2^{|\varphi|}$. Finally, it follows from Lemma 14 (below) that the active duration of any equivalence class is at most $2M+1$. □

It remains to prove the two technical lemmas quoted in the proof of Theorem 12.

Lemma 13. *If $\varrho\colon C_0 \to C_1 \to \ldots \to C_k \to \ldots$ is the sequence of configurations in a memoryless run of $\mathcal{B}_\varphi$, then $\mathsf{rank}(C_{n+1}) \preceq \mathsf{rank}(C_n)$ for each $n \in \mathbb{N}$.*

Proof. We split the transition from C_n to C_{n+1} into two steps: a time-elapse step, where each clock in C_n increases by some fixed amount, and a discrete step, where state changes are performed according to the transition function of $\mathcal{B}_\varphi$. We show that neither of these steps is rank-increasing.

For the time-elapse step, observe that for any configuration C and time delay $t \in \mathbb{R}_+$, $\mathsf{rank}(C+t) \preceq \mathsf{rank}(C)$, where $C + t = \{(\psi, v+t) : (\psi, v) \in C\}$. This is because the only possible difference between C and $C+t$ is that active clocks in C may become inactive in $C+t$; but this cannot increase the rank (reflecting the fact that $\bot < \top$).

Write δ for the transition function of $\mathcal{B}_\varphi$, and let $\sigma \in \Sigma$. For the discrete step, suppose that $C = \{(\psi_i, v_i)\}_{i\in I}$ is a configuration and that $C' = \bigcup_i D_i$, where D_i is a minimal model of $\delta(\psi_i, \sigma)$ with respect to v_i for each $i \in I$.[5] Furthermore, for a contradiction, suppose that $\mathsf{rank}(C) \prec \mathsf{rank}(C')$, with $\gamma \in \Gamma$ the letter in $\mathsf{rank}(C')$ occurring in the first position in which $\mathsf{rank}(C)$ and $\mathsf{rank}(C')$ differ. Since the letters in $\mathsf{rank}(C)$ appear in descending order we can assume that γ does not appear in $\mathsf{rank}(C)$ at all. We consider two cases according to whether γ is inactive or active.

The first case is that $\gamma = (\psi, \bot)$ for some $\psi \in \mathrm{Sub}(\varphi) \setminus \mathsf{LTL}$. Then there exists $i \in I$ such that $(\psi, \bot) \in D_i$. By locality of $\mathcal{B}_\varphi$ (cf. Section 4.1), we must have $\psi_i = \psi$ and $v_i = \bot$. Thus $\gamma = (\psi, \bot)$ appears in $\mathsf{rank}(C)$, contradicting the assumption on γ.

The second case is that $\gamma = (\psi, \top)$ for some $\psi \in \mathrm{Sub}(\varphi) \setminus \mathsf{LTL}$. Then there exists $i \in I$ and a clock value $v \leqslant M$ such that $(\psi, v) \in D_i$. By linearity of $\mathcal{B}_\varphi$ we have $\psi \leqslant \psi_i$; if also $v_i \leqslant M$ then some active state at least as high as $(\psi, \top)$ appears in $\mathsf{rank}(C)$. Since $\mathsf{rank}(C)$ and $\mathsf{rank}(C')$ agree on letters higher than γ, this active state can only be γ itself, which contradicts our hypothesis on γ. Thus we may assume that $v_i = \bot$.

[5] Since ϱ is memoryless, the set of states at each configuration in ϱ can always be calculated from the set of states of the previous configuration in this manner.

But then, since $v_i \neq v$, by locality of $\mathcal{B}_\varphi$ it must hold that $\psi < \psi_i$, and by flatness of $\mathcal{B}_\varphi$, we have that ψ_i does not appear in D_i. (Flatness dictates that ψ and ψ_i cannot both appear in D_i.) In fact, we can conclude that $(\psi_i, \bot)$ does not appear in C' (by locality of $\mathcal{B}_\varphi$ it cannot appear in D_j for $j \neq i$). But then (ψ_i, v_i) does not appear in $\mathsf{rank}(C')$ and $(\psi_i, v_i) > \gamma$, contradicting the assumption on γ.

□

Lemma 14. *Suppose $\varrho\colon C_0 \to C_1 \to \ldots \to C_k \to \ldots$ is the sequence of configurations in a memoryless run of $\mathcal{B}_\varphi$ on a timed word $(\sigma_i, t_i)_{i\in\mathbb{N}}$. If C_i is active, $j > i$ and $t_j - t_i > M$, then $\mathsf{rank}(C_j) \prec \mathsf{rank}(C_i)$.*

Proof. Write $\mathsf{rank}(C_i) = (\varphi_k, \bot) \ldots (\varphi_2, \bot)(\varphi_1, \top)$ and suppose, for a contradiction, that $\mathsf{rank}(C_i) = \mathsf{rank}(C_{i+1}) = \ldots = \mathsf{rank}(C_j)$. In particular, for $2 \leqslant p \leqslant k$, the node $(\varphi_p, \bot)$ is present in each of the configurations $C_i, C_{i+1}, \ldots, C_j$. This means that in the execution tree underlying ϱ, between depths i and j, any node labelled $(\varphi_p, \bot)$, for $2 \leqslant p \leqslant k$, also has a child labelled $(\varphi_p, \bot)$ (by locality of $\mathcal{B}_\varphi$, no state can make a (discrete) transition to $(\varphi_p, \bot)$ apart from $(\varphi_p, \bot)$ itself). By flatness of $\mathcal{B}_\varphi$ we conclude that the only possible depth-j descendents of a depth-i node labelled $(\varphi_p, \bot)$, $2 \leqslant p \leqslant k$, are also labelled by $(\varphi_p, \bot)$, or by LTL formulas.

Now, since $\mathsf{rank}(C_i) = \mathsf{rank}(C_j)$, $(\varphi_1, \top)$ occurs in $\mathsf{rank}(C_j)$. Thus there is a state $(\varphi_1, v) \in C_j$ such that $v \leqslant M$. From the above argument, the depth-i ancestor of this state can only be labelled (φ_1, u) for some u. Since $t_j - t_i > M$, the clock x is reset somewhere on the path from (φ_1, u) to (φ_1, v). But this contradicts linearity and locality of $\mathcal{B}_\varphi$, since these conditions imply that any clock reset on a path must be accompanied by a strict reduction in the rank of the locations along the path. □

5 From ATAs to CAROTs

In this section, we define a simulation of ATAs by CAROTs. This roughly corresponds to the powerset construction used for transforming an (untimed) alternating automaton into a non-deterministic automaton [17], except that we cannot bound the size of a configuration in the timed case due to the presence of clock variables. Instead we use the channel to store encodings of configurations, which are levels in a run tree of the ATA being simulated. In this simulation, the cycling of the channel corresponds to the evolution of time, and global renaming and occurrence testing are used to simulate discrete transitions of the ATA.

Using Theorem 12, we show that an ATA $\mathcal{B}_\varphi$ corresponding to a $\mathsf{Flat\text{-}MTL}$ formula φ can be simulated by a cycle-bounded CAROT. Then we use Theorem 3, concerning the cycle-bounded reachability problem for CAROTs, to prove an $\mathsf{EXPSPACE}$ upper bound for model checking.

We first fix some notation: let $\mathcal{A} = (L_\mathcal{A}, X_\mathcal{A}, \Sigma, L^0_\mathcal{A}, \delta_\mathcal{A})$ be the timed automaton under study, and $\mathcal{B} = (L_\mathcal{B}, \Sigma, \delta^0_\mathcal{B}, \delta_\mathcal{B}, F_\mathcal{B})$ be the ATA corresponding to $\neg\varphi$ (previously called $\mathcal{B}_{\neg\varphi}$) constructed in the previous section. We call $x_\mathcal{B}$ its single clock.

We note $\mathsf{REG} = \{0, 1, \ldots, M, \bot\}$ where M is the maximal constant appearing in $\mathcal{A}$ or in $\mathcal{B}$. If $\gamma \in \mathbb{R}^+$ and $\gamma \leqslant M$, we write $\mathsf{reg}(\gamma)$ for the largest integer in REG which is smaller than or equal to γ. We write $\mathsf{reg}(\bot) = \bot$. We also define the following two sets:

$$S = (L_\mathcal{B} \times \{x_\mathcal{B}\} \times \mathsf{Val}) \cup (L_\mathcal{A} \times X_\mathcal{A} \times \mathsf{Val})$$
$$R = (L_\mathcal{B} \times \{x_\mathcal{B}\} \times \mathsf{REG}) \cup (L_\mathcal{A} \times X_\mathcal{A} \times \mathsf{REG})$$

and their sets of subsets $V = \wp(S)$ and $\Lambda = \wp(R)$.

A joint $\mathcal{A}/\mathcal{B}$-configuration is composed of a state (ℓ, v) of $\mathcal{A}$ with $\ell \in L_\mathcal{A}$, $v\colon X_\mathcal{A} \to \mathsf{Val}$ and a finite set of states (ℓ_i, v_i) of $\mathcal{B}$, with $\ell_i \in L_\mathcal{B}$ and $v_i \in \mathsf{Val}$ for $i \in I$. Such a configuration C can be written as the element $\{(\ell_i, x_\mathcal{B}, v_i) \mid i \in I\} \cup \{(\ell, x, v(x)) \mid x \in X_\mathcal{A}\}$ of V.

Now given a configuration C, partition C into a sequence of subsets $C_0, C_1, \ldots, C_n, C_\bot$, such that $C_\bot = \{(\ell, x, v) \in C \mid \ell \in \mathsf{LTL} \text{ or } v = \bot\}$, $\bigcup_{i=0}^n C_i = C \setminus C_\bot$, and if $i, j \neq \bot$, for all $(\ell, x, v) \in C_i$ and $(\ell', x', v') \in C_j$, $\mathit{frac}(v) \leqslant \mathit{frac}(v')$ iff $i \leqslant j$ (so that (ℓ, x, v) and (ℓ', x', v') are in the same block C_i iff v and v' have the same fractional part). We assume in addition that the fractional part of elements in C_0 is 0 (even if it means that $C_0 = \varnothing$). Note that $C_\bot$ contains all inactive and LTL formulas of the configuration (following the vocabulary of the previous section).

We then define $H\colon V \to \Lambda^*$ with $H(C) = \mathsf{reg}(C_0) \cdot \mathsf{reg}(C_1) \cdot \mathsf{reg}(C_2) \cdots \mathsf{reg}(C_n) \cdot \mathsf{reg}(C_\bot)$, where $\mathsf{reg}(C)$ is obtained by replacing each value v that appears in C with $\mathsf{reg}(v)$. In the following, we remove the superfluous $x_\mathcal{B}$'s and $\bot$'s in the letters, in order to ease readability.

The joint $\mathcal{A}/\mathcal{B}$-behaviour is then composed of transitions $C \xrightarrow{\sigma} C'$ for $\sigma \in \Sigma$ and $C \xrightarrow{t} C'$ for $t \in \mathbb{R}^+$ in the usual way. Using the abstraction function, it is possible to define a discrete transition system which abstracts away precise timing information, but which simulates joint $\mathcal{A}/\mathcal{B}$-behaviours, see [18, 19].

Example 15. *Consider for instance a configuration C encoded by the word $H(C) = \{(\ell_0, 2), (\ell, x, 3)\} \cdot \{(\ell, y, 1)\} \cdot \{(\ell_1, 3), (\ell_2, 1)\} \cdot \{(\ell, z), \ell_3\}$. We assume that the maximal constant is 4. The encoding of the successor of C is obtained by cycling around the letters (except the last one) of the word (and increasing the values of the regions accordingly). Thus the first delay successor of $H(C)$ is $\varnothing \cdot \{(\ell_0, 2), (\ell, x, 3)\} \cdot \{(\ell, y, 1)\} \cdot \{(\ell_1, 3), (\ell_2, 1)\} \cdot \{(\ell, z), \ell_3\}$ (all states with integral values are now just above the integer), the next successor is $\{(\ell_1, 4), (\ell_2, 2)\} \cdot \{(\ell_0, 2), (\ell, x, 3)\} \cdot \{(\ell, y, 1)\} \cdot \{(\ell, z), \ell_3\}$ (the states with maximal fractional part reach the next integer), the next one* *is* $\varnothing\cdot\{(\ell_2,2)\}\cdot\{(\ell_0,2),(\ell,x,3)\}\cdot\{(\ell,y,1)\}\cdot\{(\ell,z),\ell_3,\ell_1\}$ *as the state* ℓ_1 *is now over the maximal constant* 4*. Simulating discrete transitions is easy as it only consists in applying the transition rules of* $\mathcal{A}$ *and* $\mathcal{B}$ *to all states of the word (see above-mentioned references for more details).*

We will take advantage of this discrete abstraction to define a CAROT $\mathcal{C}$ which will 'recognize' the discrete joint $\mathcal{A}/\mathcal{B}$-behaviours. The channel will be used to store the 'unbounded' part of the information, namely the successive configurations of $\mathcal{B}$. Since $\mathcal{A}$ must synchronize with $\mathcal{B}$, its timing information will also be stored on the channel. The discrete states of the CAROT will store only a bounded amount of information, namely the location of $\mathcal{A}$, the region it lies in, the set of clocks of $\mathcal{A}$ having integer values, and the $\mathcal{B}$-part of the sets $\mathsf{reg}(C_0)$ and $\mathsf{reg}(C_\perp)$. For instance, a configuration C such that

$$H(C)=\{(\ell_1,r_0),(\ell_2,r_4),(\ell,x,r_2)\}\cdot \\ \{(\ell,y,r_1),(\ell_1,r_5),(\ell_2,r_3)\}\cdot\{(\ell,z,r_7)\}\cdot\{\ell_3\}$$

is encoded by the discrete information

$$\Big(\ell,\varrho,\{x\},\{(\ell_1,r_0),(\ell_2,r_4)\},\{\ell_3\}\Big)$$

where $\varrho(x)=r_2$, $\varrho(y)=r_1$ and $\varrho(z)=r_7$, and by the channel content (where we read from the left):

	$\langle$	z	$\rangle$	$\langle$	(ℓ_2,r_3)	(ℓ_1,r_5)	y	$\rangle$	

We construct the CAROT $\mathcal{C}=(Q,q_0,\Gamma,\Delta)$ (without accepting conditions for the moment) as follows:

- The channel alphabet Γ is the union of $L_\mathcal{B}\times\mathsf{REG}\setminus\{\perp\}$, the set of clocks $X_\mathcal{A}$, and the two brackets $\langle$ and $\rangle$.
- The set Q of states is the product set $L_\mathcal{A}\times\mathsf{REG}^{X_\mathcal{A}}\times\wp(X_\mathcal{A})\times\wp(L_\mathcal{B}\times(\mathsf{REG}\setminus\{\perp\}))\times\wp(L_\mathcal{B})$. It stores the current location of $\mathcal{A}$, the integral part of the clocks of $\mathcal{A}$, the set of clocks of $\mathcal{A}$ having integer value, the set of states $(\ell_\mathcal{B},x_\mathcal{B})$ of the current configuration of $\mathcal{B}$ in which the clock $x_\mathcal{B}$ is an integer, and the set of inactive (or LTL) formulas in the current configuration of $\mathcal{B}$.
- The initial states of $\mathcal{C}$ are the states that correspond to an initial state of $\mathcal{A}$ with all clocks being equal to zero, and to sets of states of $\mathcal{B}$ satisfying the initial condition of $\mathcal{B}$.
- There are three kinds of transitions. First, from a state $(\ell_\mathcal{A},r_\mathcal{A},z_\mathcal{A},\lambda_\mathcal{B},\kappa_\mathcal{B})$ where $z_\mathcal{A}$ and/or $\lambda_\mathcal{B}$ are non-empty, applying an (abstract) delay transition corresponds to entering a state where both $z_\mathcal{A}$ and $\lambda_\mathcal{B}$ are empty, and to pushing $z_\mathcal{A}\cup\lambda_\mathcal{B}$ on the channel.

Symmetrically, from a state where both $z_\mathcal{A}$ and $\lambda_\mathcal{B}$ are empty, an abstract delay transition reads the leftmost item of the channel (corresponding to the set of states having the highest fractional parts) and stores it in the discrete state of the CAROT, updating the integral values of the corresponding clocks.

Finally, action transitions consist in simultaneously applying a discrete step of $\mathcal{A}$ and a discrete step of $\mathcal{B}$. Occurrence testing is used to determine (an overapproximation) of the set of locations of $\mathcal{B}$ that lie on the channel, and global deletion is used to remove a letter (ℓ,r) from the channel when the clock corresponding to ℓ is reset to 0.

It is not difficult to prove that the constructed CAROT simulates the joint $\mathcal{A}/\mathcal{B}$-behaviours, but also that any run of the CAROT can be simulated by a joint $\mathcal{A}/\mathcal{B}$-behaviour.

The above construction thus encodes the synchronized behaviour of $\mathcal{A}$ and $\mathcal{B}$, but it remains to encode the acceptance condition of $\mathcal{B}$. This is achieved through the Miyano-Hayashi construction [17, 24]. This requires us to add some extra structure to $\mathcal{C}$ in order to keep track of branches in the execution forest of $\mathcal{B}$ that are still 'waiting' to enter an accepting state.

The most important aspect of the above simulation concerns its relationship to Theorem 12. In particular, in any segment of an execution of $\mathcal{B}$ in which all clock values in the states of $\mathcal{B}$ are inactive, the simulating CAROT $\mathcal{C}$ has at most $|X_\mathcal{A}|$ items on the channel, corresponding to the clocks of $\mathcal{A}$ (recall that $X_\mathcal{A}$ is the set of clocks of $\mathcal{A}$). In such a segment, the current configuration of $\mathcal{B}$ is encoded entirely in the control state of the CAROT. Otherwise, if some of the clocks of $\mathcal{B}$ are active, the simulating CAROT requires one cycle of its channel to simulate one time unit of $\mathcal{B}$'s execution (since the active clocks of $\mathcal{B}$ are stored on the channel in order of their fractional parts). Theorem 12 then yields an upper bound on the number of cycles of $\mathcal{C}$'s channel, as made precise below.

Proposition 16. *Let* $\mathcal{A}$ *be a timed automaton with set of clocks* X*, and* $\varphi\in\mathsf{coFlat\text{-}MTL}$*. Let* $\mathcal{C}_{\mathcal{A},\neg\varphi}$ *denote the CAROT that simulates joint executions of* $\mathcal{A}$ *and* $\mathcal{B}_\varphi$*, as described above. Then,* $\mathcal{A}\models\varphi$ *iff there is an infinite computation* ϱ *of* $\mathcal{C}_{\mathcal{A},\neg\varphi}$ *such that we can write* ϱ *as* $\varrho'\cdot\varrho''$ *where:*
(1) the number of cycles of the channel during ϱ' *is bounded by an exponential in the sizes of* φ *and* $\mathcal{A}$*,*
(2) along ϱ''*, the size of the channel is bounded by* $|X|$*,*
(3) the Büchi condition of $\mathcal{C}_{\mathcal{A},\neg\varphi}$ *is satisfied along* ϱ''*.*

Note that the number of control states of the CAROT $\mathcal{C}_{\mathcal{A},\neg\varphi}$ is doubly exponential in the sizes of $\mathcal{A}$ and φ. Indeed, the states consist of subsets of REG, which is exponential in the size of the encoding of the maximal constant. However, we can apply the algorithm of Theorem 3 on-the-fly, without explicitely building the CAROT $\mathcal{C}_{\mathcal{A},\neg\varphi}$. This algorithm will be applied on the first part ϱ' of ϱ. Since the channel is bounded along ϱ'', the CAROT can be transformed into a Büchi automaton (still doubly exponential) for verifying the second part, which can also be achieved on-the-fly using exponential space. Finally:

Theorem 17. *The model-checking problem for* coFlat-MTL *is EXPSPACE-Complete.*

We refer the reader to [7] for the hardness proof.

6 Conclusion

In this paper, we have proposed the logic coFlat-MTL as a counterpart to MITL, until now considered to be the only linear-time timed temporal logic having reasonable complexity. Although both logics are incomparably expressive, coFlat-MTL allows most specifications that are interesting in practice, whilst retaining punctuality. Moreover, its model-checking problem exhibits no cost over that of MITL. As specifications tend to be relatively small, we feel justified in considering the complexity of model checking coFlat-MTL to be feasible, at least in theory. The real test will consist in applying our results in practice.

References

[1] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

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

[3] R. Alur and T. A. Henzinger. Logics and models of Real-Time: a survey. In *Real-Time: Theory in Practice, Proc. REX Workshop 1991*, vol. 600 of *LNCS*, pp. 74–106. Springer, 1992.

[4] R. Alur and T. A. Henzinger. Real-time logics: Complexity and expressiveness. *Information and Computation*, 104(1):35–77, 1993.

[5] R. Alur and T. A. Henzinger. A really temporal logic. *Journal of the ACM*, 41(1):181–204, 1994.

[6] P. Bouyer, N. Markey, J. Ouaknine, Ph. Schnoebelen, and J. Worrell. On the complexity of termination in faulty channel machines. 2007. Submitted.

[7] P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. The cost of punctuality. Research report LSV-07-05, Lab. Spécification & Vérification, ENS Cachan, France, 2007.

[8] D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, 1983.

[9] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *Proc. 32nd Annual Symp. on Found. of Computer Science (FOCS'91)*, pp. 368–377. IEEE Comp. Soc. Press, 1991.

[10] T. A. Henzinger. *The Temporal Specification and Verification of Real-Time Systems*. PhD thesis, Stanford University, CA, USA, 1991.

[11] T. A. Henzinger. It's about time: Real-time logics reviewed. In *Proc. 9th Int. Conf. on Concurr. Theory (CONCUR'98)*, vol. 1466 of *LNCS*, pp. 439–454. Springer, 1998.

[12] T. A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *Proc. 19th Int. Coll. on Automata, Languages and Programming (ICALP'92)*, vol. 623 of *LNCS*, pp. 545–558. Springer, 1992.

[13] Y. Hirshfeld and A. M. Rabinovich. Logics for real time: Decidability and complexity. *Fundamenta Informaticae*, 62(1):1–28, 2004.

[14] R. Koymans. Specifying real-time properties with Metric Temporal Logic. *Real-Time Systems*, 2(4):255–299, 1990.

[15] S. Lasota and I. Walukiewicz. Alternating timed automata. In *Proc. 8th Int. Conf. on Found. of Software Science and Computation Structures (FoSSaCS'05)*, vol. 3441 of *LNCS*, pp. 250–265. Springer, 2005.

[16] O. Maler, D. Nickovic, and A. Pnueli. From MITL to timed automata. In *Proc. 4th Int. Conf. on Formal Modeling and Analysis of Timed Systems (FORMATS'06)*, vol. 4202 of *LNCS*, pp. 274–189. Springer, 2006.

[17] S. Miyano and T. Hayashi. Alternating finite automata on omega-words. *Theoretical Computer Science*, 32:321–330, 1984.

[18] J. Ouaknine and J. Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *Proc. 19th Annual Symp. on Logic in Computer Science (LICS'04)*, pp. 54–63. IEEE Comp. Soc. Press, 2004.

[19] J. Ouaknine and J. Worrell. On the decidability of Metric Temporal Logic. In *Proc. 19th Annual Symp. on Logic in Computer Science (LICS'05)*, pp. 188–197. IEEE Comp. Soc. Press, 2005.

[20] J. Ouaknine and J. Worrell. On Metric Temporal Logic and faulty Turing machines. In *Proc. 9th Int. Conf. on Found. of Software Science and Computation Structures (FoSSaCS'06)*, vol. 3921 of *LNCS*, pp. 217–230. Springer, 2006.

[21] J. Ouaknine and J. Worrell. Safety Metric Temporal Logic is fully decidable. In *Proc. 12th Int. Conf. on Tools and Algorithms for the Constr. and Analysis of Systems (TACAS'06)*, vol. 3920 of *LNCS*, pp. 411–425. Springer, 2006.

[22] J. Ouaknine and J. Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Computer Science*, 3(1), 2007.

[23] J.-F. Raskin. *Logics, Automata and Classical Theories for Deciding Real Time*. PhD thesis, Université de Namur, Belgium, 1999.

[24] M. Y. Vardi. An automata-theoretic approach to Linear Temporal Logic. In *Proc. Logics for Concurr.: Structure versus Automata*, vol. 1043 of *LNCS*, pp. 238–266. Springer, 1996.

[25] Th. Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In *Proc. 3rd Int. Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'94)*, vol. 863 of *LNCS*, pp. 694–715. Springer, 1994.