# On Process-Algebraic Extensions of Metric Temporal Logic

Christoph Haase, Joël Ouaknine, and James Worrell

**Abstract** It is known that the satisfiability problem for Metric Temporal Logic (MTL) is decidable over finite timed words. In this paper we study the satisfiability problem for extensions of this logic by various process-algebraic operators. On the negative side we show that satisfiability becomes undecidable when any of hiding, renaming, or asynchronous parallel composition are added to the logic. On the positive side we show decidability with the addition of alphabetised parallel composition and fixpoint operators. We use one-clock Timed Propositional Temporal Logic (TPTL(1)) as a technical tool for the decidability results and show that TPTL(1) with fixpoints provides a logical characterisation of the class of languages accepted by one-clock timed alternating automata.

## 1 Introduction

The model of time usually adopted in computer-aided verification and process algebra is *qualitative*: it offers an ordering of the various events a given system may go through, but abstracts away from *quantitative*, or *metric*, information regarding the precise timing of these events. If such information is required, one must adopt a more sophisticated framework, modelling time using real numbers for example. Over the last two decades, much work has gone into developing and studying such frameworks, both in the model-checking and in the process-algebraic communities.

This paper studies extensions of the linear dense-time specification formalism *Metric Temporal Logic (MTL)*. MTL, introduced by Koymans in 1990 [13], is one

Christoph Haase
Oxford University Computing Laboratory, UK, e-mail: `chrh@comlab.ox.ac.uk`

Joël Ouaknine
Oxford University Computing Laboratory, UK, e-mail: `joel@comlab.ox.ac.uk`

James Worrell
Oxford University Computing Laboratory, UK, e-mail: `jbw@comlab.ox.ac.uk`

of the most prominent logics for reasoning about real-time systems. MTL formulas can either be interpreted in a *state-based* semantics, in which observations are made continuously, or in an *event-based* semantics, in which observations are recorded as instantaneous ‘snapshots’ whenever a discrete change, or ‘event’, occurs. In the latter, models of formulas are *timed words*, i.e., sequences of events together with associated real-valued timestamps.

Unfortunately, it has long been known that MTL satisfiability is undecidable in the state-based semantics [2, 9]. Moreover, it was shown more recently that over infinite timed words, MTL is also undecidable [19]. Surprisingly, MTL turned out to be decidable—albeit with non-primitive recursive complexity—over finite timed words [20]. Subsequent to this discovery, various fragments of MTL—over both semantics and over both finite and infinite behaviours—were shown to be decidable; for a recent survey of these results, we refer the reader to [5, 21].

This paper focuses on extensions of MTL by various natural process-algebraic operators, from the point of view of computability. Accordingly, we are exclusively interested in the event-based semantics over finite timed words, as all other semantics immediately result in undecidability. We consider MTL augmented with the following various operators:

- *Hiding.* This operator, which corresponds to existential quantification, provides a convenient way to abstract away unimportant events (as regards a particular property of interest).
- *Renaming.* Similarly to hiding, the renaming operator is useful for expressing specifications and constructing abstractions of systems; it can be used, for example, to group the various possible events into a small number of categories.
- *Asynchronous parallel composition.* Also known as *interleaving* or *shuffle product*, this operator combines the behaviours of two systems in as liberal a way as possible; in particular, each system is entirely oblivious to the other one.
- *Alphabetised parallel composition.* Also known as *(partially) synchronous parallel composition*. Two systems thus composed will synchronise over their common events, and otherwise proceed independently of each other. This operator is particularly useful to model communication over a well-defined interface.
- *Fixpoints.* Fixpoint operators are omnipresent in process algebra and model checking, enhancing the expressiveness of various formalisms and allowing one, for example, to model recursion.

The results of this paper are two-fold. On the negative side, we show that MTL augmented with any of hiding, renaming, or asynchronous parallel composition becomes undecidable. The main result, however, is that we can augment MTL with both alphabetised parallel composition and least fixpoint operators and still retain decidability over finite words. The key technical tools we use to obtain decidability are the one-clock (or one-variable) fragment of Timed Propositional Temporal Logic, denoted TPTL(1) [3], and one-clock Timed Alternating Automata (1TAA) [15, 20]. Moreoever, we show that the extension of TPTL(1) with fixpoints provides a complete logical characterisation of 1TAA, which is of independent interest.

The process-algebraic operators listed above originate from Tony Hoare's Communicating Sequential Processes (CSP), undoubtedly the most prominent linear-time process algebra. These operators, or slight variations thereof, have also figured in other process algebras and in the context of temporal and dynamic logics. For example, Lange [14] considers LTL with fixpoint operators, showing that it is expressively equivalent to finite alternating automata with weak parity acceptance conditions. Hiding also appears in temporal logic in the guise of existential quantification over propositional variables. Sistla, Vardi, and Wolper [26] show that LTL with existential quantification can express all $\omega$-regular languages. Over real time, it is known that Metric Interval Temporal Logic (MITL) with existential quantification can express all languages that are accepted by timed automata [11]. Propositional Dynamic Logic with interleaving has been considered in [16].

One of the key contributions of Tony Hoare's work on CSP has been a deeper understanding of the central phenomenon of *nondeterminism* in semantics. Hoare's classic text *Communicating Sequential Processes* [12], for example, devotes an entire chapter to the subject; his perspective on nondeterminism, in particular as a mechanism of underspecification, but also as an inevitable consequence of concurrency, has proven enormously influential.

From a semantic standpoint, it seems fair to say that the development of the standard failures divergences model for CSP [6] arose principally as a solution to the problem of adequately handling nondeterminism in a denotational setting. The problems turned out considerably more resilient in the timed world, and a fully satisfactory understanding of nondeterminism in Timed CSP has not yet been reached [25]. Nonetheless, one of the pivotal notions to emerge from the study of nondeterminism in both the untimed and timed settings is that of *operators that preserve determinism*. It is remarkable—although perhaps not entirely surprising—that in the present paper, the operators that preserve decidability turn out to be precisely those that preserve determinism (quite independently of the fact that basic MTL formulas do exhibit native 'nondeterminism' through disjunction in any case).

Nondeterminism was also studied around the same time as Tony Hoare by Robin Milner, and features in his seminal work *A Calculus of Communicating Systems* [17]. Milner was however exclusively concerned with operational semantics at the time, and consequently his outlook had a very different flavour. Outside of process algebra and semantics, nondeterminism has an even older history, going back (at least) some two millennia in philosophy, and half a century in other areas of computer science [22], notably formal language theory, algorithms, and complexity. Modern applications of nondeterminism can be found, among others, in computer security, artificial intelligence, and software engineering.

Most proofs have been omitted from this paper and can be found in the technical report [8].

## 2 Preliminaries

Let $\mathbb{R}_+$ denote the set of non-negative real numbers, $\mathbb{Q}_+$ the set of non-negative rational numbers, and $\mathbb{N}$ the set of positive integers. The set $\mathbb{N}_n$ is the set of positive integers up to and including $n$, i.e., $\mathbb{N}_n := \{1,\ldots,n\}$. For an interval $\mathscr{I} \subseteq \mathbb{R}_+$ and $r \in \mathbb{R}_+$, $\mathscr{I} + r := \{u + r \mid u \in \mathscr{I}\}$. By $Id_X := \{(x,x) \mid x \in X\}$ we denote the identity relation on a set $X$. Given a binary relation $R \subseteq X \times Y$, we define its functional lifting $R : X \to \mathscr{P}(Y)$ as $R(x) := \{y \mid (x,y) \in R\}$. We call $R$ *total* if $R(x) \neq \emptyset$ for all $x \in X$. Given a function $f : X \to Y$, its update $f[x \mapsto y] : X \to Y$ is defined as $f[x \mapsto y](z) := y$ if $z = x$ and $f[x \mapsto y](z) := f(z)$ otherwise.

In the untimed world, traces of systems are usually modelled as finite or infinite words over some alphabet of events $\Sigma$. However, as discussed in the Introduction, this model does not allow one to make quantitative assertions regarding *when* events occur. A natural way to overcome this drawback, first proposed by Reed and Roscoe in the development of Timed CSP [23, 24], is to model traces of timed systems as finite or infinite words over the event alphabet together with timestamps indicating the time of occurrence of events. In the remainder of this paper we focus exclusively on finite timed words.

**Definition 1 (Timed words).** Let $\Sigma$ be a nonempty finite set of events. A **time sequence** $\tau$ is a finite sequence $\tau_1\tau_2\ldots\tau_n$ of time values from $\mathbb{R}_+$ such that $\tau_i \leq \tau_{i+1}$ for all $1 \leq i < n$. A **timed word** $\rho$ over $\Sigma$ is a tuple $(\sigma,\tau)$ where $\tau$ is a time sequence and $\sigma = \sigma_1\sigma_2\ldots\sigma_n$ is a word over $\Sigma$ of the same length as $\tau$.

The set of all finite timed words over $\Sigma$ is written $T\Sigma^*$. Note that our notion of time is *weakly monotonic*, in that we allow several events to share the same timestamp. Similar results to the ones presented here also hold for *strongly monotonic* time, although as pointed out in [10], awkward complications arise when disallowing the possibility of simultaneous events in the presence of parallel composition operators. Note that we do not require the first element of a time sequence to be zero.

The *length* of a timed word $\rho$ is denoted by $|\rho|$ and is the length of the underlying time sequence. Alternatively, we can represent a timed word as a sequence of *timed events* by writing $\rho = (\sigma_1,\tau_1)(\sigma_2,\tau_2)\ldots(\sigma_n,\tau_n)$. For convenience, we also define auxiliary functions as follows: for $1 \leq i \leq |\rho|$, $\sigma_i(\rho) := \sigma_i$ and $\tau_i(\rho) := \tau_i$, where $(\sigma_i,\tau_i)$ is the $i$-th timed event of $\rho$. Given a timed word $\rho$, denote by $\rho^{i,j}$ the timed word $(\sigma_i,0)(\sigma_{i+1},\tau_{i+1}-\tau_i)\ldots(\sigma_j,\tau_j-\tau_i), 1 \leq i \leq j \leq |\rho|$. Moreover $\rho^i := \rho^{i,|\rho|}$ and for $j > |\rho|$, $\rho^{i,j} := \rho^i$. Given $E \subseteq \Sigma$, the timed word $\rho \setminus E$ is obtained from $\rho$ by deleting all timed events $(\sigma_i,\tau_i)$ from $\rho$ with $\sigma_i \in E$.

**Definition 2 (TPTL(1) syntax).** TPTL(1) formulas are defined inductively according to the following grammar:

$$\varphi ::= a \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \,\mathscr{U}\, \varphi_2 \mid x \sim c \mid x.\varphi$$

Here, $a \in \Sigma$ is an event, $\bigcirc$ is the **next** operator, $\mathscr{U}$ is the **until** operator, $x$ is a **clock variable**, $c \in \mathbb{Q}_+$ and $\sim \in \{\leq, <, =, \neq, >, \geq\}$. Note that TPTL(1) makes use of a single clock variable, $x$.

We define the standard Boolean abbreviations $\varphi_1 \wedge \varphi_2 := \neg(\neg\varphi_1 \vee \neg\varphi_2)$, $\varphi_1 \rightarrow \varphi_2 := \neg\varphi_1 \vee \varphi_2$, $\top := a \vee \neg a$, and $\bot := \neg\top$. The *eventually* operator is defined as $\Diamond\varphi := \top \mathscr{U} \varphi$ and the *globally* operator as $\Box\varphi := \neg\Diamond\neg\varphi$. The clock variable in TPTL(1) formulas is the key reference for making quantitative statements about the evolution of time. It allows one to 'freeze' (or record) time points along a timed word, which can later be compared to the current time. When $x.\varphi$ holds at some time point $\tau$, $x$ is bound to $\tau$ in $\varphi$ and when the clock constraint $x \sim c$ is evaluated at some later time point $\tau'$, it is checked whether or not $\tau' - \tau \sim c$. This can be seen this as *resetting* the clock $x$ at time point $\tau$.

Originally, TPTL as introduced in [3] allowed for multiple clock variables. However, that logic has an undecidable satisfiability problem and we therefore only consider its one-variable fragment TPTL(1) in this paper.

We now give a non-standard presentation of the semantics of TPTL(1), which can however easily be shown to be equivalent to that commonly found in the literature. Its main advantage is to ease the definition of fixpoint operators later on.

Given a timed word $\rho$ and a TPTL(1) formula $\varphi$, the semantic function $[\![-]\!]^\rho$ maps $\varphi$ to an element of the set $\mathscr{V}(\rho) := \mathscr{P}(\mathbb{N}_{|\rho|} \times \mathbb{R}_+)$. Intuitively, $(i,r) \in [\![\varphi]\!]^\rho$ if $\varphi$ holds at position $i$ in $\rho$ when the value of the clock variable $x$ is $r$.

**Definition 3 (TPTL(1) semantics).** The semantics of a TPTL(1) formula $\varphi$ is defined by induction on the structure of $\varphi$, as follows:

$$\begin{array}{ll}
[\![a]\!]^\rho & := \{(i,r) \mid \sigma^i = a, i \in \mathbb{N}_{|\rho|}, \text{ and } r \in \mathbb{R}_+\} \\
[\![\varphi_1 \vee \varphi_2]\!]^\rho & := [\![\varphi_1]\!]^\rho \cup [\![\varphi_2]\!]^\rho \\
[\![\neg\varphi]\!]^\rho & := \{(i,r) \mid i \in \mathbb{N}_{|\rho|} \text{ and } r \in \mathbb{R}_+\} \setminus [\![\varphi]\!]^\rho \\
[\![\bigcirc\varphi]\!]^\rho & := \{(i,r) \mid (i+1,r') \in [\![\varphi]\!]^\rho \text{ and } r = r' + \tau_i - \tau_{i+1}\} \\
[\![\varphi_1 \mathscr{U} \varphi_2]\!]^\rho & := \{(i,r) \mid \exists j. i \leq j \leq |\rho| \text{ and } (j, r+\tau_j - \tau_i) \in [\![\varphi_2]\!]^\rho \text{ and} \\
& \qquad \forall k. i \leq k < j \text{ implies } (k, r+\tau_k - \tau_i) \in [\![\varphi_1]\!]^\rho\} \\
[\![x \sim c]\!]^\rho & := \{(i,r) \mid i \in \mathbb{N}_{|\rho|}, r \in \mathbb{R}_+, \text{ and } r \sim c\} \\
[\![x.\varphi]\!]^\rho & := \{(i,r) \mid (i,0) \in [\![\varphi]\!]^\rho \text{ and } r \in \mathbb{R}_+\}
\end{array}$$

We write $\rho \models \varphi$ iff $(1, \tau_1(\rho)) \in [\![\varphi]\!]^\rho$, and $L(\varphi) := \{\rho \mid \rho \models \varphi\}$ for the timed language defined by $\varphi$. A TPTL(1) formula $\varphi$ is called *satisfiable* iff $L(\varphi) \neq \emptyset$. The problem of checking whether a formula $\varphi$ is satisfiable has been shown to be decidable with non-primitive recursive complexity in [20], by translating TPTL(1) formulas into one-clock timed alternating automata (1TAA), introduced subsequently.[1]

The real-time logic MTL can be defined as a syntactic fragment of TPTL(1). It is known to be strictly less expressive than TPTL(1) [4].

**Definition 4 (MTL).** MTL formulas are defined according to the following grammar, where $a \in \Sigma$ and $\mathscr{I}$ is an open, half-open, or closed interval with endpoints in $\mathbb{Q}_+$:

[1] Technically speaking, [20] deals with Metric Temporal Logic rather than TPTL(1). The proof techniques however carry over straightforwardly.

$$\varphi ::= a \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi \mid \bigcirc_{\mathscr{I}} \varphi \mid \varphi_1 \,\mathscr{U}_{\mathscr{I}}\, \varphi_2$$

The semantics of MTL formulas is given by a translation function $(-)^\dagger$ that maps MTL formulas to TPTL(1) formulas, as follows:

$$\begin{aligned} a^\dagger &:= a & (\bigcirc_{\mathscr{I}} \varphi)^\dagger &:= x.\bigcirc(x \in \mathscr{I} \wedge \varphi^\dagger) \\ (\varphi_1 \vee \varphi_2)^\dagger &:= \varphi_1^\dagger \vee \varphi_2^\dagger & (\varphi_1 \,\mathscr{U}_{\mathscr{I}}\, \varphi_2)^\dagger &:= x.(\varphi_1^\dagger \,\mathscr{U}\, (x \in \mathscr{I} \wedge \varphi_2^\dagger)) \\ (\neg\varphi)^\dagger &:= \neg(\varphi^\dagger) \end{aligned}$$

where $x \in \mathscr{I}$ denotes the obvious corresponding conjunction of inequalities.

We call $\bigcirc_{\mathscr{I}}$ the *time-constrained next* and $\mathscr{U}_{\mathscr{I}}$ the *time-constrained until* operators. The *time-constrained eventually* operator $\Diamond_{\mathscr{I}}$ and *globally* operator $\Box_{\mathscr{I}}$ are defined similarly to their TPTL(1) counterparts. We also sometimes abuse notation and use pseudo-arithmetic expressions, such as '$=1$', to denote intervals.

Let $S$ be a finite set of *locations*, and define the set $\Phi(S)$ of formulas as follows:

$$\varphi ::= \mathbf{tt} \mid \mathbf{ff} \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid s \mid x \sim c \mid x.\varphi$$

where $s \in S$, $c \in \mathbb{Q}_+$ and $\sim \in \{<, \leq, =, \neq, \geq, >\}$. As in TPTL(1), $x \sim c$ is a clock constraint and the expression $x.\varphi$ resets the clock variable $x$, i.e., binds $x$ to 0 in $\varphi$.

**Definition 5 (1TAA).** A **one-clock timed alternating automaton (1TAA)** is a five-tuple $\mathscr{A} = (\Sigma, S, s_0, F, \delta)$ where $\Sigma$ is a **finite alphabet**, $S$ is a finite set of **locations**, $s_0$ is the **initial location**, $F \subseteq S$ is a finite set of **accepting locations** and $\delta : S \times \Sigma \to \Phi(S)$ is the **transition function**.

Given a 1TAA $\mathscr{A}$, a *state* of $\mathscr{A}$ is a tuple $(s, v)$, where $s$ is a location and $v \in \mathbb{Q}_+$ a clock value. A *configuration* $C$ of $\mathscr{A}$ is a finite set of states, and $\{(s_0, 0)\}$ is the *initial configuration* of $\mathscr{A}$. By $C + r$ we denote the configuration $\{(s, v + r) \mid (s, v) \in C\}$. We call a configuration $C$ *accepting* if $s \in F$ for every location $s$ occurring in $C$. For convenience, given a 1TAA $\mathscr{A}_i = (\Sigma_i, S_i, s_0^i, F_i, \delta_i)$ we introduce functions for accessing each of the components of $\mathscr{A}_i$, e.g., $S(\mathscr{A}_i) = S_i$, $s_0(\mathscr{A}_i) = s_0^i$, etc.

Given a configuration $C$ and a clock value $v$, we define a Boolean valuation on $\Phi(S)$ as follows:

$$\begin{array}{lll} C \models_v \mathbf{tt} & & \\ C \models_v \varphi_1 \wedge \varphi_2 & \text{iff} & C \models_v \varphi_1 \text{ and } C \models_v \varphi_2 \\ C \models_v \varphi_1 \vee \varphi_2 & \text{iff} & C \models_v \varphi_1 \text{ or } C \models_v \varphi_2 \\ C \models_v s & \text{iff} & (s, v) \in C \\ C \models_v x \sim c & \text{iff} & v \sim c \\ C \models_v x.\varphi & \text{iff} & C \models_0 \varphi \end{array}$$

**Definition 6 (Run).** Given a finite timed word $\rho$ of length $n$, define $d_j := \tau_j - \tau_{j-1}$ for $1 \leq j \leq n$ with $\tau_0 := 0$. A **run** of a 1TAA $\mathscr{A}$ on $\rho$ is a finite sequence of configurations

$$C_0 \overset{d_1}{\rightsquigarrow} C_1 \xrightarrow{\sigma_1} C_2 \overset{d_2}{\rightsquigarrow} C_3 \xrightarrow{\sigma_2} \cdots \overset{d_n}{\rightsquigarrow} C_{2n-1} \xrightarrow{\sigma_n} C_{2n}$$

such that $C_{2j+1} = C_{2j} + d_{j+1}$ and for $C_{2j+1} = \{(s_i, v_i)\}_{i \in I}$, $C_{2j+2} = \bigcup_{i \in I} C'_i$, where $C'_i \models_{v_i} \delta(s_i, \sigma_{j+1})$ with $0 \leq j < n$. Here, $C_{2j} \overset{d_{j+1}}{\rightsquigarrow} C_{2j+1}$ is called a delay step and $C_{2j+1} \xrightarrow{\sigma_{j+1}} C_{2j+2}$ is a discrete step. A run is **accepting** if $C_{2n}$ is accepting.

A finite timed word $\rho$ is accepted by a 1TAA $\mathscr{A}$ with respect to an initial clock value $v$ if $\mathscr{A}$ has an accepting run starting from $C_0 = \{(s_0, v)\}$. The language accepted by $\mathscr{A}$, $L(\mathscr{A}) \subseteq T\Sigma^*$, is the set of all finite timed words accepted by $\mathscr{A}$ with respect to the initial clock value zero.

## 3 Decidable Cases

In this section we establish the decidability of satisfiability for TPTL(1) augmented with least fixpoint and alphabetised parallel operators. Our strategy is to translate a formula $\varphi$ in the extension under consideration to a 1TAA $\mathscr{A}_\varphi$ such that $L(\varphi) = L(\mathscr{A}_\varphi)$.

### *Least Fixpoints*

Introducing the least fixpoint operator offers a natural way to express recursive specifications in TPTL(1). The resulting logic $\mu$TPTL(1) is strictly more expressive than TPTL(1).

In order to guarantee the existence of fixpoints, we restrict $\mu$TPTL(1) formulas to be in negation normal form, i.e., with negations only occurring in front of events from $\Sigma$. We moreover drop the until operator, since it can be expressed with the least fixpoint operator.

**Definition 7 ($\mu$TPTL(1) syntax).** The set of $\mu$TPTL(1) formulas is defined inductively according to the following grammar:

$$\varphi ::= \top \mid \bot \mid \dashv \mid a \mid \neg a \mid Z \mid x \sim c \mid x.\varphi \mid \bigcirc\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \mu Z.\varphi$$

Here, $Z$ is a *propositional variable* from a finite set $\mathscr{Z}$, $\mu Z$ is the *least fixpoint* operator, and $\dashv$ is an end-marker that is only true at the last position of a timed word, i.e., is equivalent to $\neg\bigcirc\top$. A $\mu$TPTL(1) formula $\varphi$ is *closed* if every $Z$ in $\varphi$ occurs within the scope of a least fixpoint operator $\mu Z$. Otherwise, the formula is deemed to be *open* and we may write $\varphi(Z_1, \ldots, Z_k)$ to indicate that $Z_1, \ldots, Z_k$ occur unbound in $\varphi$. If $Z \in \mathscr{Z}$ is bound in $\varphi$, we require without loss of generality that there be exactly one least fixpoint quantifier $\mu Z$ occurring in $\varphi$. By $fpd(\varphi)$ we denote the *fixpoint depth* of $\varphi$, which is the maximum nesting depth of least fixpoint operators, e.g., $fpd(\mu Y.(\bigcirc(a \vee Y) \vee \mu Z.(b \vee \bigcirc(Y \wedge Z)))) = 2$. Note that the until operator $\varphi_1 \mathscr{U} \varphi_2$ can be introduced as an abbreviation for $\mu Z.(\varphi_2 \vee (\varphi_1 \wedge \bigcirc Z))$.

The semantics of $\mu$TPTL(1) formulas is given with respect to an *environment* $\xi$, which enables one to evaluate open $\mu$TPTL(1) formulas. Given a timed word $\rho$, $\xi$ is a mapping from the propositional variables in $\mathscr{Z}$ to $\mathscr{V}(\rho)$.

The clauses of Definition 3, which prescribe the semantics of TPTL(1) formulas, carry over to $\mu$TPTL(1) formulas whose outermost connective is in TPTL(1). The additional clauses specific to $\mu$TPTL(1) are as follows:

$$\begin{aligned}
[\![\dashv]\!]^{\rho}_{\xi} &:= \{(|\rho|, r) \mid r \in \mathbb{R}_+\}\\
[\![Z]\!]^{\rho}_{\xi} &:= \xi(Z)\\
[\![\mu Z.\psi(Z)]\!]^{\rho}_{\xi} &:= \bigcap\{M \in \mathscr{V}(\rho) \mid [\![\psi(Z)]\!]^{\rho}_{\xi[Z\mapsto M]} \subseteq M\}
\end{aligned}$$

Thus, $[\![\mu Z.\psi(Z)]\!]^{\rho}_{\xi}$ is the least fixpoint of the function $F_{\psi,Z,\rho,\xi}(M) := [\![\psi]\!]^{\rho}_{\xi[Z\mapsto M]}$.

Before we show the decidability of $\mu$TPTL(1) by translation to 1TAA, we give an example of the usefulness of this extension of TPTL(1).

*Example 1.* The formula *even*$(\varphi)$ expresses the property that $\varphi$ is true on a timed word an even number of times. The untimed language of $L(even(\varphi))$ is not counter-free and not expressible in TPTL(1).

$$even(\varphi) = \mu Y.((\neg\varphi \wedge (\dashv \vee \bigcirc Y)) \vee (\varphi \wedge \bigcirc \mu Z.((\neg\varphi \wedge \bigcirc Z) \vee (\varphi \wedge (\dashv \vee \bigcirc Y)))))$$

(Of course, one would need to put $\neg\varphi$ in negation normal form, which can readily be done as soon as a concrete $\varphi$ is supplied.)

The existence of least fixpoints is a consequence of the subsequent lemma and the Knaster-Tarski fixpoint theorem.

**Lemma 1.** *For any timed word $\rho$, $\mu$TPTL(1) formula $\varphi(Z, Z_1, \ldots, Z_k)$, and valuation of the propositional variables $\xi$, the function $F_{\varphi,Z,\rho,\xi}$ is monotone with respect to $\subseteq$.*

Let $\varphi[Z/\psi]$ be the $\mu$TPTL(1) formula obtained from $\varphi$ in which every occurrence of $Z$ in $\varphi$ is replaced by $\psi$. Approximants of a formula $\mu Z.\psi(Z)$ are inductively defined for any $i \in \mathbb{N}$ as:

$$\begin{aligned}
\mu^0 Z.\psi(Z) &:= \bot\\
\mu^{i+1} Z.\psi(Z) &:= \psi[Z/\mu^i Z.\psi(Z)]
\end{aligned}$$

The next lemma is a standard result about approximants:

**Lemma 2.** *For any timed word $\rho$ and $\mu$TPTL(1) formula $\varphi = \mu Z.\psi(Z)$, $M = [\![\varphi]\!]^{\rho}_{\xi}$ iff there exists an $i \in \mathbb{N}$ such that $M = [\![\mu^i Z.\psi(Z)]\!]^{\rho}_{\xi}$.*

Given a $\mu$TPTL(1) formula $\varphi(Z, Z_1, \ldots, Z_k)$, $Z$ is *guarded* in $\varphi$ if it occurs in the scope of a next operator. We call a formula $\varphi$ *proper* if for every subformula $\mu Z.\psi(Z)$ in $\varphi$, $Z$ is guarded in $\psi(Z)$. Properness of $\mu$TPTL(1) formulas will be assumed in the following without loss of generality, since $\mu Z.(Z \vee \psi(Z))$ is equivalent to $\mu Z.\psi(Z)$ and $\mu Z.(Z \wedge \psi(Z))$ is equivalent to $\bot$. Since we are dealing with finite timed words the fixpoint of $F_{\varphi,Z,\rho,\xi}$ is unique for proper formulas. It therefore follows that least and greatest fixpoints coincide for $\mu$TPTL(1), obviating the need for two distinct fixpoint operators.

**Lemma 3.** *Let $\rho$ be a timed word and $\varphi(Z, Z_1, \ldots, Z_k)$ be a formula such that $Z$ occurs guarded in $\varphi(Z)$. Then for all $\xi, M^*, N^*$, $F_{\varphi,Z,\rho,\xi}(M^*) = M^*$ and $F_{\varphi,Z,\rho,\xi}(N^*) = N^*$ implies $M^* = N^*$.*

Although we have not explicitly allowed for arbitrary negation, $\mu$TPTL(1) still is closed under complement. Given a formula $\varphi$, we define its complement $\overline{\varphi}$ by induction on the structure of $\varphi$, where, $\overline{\sim}$ maps the relation $\sim$ to its complementary relation, e.g., $<$ to $\geq$, $=$ to $\neq$ etc.

$$\begin{array}{llll}
\overline{\top} := \bot & \overline{a} := \neg a & \overline{x \sim c} := x \overline{\sim} c & \overline{\varphi_1 \vee \varphi_2} := \overline{\varphi_1} \wedge \overline{\varphi_2} \\
\overline{\bot} := \top & \overline{\neg a} := a & \overline{\bigcirc \varphi} := \dashv \vee \bigcirc \overline{\varphi} & \overline{\mu Z.\varphi} := \mu Z.\overline{\varphi} \\
\overline{\dashv} := \bigcirc \top & \overline{Z} := Z & \overline{\varphi_1 \wedge \varphi_2} := \overline{\varphi_1} \vee \overline{\varphi_2} &
\end{array}$$

**Lemma 4.** *Let $\varphi$ be a proper $\mu$TPTL(1) formula. Then $(i,r) \in [\![\varphi]\!]^{\rho}_{\xi}$ iff $(i,r) \notin [\![\overline{\varphi}]\!]^{\rho}_{\overline{\xi}}$, where $\overline{\xi}(Z) := \{(i,r) \mid i \in \mathbb{N}_{|\rho|}$ and $r \in \mathbb{R}_+\} \setminus \xi(Z)$.*

The lemma can be proved straightforwardly by induction on the structure of $\varphi$ using the properness of the subformulas $\mu Z.\psi(Z)$ of $\varphi$ and the resulting unique fixpoint property.

The translation of a $\mu$TPTL(1) formula $\varphi$ into a 1TAA $\mathscr{A}_\varphi$ is given by induction on $fpd(\varphi)$ and is somewhat similar to the untimed case considered in [14]. Recall that $Z$ is assumed to be guarded in $\psi(Z)$ for any subformula $\mu Z.\psi(Z)$ of $\varphi$. For $fpd(\varphi) = 0$, we define $\mathscr{A}_\varphi$ by induction on the structure of $\varphi$:

- *Case* $\varphi = a$. Define $\mathscr{A}_\varphi = (\Sigma, \{s_\varphi\}, s_\varphi, \emptyset, \delta)$ with $\delta(s_\varphi, a) = \mathbf{tt}$ and $\delta(s_\varphi, b) = \mathbf{ff}$ if $b \neq a$.
- *Case* $\varphi = \dashv$. Define $\mathscr{A}_\varphi = (\Sigma, \{s_\varphi, s_\dashv\}, s_\varphi, \{s_\dashv\}, \delta)$ with $\delta(s_\varphi, a) = s_\dashv$ and $\delta(s_\dashv, a) = \mathbf{ff}$ for all $a \in \Sigma$.
- *Case* $\varphi = Z$. Define $\mathscr{A}_\varphi = (\Sigma, \{s_\varphi\}, s_\varphi, \emptyset, \delta)$ with $\delta(s_\varphi, a) = \mathbf{tt}$ for all $a \in \Sigma$. Note that we will refer to $s_\varphi$ as $s_Z$ in the induction step $\varphi = \mu Z.\psi(Z)$.
- *Case* $\varphi = x \sim c$. Define $\mathscr{A}_\varphi = (\Sigma, \{s_\varphi\}, s_\varphi, \emptyset, \delta)$ with $\delta(s_\varphi, a) = x \sim c$ for all $a \in \Sigma$.
- *Case* $\varphi = x.\psi$. Define $\mathscr{A}_\varphi = (\Sigma, \{s_\varphi\} \cup S(\mathscr{A}_\psi), s_\varphi, F(\mathscr{A}_\psi), \delta)$ with $\delta(s_\varphi, a) = x.\delta(\mathscr{A}_\psi)(s_0(\mathscr{A}_\psi), a)$ and $a \in \Sigma$ and $\delta(s, a) = \delta(A_\psi)(s, a)$ for all $s \in S(A_\psi)$.
- *Case* $\varphi = \psi_1 \wedge \psi_2$. Define $\mathscr{A}_\varphi = (\Sigma, \{s_\varphi\} \cup S(\mathscr{A}_{\psi_1}) \cup S(\mathscr{A}_{\psi_2}), s_\varphi, F(\mathscr{A}_{\psi_1}) \cup F(\mathscr{A}_{\psi_2}), \delta)$ with $\delta(s_\varphi, a) = \delta(\mathscr{A}_{\psi_1})(s_0(\mathscr{A}_{\psi_1}), a) \wedge \delta(\mathscr{A}_{\psi_2})(s_0(\mathscr{A}_{\psi_2}), a)$ and $\delta(s, a) = \delta(\mathscr{A}_{\psi_i}, a)$ if $s \in S(\mathscr{A}_{\psi_i})$ for all $a \in \Sigma$.
- *Case* $\varphi = \bigcirc\psi$. Define $\mathscr{A}_\varphi = (\Sigma, \{s_\varphi\} \cup S(\mathscr{A}_\psi), s_\varphi, F(\mathscr{A}_\psi), \delta)$ with $\delta(s_\varphi, a) = s_0(\mathscr{A}_\psi)$ and $\delta(s, a) = \delta(\mathscr{A}_\psi)(s, a)$ for all $s \in S(\mathscr{A}_\psi)$ and $a \in \Sigma$.

The cases when $\varphi$ is $\top, \bot, \neg a$, or $\psi_1 \vee \psi_2$ are defined in a similar way. In the construction above we assume different subformulas to have disjoint sets of states, but that $\mathscr{A}_\varphi = \mathscr{A}_\psi$ if $\varphi = \psi$. In particular, if $Z$ is a free variable in $\varphi$, then $\mathscr{A}_\varphi$ contains exactly one state $s_Z$ corresponding to $Z$.

Now for $fpd(\varphi) = n+1$, we consider the only relevant case $\varphi = \mu Z.\psi(Z)$; the construction for the remaining cases can be done along similar lines as the above. Define $\mathscr{A}_\varphi = (\Sigma, \{s_\varphi\} \cup S(\mathscr{A}_{\psi(Z)}), F(\mathscr{A}_{\psi(Z)}), \delta)$ with $\delta(s_\varphi, a) = \delta(\mathscr{A}_{\psi(Z)})(s_0(\mathscr{A}_{\psi(Z)}), a)$, $\delta(s_Z, a) = \delta(s_\varphi, a)$, and $\delta(s, a) = \delta(\mathscr{A}_{\psi(Z)})(s, a)$ for all $s$ distinct from $s_\varphi$ and $s_Z$. Here, $a \in \Sigma$ and $s_Z$ is the state obtained from the 1TAA corresponding to $Z$ during the inductive construction of $\psi(Z)$. The transition function is well-defined since $Z$ occurs guarded in $\psi(Z)$.

**Lemma 5.** *Let $\varphi$ be a closed $\mu$TPTL(1) formula and $\rho$ a timed word. Then $(i,r) \in [\![\varphi]\!]^\rho$ iff $\mathscr{A}_\varphi$ has an accepting run on $\rho^i$ with initial clock value $r$.*

**Theorem 1.** *Satisfiability in $\mu$TPTL(1) over finite words is decidable with non-primitive recursive complexity.*

The translation from $\mu$TPTL(1) formulas to 1TAA also works in the other direction, i.e., for any 1TAA there exists a closed $\mu$TPTL(1) formula $\varphi_{\mathscr{A}}$ such that $\rho \in L(\mathscr{A})$ iff $\rho \models \varphi_{\mathscr{A}}$, as we now demonstrate. The translation has the same structure as the analogous construction of $\mu$-calculus formulas from alternating automata in the untimed case.

At this point it is helpful to extend the definition of $\mu$TPTL(1) to allow fixed points in *vectorial form*. Given an $n$-dimensional vector of variables $\mathbf{Z} = (Z_1, \ldots, Z_n)$ and an $n$-dimensional vector of formulas $(\varphi_1, \ldots, \varphi_n)$, we allow for vectorial fixpoints $\mu\mathbf{Z}.(\varphi_1, \ldots, \varphi_n)$. Given a timed word $\rho$, such a vectorial fixed point is interpreted as an element of the $n$-fold product $\mathscr{V}(\rho)^n$ according to the following rule, where $M_i$ is the $i$-th component of $\mathbf{M}$:

$$[\![\mu\mathbf{Z}.(\varphi_1, \ldots, \varphi_n)]\!]^\rho_\xi := \bigcap\{\mathbf{M} \in \mathscr{V}(\rho)^n \mid [\![\varphi_i]\!]^\rho_{\xi[\mathbf{Z}\mapsto\mathbf{M}]} \subseteq M_i \text{ for all } 1 \le i \le n\}\,.$$

Let $\rho$ be a timed word and let $\pi_i$ denote the $i$-th projection $\mathscr{V}(\rho)^n \to \mathscr{V}(\rho)$ for $1 \le i \le n$.

**Proposition 1.** *Given a vectorial fixed point formula $\mu\mathbf{Z}.(\varphi_1, \ldots, \varphi_n)$, for each $i \in \{1, \ldots, n\}$ there is a corresponding $\mu$TPTL(1) formula $\psi_i$ such that $\pi_i([\![\varphi]\!]^\rho_\xi) = [\![\psi_i]\!]^\rho_\xi$.*

*Proof.* The proof is by repeated application of the Bekić identity

$$\pi_1[\![\mu(Y,\mathbf{Z}).(\varphi_1, \ldots, \varphi_n)]\!]^\rho_\xi = [\![\mu Y.\varphi_1[\mathbf{Z}/\mu\mathbf{Z}.(\varphi_2, \ldots, \varphi_n)]]\!]^\rho_\xi\,.$$

This identity is valid in any complete lattice, so holds for our semantics.

Now let $\mathscr{A} = (\Sigma, S, s_0, F, \delta)$ be a 1TAA. Let $\mathbf{Z} = (Z_s \mid s \in S)$ be a vector of variables indexed by the set of locations of $\mathscr{A}$. Recall from Definition 5 that the transition function $\delta$ of $\mathscr{A}$ takes values in the set of expressions $\Phi(S)$. The first step in defining $\varphi_{\mathscr{A}}$ is to give a translation mapping each expression $\varphi$ in $\Phi(S)$ to a corresponding $\mu$TPTL(1) formula $\varphi^\ddagger$ with free variables in $\mathbf{Z}$. To this end we write:

$$\begin{array}{lll} \mathbf{tt}^{\ddagger} = \top & (\varphi_1 \wedge \varphi_2)^{\ddagger} = \varphi_1^{\ddagger} \wedge \varphi_2^{\ddagger} & (x \sim c)^{\ddagger} = x \sim c \\ \mathbf{ff}^{\ddagger} = \bot & (\varphi_1 \vee \varphi_2)^{\ddagger} = \varphi_1^{\ddagger} \vee \varphi_2^{\ddagger} & (x.\varphi)^{\ddagger} = x.\varphi^{\ddagger} \\ & & s^{\ddagger} = \bigcirc Z_s \end{array}$$

For each location $s \in S$ we define a $\mu$TPTL(1) formula $\varphi_s(\mathbf{Z})$, where

$$\varphi_s(\mathbf{Z}) = \begin{cases} \bigvee_{a \in \Sigma}(a \wedge \delta(s,a)^{\ddagger}) & \text{if } s \text{ is not accepting} \\ \dashv \vee \bigvee_{a \in \Sigma}(a \wedge \delta(s,a)^{\ddagger}) & \text{if s is accepting} \end{cases}$$

Recall that $s_0 \in S$ is the initial location of $\mathscr{A}$. We define $\varphi_{\mathscr{A}}$ to be the $\mu$TPTL(1) formula that is equivalent to the $s_0$-th component of the vectorial fixed point $\mu\mathbf{Z}.(\varphi_s \mid s \in S))$. Such a formula is guaranteed to exist by Proposition 1.

**Theorem 2.** *Let $\mathscr{A}$ be a 1TAA and $\varphi_{\mathscr{A}}$ its corresponding $\mu$TPTL(1) formula $\varphi$. Then $L(\mathscr{A}) = L(\varphi_{\mathscr{A}})$.*

This result, together with the construction underlying Theorem 1, shows that $\mu$TPTL(1) characterises the class of languages accepted by one-clock timed alternating automata.

## *Alphabetised Parallel Composition of TPTL(1) Formulas*

In this section we consider TPTL(1) extended with the alphabetised parallel composition operator $\|$ and show the decidability of the augmented logic. This extension is useful for specifying systems that run independently subject to sharing some events in common. As an example, consider the following specification:

$$(\Box(processed \to \bigcirc_{\leq 1} queued)) \parallel (\Box(queued \to \bigcirc_{=1} send)).$$

It describes a system consisting of a processor and a sender that run independently of each other and only synchronise on the *queued* event. The specification requires that a processed item be queued by the processor in the *next* step within one time unit, and that each queued item be sent by the sender in the *next* step one time unit later. However, in the timed trace of the composed system internal events from the sender may occur between a *processed* and *queue*-event of the processor. This issue is resolved by the $\|$ operator which ensures that the events unrelated to each specification do not interfere with it.

Formally, we augment the syntax of TPTL(1) in Definition 2 with an additional term for the *alphabetised parallel composition* $\varphi_1 \parallel \varphi_2$. The *alphabetised parallel composition* of timed words $\rho_1$ and $\rho_2$ over the alphabets $\Sigma_1$ and $\Sigma_2$ respectively is defined as follows: $\rho \in \rho_1 \parallel \rho_2 \subseteq T(\Sigma_1 \cup \Sigma_2)^*$ iff $\rho_1 = \rho \setminus (\Sigma_2 - \Sigma_1)$ and $\rho_2 = \rho \setminus (\Sigma_1 - \Sigma_2)$. Informally speaking, the timed events from $\rho_1$ and $\rho_2$ are merged in $\rho$ with the requirement that the timed events from $\Sigma_1 \cap \Sigma_2$ occur at the same time points in both $\rho_1$ and $\rho_2$. The semantics of TPTL(1) together with alphabetised parallel composition is obtained by adding the following clause to Definition 3:

$$[\![\varphi_1 \parallel \varphi_2]\!]^\rho := \{(i,r) \mid \exists \rho_1, \rho_2 . \rho^i \in \rho_1 \parallel \rho_2, (1,r) \in [\![\varphi_1]\!]^{\rho_1}, (1,r) \in [\![\varphi_2]\!]^{\rho_2}\}$$

We have seen in the previous section how to construct a 1TAA from a TPTL(1) formula $\varphi$. We now extend this construction to show decidability of TPTL(1) with alphabetised parallel composition. For $i \in \{1,2\}$, given TPTL(1) formulas $\varphi_i$ with their corresponding event alphabets $\Sigma_i$ and 1TAA $\mathscr{A}_i$, we show how to construct a 1TAA $\mathscr{A}_1 \parallel \mathscr{A}_2$ such that $L(\mathscr{A}_1 \parallel \mathscr{A}_2) = L(\varphi_1 \parallel \varphi_2)$. Let $\mathscr{A}_1 = (\Sigma_1, S, s_0, F, \delta)$, define $\mathscr{A}_1$ extended with $\Sigma_2$ as $\mathscr{A}_1^{\Sigma_2} := (\Sigma_1 \cup \Sigma_2, S, s_0, F, \delta')$, where $\delta'(s,a) = \{s\}$ for all $a \in \Sigma_2 - \Sigma_1$ and $\delta'(s,a) = \delta(s,a)$ otherwise. Without loss of generality we assume the set of states of $\mathscr{A}_1$ and $\mathscr{A}_2$ to be disjoint. Define $\mathscr{A}_1 \parallel \mathscr{A}_2 := (\Sigma_1 \cup \Sigma_2, S(\mathscr{A}_1) \cup S(\mathscr{A}_2) \cup \{s_\parallel\}, \{s_\parallel\}, F(\mathscr{A}_1) \cup F(\mathscr{A}_2), \delta)$, where

$$\delta(s,a) = \begin{cases} \delta(\mathscr{A}_1^{\Sigma_2})(s,a) & \text{if } s \in S(\mathscr{A}_1) \\ \delta(\mathscr{A}_2^{\Sigma_1})(s,a) & \text{if } s \in S(\mathscr{A}_2) \\ \delta(\mathscr{A}_1^{\Sigma_2})(s_0(\mathscr{A}_1),a) \wedge \delta(\mathscr{A}_2^{\Sigma_1})(s_0(\mathscr{A}_2),a) & \text{if } s = s_\parallel \end{cases}$$

Decidability of TPTL(1) + alphabetised parallel composition is then a consequence of the following lemma.

**Lemma 6.** *Let $\mathscr{A}_1$ and $\mathscr{A}_2$ be two 1TAA over the alphabets $\Sigma_1$ and $\Sigma_2$ respectively. Then $\mathscr{A}_1 \parallel \mathscr{A}_2$ has an accepting run on $\rho$ iff $\mathscr{A}_1$ has an accepting run on $\rho \setminus (\Sigma_2 - \Sigma_1)$ and $\mathscr{A}_2$ has an accepting run on $\rho \setminus (\Sigma_1 - \Sigma_2)$.*

*Proof.* Given an accepting run $C_0 \overset{d_1}{\rightsquigarrow} C_1 \xrightarrow{\sigma_1} \ldots \xrightarrow{\sigma_n} C_{2n}$ of $\mathscr{A}_1 \parallel \mathscr{A}_2$ on $\rho$, by exhaustively replacing $C_{2i-2} \overset{d_i}{\rightsquigarrow} C_{2i-1} \xrightarrow{\sigma_i} C_{2i} \overset{d_{i+1}}{\rightsquigarrow} C_{2i+1}$ with $C_{2i-2} \overset{d_i+d_{i+1}}{\rightsquigarrow} C_{2i+1}$ if $\sigma_i \in \Sigma_2 - \Sigma_1$, intersecting each remaining $C_i$ with $\{(s,r) \mid s \in S(\mathscr{A}_1), r \in \mathbb{R}_+\}$, and replacing $C_0$ with $\{(s_0(\mathscr{A}_1), 0)\}$, we obtain an accepting run of $\mathscr{A}_1$. The construction works, since $C'_{2i-2} \cap \{(s,r) \mid s \in S(\mathscr{A}_1), r \in \mathbb{R}_+\} = C'_{2i+1} \cap \{(s,r) \mid s \in S(\mathscr{A}_1), r \in \mathbb{R}_+\}$ ensures that we obtain a valid run of $\mathscr{A}_1$ on $\rho \setminus (\Sigma_2 - \Sigma_1)$. Similarly, we obtain an accepting run of $\mathscr{A}_2$ on $\rho \setminus (\Sigma_1 - \Sigma_2)$.

Conversely, let $C_0 \overset{d_1}{\rightsquigarrow} C_1 \xrightarrow{\sigma_1} \ldots \xrightarrow{\sigma_n} C_{2n}$ be an accepting run of $\mathscr{A}_1$ on $\rho_1$. This run can be altered to become an accepting run of $\mathscr{A}_1^{\Sigma_2}$ on $\rho$. In general, for $(\sigma_j, \tau_j)(\sigma_{j+1}, \tau_{j+1}) \ldots (\sigma_k, \tau_k)$ in $\rho$ with $\sigma_j, \sigma_k \in \Sigma_1$, $\sigma_\ell \in \Sigma_2 - \Sigma_1$ for $j < \ell < k$ and $(\sigma_j, \tau_j)$ equal to $(\sigma_i, \tau_i)$ in $\rho_1$, $C_{2i-1} \xrightarrow{\sigma_i} C_{2i} \overset{\tau_{i+1}(\rho_1) - \tau_i(\rho_1)}{\rightsquigarrow} C_{2i+1}$ can be exhaustively replaced with $C_{2i-1} \xrightarrow{\sigma_i(\rho_1)} C_{2i} \overset{\tau_{j+1}(\rho) - \tau_j(\rho)}{\rightsquigarrow} C_{2i} + \tau_{j+1}(\rho) - \tau_j(\rho) \xrightarrow{\sigma_{j+1}} \ldots \overset{\tau_{i+1}(\rho_1) - \tau_{k-1}(\rho)}{\rightsquigarrow} C_{2i+1} \xrightarrow{\sigma_{i+1}(\rho_1)} C_{2i+2}$ in order to obtain an accepting run of $\mathscr{A}_1^{\Sigma_2}$ on $\rho$. Then by joining the accepting runs of $\mathscr{A}_1^{\Sigma_2}$ and $\mathscr{A}_2^{\Sigma_1}$ and setting $C_0 = \{(s_\parallel, 0)\}$ we obtain an accepting run of $\mathscr{A}_1 \parallel \mathscr{A}_2$ on $\rho$.

**Theorem 3.** *Satisfiability in TPTL(1) augmented with alphabetised parallel composition is decidable over finite words with non-primitive recursive complexity.*

It is not hard to see that it is possible to combine the inductive constructions of 1TAA from $\mu$TPTL(1) and TPTL(1) together with alphabetised parallel composition. Hence satisfiability for TPTL(1) augmented both with fixpoint operators and alphabetised parallel composition is decidable.

**Theorem 4.** *Satisfiability for TPTL(1) augmented with fixpoint operators and alphabetised parallel composition is decidable over finite words, with non-primitive recursive complexity.*

## 4 Undecidable Cases

In this section we show that augmenting MTL (and a fortiori also TPTL(1)) with any of hiding, renaming, or asynchronous parallel composition renders the corresponding satisfiability problem undecidable.

To establish these results, we reduce the reachability problem for deterministic two-counter machines (2CM) to satisfiability for MTL with the extensions under consideration. A 2CM $\mathscr{M} = (S, init, \delta)$ is a finite-state automaton augmented with two counters over the naturals, where $S$ is a finite set of states, $init \in S$ is the initial location and $\delta$ is the transition function. A configuration of $\mathscr{M}$ is a triple $(s, n_0, n_1) \in S \times \mathbb{N} \times \mathbb{N}$. From a given configuration, the transition function can test each of the counters for zero and accordingly change configurations by jumping to a new location and incrementing, decrementing, or leaving each of the counters untouched. A run of a 2CM is a finite sequence of configurations that is consistent with the transition function. The reachability problem asks whether for a given 2CM $\mathscr{M}$ it is possible to reach a configuration $(s, 0, 0)$ starting from the initial configuration $(init, 0, 0)$. This problem is well-known to be undecidable [18].

Following [1] and [7], we can encode a run of an $m$-location 2CM $\mathscr{M}$ as a timed word $\rho$ over the alphabet $\Sigma = \{a, b_1, \dots, b_m, c\}$. The $a$-events are used to encode the value of the counters in unary, each $b_i$ represents a location of $\mathscr{M}$, and $c$ is used as a marker. The $i$-th configuration $(s_j, v_1, v_2)$ of a run is stored in the interval $[i, i+1)$ of $\rho$. The event $b_j$ occurs at time $i$, representing the current location $s_j$. In the following, let $\mathscr{I} := (0, 0.25)$. The number of $a$-events in the interval $\mathscr{I} + i$ encodes the value of the first counter. Likewise, the value of the second counter is encoded in the interval $\mathscr{I} + i + 0.5$. The marker $c$ occurs at time $i + 0.5$ and the remaining intervals in $[i, i+1)$ do not contain any $a$- or $b_i$-events. We assume that $init = s_1$, so that $b_1$ is the first event to occur. It is not hard to see that we can construct an MTL[2] formula $\varphi_{\mathscr{M}}$ such that $\varphi_{\mathscr{M}} \wedge \Diamond b_i$ is satisfiable if $(s_i, 0, 0)$ is reachable. The converse however does not hold, since MTL is incapable of detecting *insertion errors*.

[2] This even holds for the until-free fragment of MTL, which is obtained from Definition 4 by dropping the $\mathscr{U}_{\mathscr{I}}$-definition and introducing $\Box_{\mathscr{I}}$ and $\Diamond_{\mathscr{I}}$ as primitives.

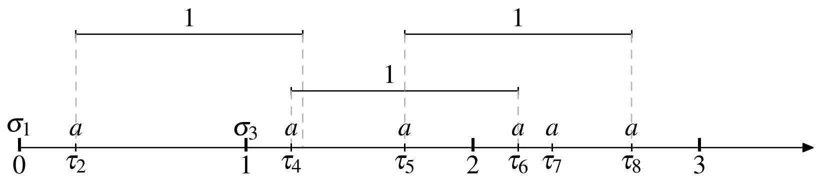

**Fig. 1** Illustration of a timed word suffering from insertion errors.

**Definition 8.** Let $\rho \models \varphi_{\mathscr{M}}$ be a timed word representing a run of $\mathscr{M}$. Then $\rho$ **suffers from insertion errors** if there are $1 \leq i < j \leq |\rho|$ and $\sigma_i, \sigma_j \in \Sigma$ such that $\rho$ contains $(\sigma_i, \tau_i)(a, \tau_{i+1})$ and $(\sigma_j, \tau_i + 1)(a, \tau_{j+1})$ with $\tau_{j+1} < \tau_{i+1} + 1$.

An illustration of this definition is presented in Figure 1. First, consider the events $(\sigma_1, 0)$ and $(a, \tau_2)$. We have that $(\sigma_1, 0)$ is followed one time unit later by $(\sigma_3, 1)$ which itself is followed by $(a, \tau_4)$. However, $\tau_4 < \tau_2 + 1$ and hence $(a, \tau_4)$ is wrongly inserted. Observe that $(a, \tau_2)$ does not have any corresponding event one time unit later. Second, the $a$-event at time $\tau_7$ is also wrongly inserted, since it lies strictly between $\tau_4 + 1$ and $\tau_5 + 1$.

D'Souza and Prabhakar show in [7] that MTL augmented with any extension that is able to characterise a slightly more restricted version of the language from Definition 8 has an undecidable satisfiability problem. We now use their observation to establish undecidability of satisfiability of MTL extended with any of hiding, renaming, or asynchronous parallel composition. For each extension, we define a formula $\varphi_{ie}$ that is capable of detecting insertion errors. Whence there exists a run of $\mathscr{M}$ reaching $s_i$ iff $\varphi_{\mathscr{M}} \wedge \Diamond b_i \wedge \neg \varphi_{ie}$ is satisfiable.

## 4.1 Hiding

Let $E \subseteq \Sigma$ be a set of events. We augment the syntax of TPTL(1) in Definition 2 with an additional term for the *hiding* operator $\backslash E$. In designing specifications, hiding is used to abstract away irrelevant events. For example, given a set $I \subseteq \Sigma$, the formula $(\Box(\varphi \rightarrow \bigcirc_{<1} \psi)) \setminus I$ specifies a bounded response property that 'ignores' events from $I$ that could occur between $\varphi$ and $\psi$. Formally, the semantic mapping for hiding is obtained by adding the following clause to Definition 3:

$$[\![\varphi \setminus E]\!]^{\rho} := \{(i,r) \mid \exists \rho'.\rho^i = \rho' \setminus E \text{ and } (1,r) \in [\![\varphi]\!]^{\rho'}\}$$

It has been observed in [10] that hidden propositions lead to an undecidable satisfiability problem for real-time logics when the underlying time model is dense. In order to detect insertion errors, we add $d$ to the event alphabet and use the hiding operator in the following way: $\rho$ suffers from insertion errors (following Definition 8) iff we can insert a $d$-event into $\rho$ immediately preceding an $a$-event, in such a way that $d$ is followed exactly one time unit later by an $a$. Formally:

**Lemma 7.** *Let $\varphi_{ie} = (\varphi_{sm} \wedge \Diamond(d \wedge \Diamond_{\mathscr{I}} a \wedge \Diamond_{=1} a)) \setminus \{d\}$ and $\rho \models \varphi_{\mathscr{M}}$ be a timed word representing a run of $\mathscr{M}$. Then $\rho$ suffers from insertion errors iff $\rho \models \varphi_{ie}$.*

(In the above, $\varphi_{sm}$ stands for a formula that captures precisely all strongly monotonic timed words, i.e., words in which no two events share the same timestamp.)

The lemma shows that hiding renders the satisfiability problem undecidable even if applied at the outermost level, i.e., checking satisfiability of $\varphi \setminus E$ is undecidable for MTL formulas $\varphi$. This is not the case for MITL, where checking satisfiability of $\varphi \setminus E$ for some MITL formula $\varphi$ still is decidable [10].

## 4.2 Renaming

Let $R \subseteq \Sigma \times \Sigma$ be a total renaming relation over $\Sigma$. We augment the syntax of TPTL(1) in Definition 2 with an additional term for the *renaming* operator $[R]$. Let us write that $\rho \in \rho'[R]$ iff $|\rho| = |\rho'|$, $\tau_i(\rho) = \tau_i(\rho')$, and $\sigma_i(\rho) \in R(\sigma_i(\rho'))$ for all $1 \le i \le |\rho|$. The semantics of TPTL(1) together with renaming is obtained by adding the following clause to Definition 3:

$$[\![\varphi[R]]\!]^{\rho} := \{(i,r) \mid \exists \rho'.\rho^{i} \in \rho'[R] \text{ and } (1,r) \in [\![\varphi]\!]^{\rho'}\}$$

The effect of renaming is less drastic than that of hiding, since it does not delete timed events from timed words. It however still provides a convenient means of abstraction in specifications. As an example, given a set $I \subseteq \Sigma$ of internal events and renaming relation $R := \{i/b\}_{i \in I} \cup Id_{\Sigma \setminus I}$, the formula $(\Box(\varphi \rightarrow (b\, \mathscr{U}_{<1}\, \psi))[R]$ expresses a bounded response property that treats all events from $I$ in the same way by grouping them into a single event $b$.

Using the renaming operator to detect insertion errors is slightly more involved than in the previous case, and we describe the procedure with the help of an example given in Figure 2. Observe that the $a$-event at time $\tau_3$ is wrongly inserted in the timed word shown in the lower part of Figure 2. Our tactic is to non-deterministically rename some $a$-events to $d$-events in this timed word in such a way as to identify the wrongly inserted $a$-event. Such a renaming is shown in the upper part of the figure. There we have that the $a$-event at time $\tau_1$ is immediately followed by exactly one $d$-event, which itself is followed exactly one time unit later by a $d$-event. For the event at time $\tau_1$, we can then check that there is in strictly more than one time unit later an $a$-event followed immediately by a $d$-event—which identifies the wrongly inserted $a$-event.

The formulas below also have to take account of the case in which the $a$-event at time $\tau_2$ does not have a corresponding $a$-event one time unit later and are therefore somewhat trickier to read. However, it is not hard to check that they capture the intuition described above.

**Lemma 8.** *Let $\rho \models \varphi_{\mathscr{M}}$ be a timed word representing a run of $\mathscr{M}$, and let*

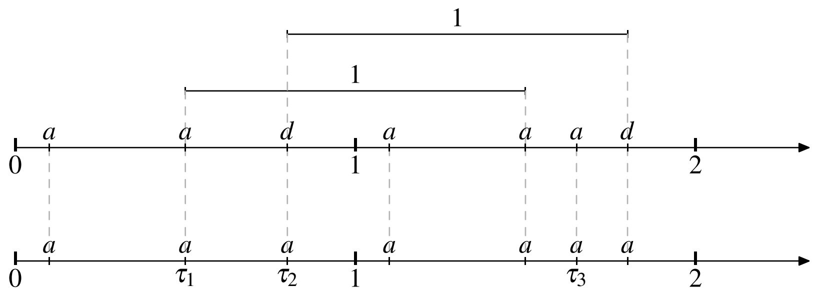

**Fig. 2** On the bottom, a timed word suffering from an insertion error, and above its renaming that allows one to detect it.

$$\psi := \Box_{\mathscr{I}}((a \vee d) \rightarrow \Box_{\mathscr{I}} \neg d) \wedge \Diamond_{\mathscr{I}}(d \wedge (\Diamond_{=1}(d \wedge \Box_{\mathscr{I}} \neg d) \vee \Box_{[1,1.25)} \bot))$$
$$\varphi_{ie} := \Diamond(\psi \wedge \Diamond_{=1}((\Diamond_{\mathscr{I}}(a \wedge \Diamond_{\mathscr{I}} d) \vee (\Box_{\mathscr{I}} \neg d \wedge \Diamond_{\mathscr{I}} a))))[\{(d,a)\} \cup Id_{\Sigma \setminus \{d\}}]$$

*Then $\rho$ suffers from insertion errors iff $\rho \models \varphi_{ie}$.*

## 4.3 Asynchronous Parallel Composition

We augment the syntax of TPTL(1) in Definition 2 with an additional term for the *asynchronous parallel composition* operator $|||$, also known as *interleaving* and *(timed) shuffle product*. This operator is similar to its alphabetised counterpart in that it allows one to express specifications on systems that run concurrently.

Given timed words $\rho$, $\rho_1$, and $\rho_2$ with $|\rho| = n$, $|\rho_1| = n_1$, and $|\rho_2| = n_2$, we let $\rho \in \rho_1 \mathbin{|||} \rho_2$ iff the set of positions $\{1,\ldots,n\}$ of $\rho$ can be partitioned into disjoint sets $\{i_1,\ldots,i_{n_1}\}$ and $\{j_1,\ldots,j_{n_2}\}$ such that $\sigma_k(\rho_1) = \sigma_{i_k}(\rho)$, $\tau_k(\rho_1) = \tau_{i_k}(\rho)$ for $1 \le k \le n_1$, and $\sigma_\ell(\rho_2) = \sigma_{j_\ell}(\rho)$, $\tau_\ell(\rho_2) = \tau_{j_\ell}(\rho)$ for $1 \le \ell \le n_2$.

The semantics of TPTL(1) together with asynchronous parallel composition is obtained by adding the following clause to Definition 3:

$$[\![\varphi_1 \mathbin{|||} \varphi_2]\!]^\rho := \{(i,r) \mid \exists \rho_1,\rho_2.\rho^i = \rho_1 \mathbin{|||} \rho_2, (1,r) \in [\![\varphi_1]\!]^{\rho_1}, (1,r) \in [\![\varphi_2]\!]^{\rho_2}\}$$

In order to show undecidability, we use the fact $\varphi_{\mathscr{M}}$ holds on timed words with *and* without insertion errors. Consequently, the interleaving $\varphi_{\mathscr{M}} \mathbin{|||} a$ only holds on timed words *with* insertion errors.

**Lemma 9.** *Let $\rho \models \varphi_{\mathscr{M}}$ be a timed word representing a run of $\mathscr{M}$ and let $\varphi_{ie} := \varphi_{\mathscr{M}} \mathbin{|||} a$. Then $\rho$ suffers from insertion errors iff $\rho \models \varphi_{ie}$.*

We sum up the results of this section in the following theorem.

**Theorem 5.** *The satisfiability problem for MTL augmented with any of hiding, renaming, or asynchronous parallel composition is undecidable.*

## 5 Conclusion

In this paper, we have considered various extensions of the central timed specification formalism of Metric Temporal Logic by process-algebraic operators originating from Tony Hoare's Communicating Sequential Processes. We have argued that such extensions, each of which strictly enhances the expressive power of MTL, allow for more natural and versatile specification of timed systems.

On the positive side, we have shown that MTL augmented with both fixpoint operators and alphabetised parallel composition remains decidable. On the other hand, the addition of any of hiding, renaming, or asynchronous parallel composition (also known as interleaving and shuffle product) immediately yields undecidability.

One of our main technical tools has been the one-clock fragment of Timed Propositional Temporal Logic, TPTL(1). We have shown that extending this formalism with fixpoint operators provides a precise logical characterisation of the class of languages accepted by one-clock timed alternating automata, a result of independent interest. An intriguing question is whether the fixpoint-extension of the $n$-clock fragment of TPTL precisely characterises the class of languages accepted by $n$-clock timed alternating automata, thereby extending Theorem 2 (notwithstanding the fact that such languages are in general not recursive).

An interesting avenue for future work would be to investigate more thoroughly the methodological applications of our decidability results towards the specification of timed systems themselves built from recursive and concurrent components, such as Timed CSP processes.

## References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
2. Rajeev Alur and Thomas A. Henzinger. Real-time logics: complexity and expressiveness. Technical report, Stanford University, 1990.
3. Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *J. ACM*, 41(1):181–203, 1994.
4. Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of TPTL and MTL. In *Proceedings of FSTTCS*, volume 3821 of *Lecture Notes in Computer Science*, pages 432–443. Springer, 2005.
5. Patricia Bouyer, Nicolas Markey, Joël Ouaknine, and James Worrell. On expressiveness and complexity in real-time model checking. In *Proceedings of ICALP*, volume 5126 of *Lecture Notes in Computer Science*, pages 124–135. Springer, 2008.
6. Stephen D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, 1984.
7. Deepak D'Souza and Pavithra Prabhakar. On the expressiveness of MTL in the pointwise and continuous semantics. *STTT*, 9(1):1–4, 2007.
8. Christoph Haase, Joël Ouaknine, and James Worrell. On extensions of Metric Temporal Logic. Technical report, Oxford University Computing Laboratory, 2009. http://www.comlab.ox.ac.uk/files/2180/how-09.pdf.
9. Thomas A. Henzinger. *The temporal specification and verification of real-time systems*. PhD thesis, Stanford University, 1992.

10. Thomas A. Henzinger. Its about time: Real-time logics reviewed. In *Proceedings of CONCUR*, volume 1466 of *Lecture Notes in Computer Science*, pages 439–454, 1998.
11. Thomas A. Henzinger, Jean-François Raskin, and Pierre-Yves Schobbens. The regular real-time languages. In *Proceedings of ICALP*, volume 1443 of *Lecture Notes in Computer Science*, pages 580–591. Springer, 1998.
12. C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, 1985.
13. Ron Koymans. Specifying real-time properties with Metric Temporal Logic. *Real-Time Syst.*, 2(4):255–299, 1990.
14. Martin Lange. Weak automata for the linear time $\mu$-calculus. In *Proceedings of VMCAI*, volume 3385 of *Lecture Notes in Computer Science*, pages 267–281. Springer, 2005.
15. Slawomir Lasota and Igor Walukiewicz. Alternating timed automata. *ACM Trans. Comput. Logic*, 9(2):1–27, 2008.
16. Alain J. Mayer and Larry J. Stockmeyer. The complexity of PDL with interleaving. *Theor. Comput. Sci.*, 161(1-2):109–122, 1996.
17. Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.
18. Marvin L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967.
19. Joël Ouaknine and James Worrell. On Metric Temporal Logic and faulty Turing machines. In *Proceedings of FoSSaCS*, volume 3921 of *Lecture Notes in Computer Science*, pages 217–230. Springer, 2006.
20. Joël Ouaknine and James Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logic. Meth. Comp. Sci.*, 3(1), 2007.
21. Joël Ouaknine and James Worrell. Some recent results in Metric Temporal Logic. In *Proceedings of FORMATS*, volume 5215 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2008.
22. Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):115–125, 1959.
23. George M. Reed and A. W. Roscoe. A timed model for Communicating Sequential Processes. In *Proceedings of ICALP*, volume 226 of *Lecture Notes in Computer Science*, pages 314–323. Springer, 1986.
24. George M. Reed and A. W. Roscoe. Metric spaces as models for real-time concurrency. In *Proceedings of MFPS*, volume 298 of *Lecture Notes in Computer Science*, pages 331–343. Springer, 1987.
25. George M. Reed and A. W. Roscoe. The timed failures-stability model for CSP. *Theor. Comput. Sci.*, 211(1-2):85–127, 1999.
26. A. Prasad Sistla, Moshe Y. Vardi, and Pierre Wolper. The complementation problem for Büchi automata with applications to temporal logic (extended abstract). In *Proceedings of ICALP*, volume 194 of *Lecture Notes in Computer Science*, pages 465–474. Springer, 1985.