MTL and TPTL for One-Counter Machines: Expressiveness, Model Checking, and Satisfiability

SHIGUANG FENG, Universität Leipzig
CLAUDIA CARAPELLE and OLIVER FERNÁNDEZ GIL, TU Dresden
KARIN QUAAS, Universität Leipzig

Metric Temporal Logic (MTL) and Timed Propositional Temporal Logic (TPTL) are quantitative extensions of Linear Temporal Logic (LTL) that are prominent and widely used in the verification of real-timed systems. We study MTL and TPTL as specification languages for one-counter machines. It is known that model checking one-counter machines against formulas of Freeze LTL (FLTL), a strict fragment of TPTL, is undecidable. We prove that in our setting, MTL is strictly less expressive than TPTL, and incomparable in expressiveness to FLTL, so undecidability for MTL is not implied by the result for FLTL. We show, however, that the model-checking problem for MTL is undecidable. We further prove that the satisfiability problem for the unary fragments of TPTL and MTL are undecidable; for TPTL, this even holds for the fragment in which only one register and the finally modality is used. This is opposed to a known decidability result for the satisfiability problem for the same fragment of FLTL.

CCS Concepts: • **Theory of computation** → **Logic and verification**; **Verification by model checking**; *Automata over infinite objects*;

Additional Key Words and Phrases: One-counter machines, metric temporal logic, timed propositional temporal logic, Freeze LTL, data words, Ehrenfeucht-Fraïssé games

ACM Reference format:
Shiguang Feng, Claudia Carapelle, Oliver Fernández Gil, and Karin Quaas. 2019. MTL and TPTL for One-Counter Machines: Expressiveness, Model Checking, and Satisfiability. *ACM Trans. Comput. Logic* 21, 2, Article 12 (December 2019), 34 pages.
https://doi.org/10.1145/3372789

1 INTRODUCTION

In recent years, there has been an active research programme on structures that enrich classical words and trees with additional information. *Data words* have attracted remarkable interest, especially in the fields of program verification and database management. A data word is a finite or infinite sequence in which every position carries two pieces of information: a label from a finite alphabet and a data value from an infinite data domain. This simple model captures *timed words* accepted by timed automata [Alur and Dill 1994], which are words of the form $(a_1, t_1)(a_2, t_2)\ldots$, where t_i is a non-negative real-numbered *time value* that gives the precise time of action a_i (from a

Authors' addresses: S. Feng and K. Quaas, Universität Leipzig, Germany; emails: shig.feng@gmail.com, quaas@informatik.uni-leipzig.de; C. Carapelle and O. Fernández Gil, TU Dresden, Germany; emails: claudia.carapelle@gmail.com, oliver.fernandez@tu-dresden.de.

1529-3785/2019/12-ART12 $15.00
https://doi.org/10.1145/3372789

finite alphabet); thus, we have $t_i \leq t_{i+1}$ (i.e., the sequence of time values is monotonically increasing). In this article, we regard data words most and for all as behavioural models of *one-counter machines*. A one-counter machine is a finite-state controller operating on a counter ranging over the non-negative integers. The counter can be incremented, decremented, and tested for zero. Thus, computations of one-counter machines can be regarded as data words where the data comes from the non-negative integers, representing the current value of the counter. In this article, whenever we speak of data words, we mean data words over the non-negative integers. Note that in data words, the sequence of data values does not need to be monotonically increasing as is the case for timed words.

One-counter machines are a simple yet fundamental computational model that has proved useful, such as in the verification of cryptographic protocols [Lafourcade et al. 2005] and in the validation of XML streams [Chitic and Rosu 2004]. One-counter machines have received a lot of interest in the recent past (e.g., see [Bollig et al. 2017; Demri et al. 2010; Feng et al. 2017; Göller et al. 2010, 2012; Göller and Lohrey 2013; Haase et al. 2016; Hofman et al. 2016; Lechner et al. 2016; Serre 2006], to mention only some of the very recent articles).

For reasoning about data words, various formalisms have been considered, ranging from data automata [Bojańczyk et al. 2006; Bouyer et al. 2003] and register automata [Demri and Lazic 2009; Kaminski and Francez 1994; Neven et al. 2004; Sakamoto and Ikeda 2000], first-order logics for data words [Bojańczyk et al. 2006; Bouajjani et al. 2007], to extensions of temporal logics [Demri and Lazic 2009; Demri et al. 2007, 2010; Lisitsa and Potapov 2005]. *Linear temporal logic* (LTL) is now one of the main logical formalisms for describing system behaviour. It is long known that model checking against formulas of LTL is decidable for timed automata, vector addition systems with states [Esparza 1996], and pushdown systems [Bouajjani et al. 1997]. Triggered by real-time applications, various extensions of LTL have been studied. Two of the most prominent examples are *metric temporal logic* (MTL) [Koymans 1990] and *timed propositional temporal logic* (TPTL) [Alur and Henzinger 1994]. In MTL, the *until* modality (U) is annotated by time intervals. For instance, the formula $p \,\mathsf{U}_{[2,3)}\, q$ holds at time t, if there is a time $t' \in [t+2, t+3)$, where q holds, and p holds during the interval $[t, t')$. TPTL is a more powerful logic that is equipped with a freeze mechanism. It uses register variables that can be set to the current time value; later, the values stored in the register variables can be compared with the current time value. For instance, the preceding MTL formula $p \,\mathsf{U}_{[2,3)}\, q$ is equivalent to the TPTL formula $x.(p \,\mathsf{U}\, (q \wedge 2 \leq x < 3))$. Here, the constraint $2 \leq x < 3$ should be read as such: the difference of the current time value and the value stored in x is in the interval $[2, 3)$.

In the context of real-timed systems, the satisfiability problem (SAT) and the model-checking problem for MTL and TPTL have been studied intensively. Model checking timed automata against formulas of MTL is decidable and hyper-Ackermann-complete for finite timed words [Ouaknine and Worrell 2007; Schmitz and Schnoebelen 2011]; it is undecidable for infinite timed words [Ouaknine and Worrell 2006a] and for timed automata equipped with additional resources [Bouyer et al. 2008]. The SAT for TPTL is undecidable already for finite timed words [Alur and Henzinger 1993]. For the conceptually simpler model of timed words over the discrete time domain (where the time values stem from the non-negative integers instead of the non-negative real numbers), the model-checking problem and the SAT for TPTL are EXPSPACE-complete [Alur and Henzinger 1994; Laroussinie et al. 2002]. When evaluated over timed words over the discrete time domain, MTL and TPTL have the same expressive power [Alur and Henzinger 1993], and the EXPSPACE-completeness result also holds for MTL. Note, however, that these results only hold for the case that the sequence of data values occurring in a data word is *monotonic*.

For analysing general data words over the non-negative integers (i.e., without the restriction of being monotonic), a strict fragment of TPTL, called *Freeze LTL* (FLTL), has received much attention [Demri and Lazic 2009; Demri et al. 2010; Demri and Sangnier 2010; Figueira and Segoufin 2009]. Like in TPTL, FLTL can store the current data value in a register variable using the freeze mechanism. But in contrast to TPTL, the value of the register variable can be compared with the current data value only for (in)equality. Despite this limited access to data values, the SAT for FLTL is undecidable [Demri and Lazic 2009]. However, over *finite* data words, and if the logic is restricted to a single register, then the SAT is decidable and Ackermann-complete [Demri and Lazic 2009; Figueira et al. 2011]. The lower bound has been confirmed for the fragment of FLTL where the only temporal modality is the *finally* modality [Figueira and Segoufin 2009]. In contrast to this, model checking one-counter machines against formulas of FLTL is undecidable, even for the fragment that only allows for a single-register [Demri et al. 2008]. On the positive side, the model-checking problem becomes decidable and PSPACE-complete [Demri et al. 2010] for *deterministic* one-counter machines; this result has been generalized recently to TPTL [Feng et al. 2017].

Since FLTL is a fragment of TPTL, it is clear that one cannot obtain better decidability and complexity results for TPTL than for FLTL. In fact, it is long known that the SAT for TPTL over data words is undecidable [Alur and Henzinger 1994]. With MTL, however, the situation is not clear: although MTL and TPTL are equally expressive over *monotonic* data words [Alur and Henzinger 1993], the relative expressiveness between MTL and TPTL over data words is not known. Indeed, for timed words with timestamps in the non-negative real numbers, TPTL is strictly more expressive than MTL [Bouyer et al. 2010] (cf., the different decidability statuses of the SAT for MTL and TPTL over finite timed words). The relative expressiveness of MTL and FLTL is not known either.

Our contribution. As a first main result, we prove that MTL is strictly less expressive than the single-register fragment of TPTL. For obtaining this result, we define a quantitative version of Ehrenfeucht-Fraïssé games (EF games, for short), tailored for proving expressiveness results for MTL and TPTL both over finite and infinite data words. We use EF games to prove further expressiveness results about several fragments of MTL and TPTL. For instance, we prove that MTL is incomparable in expressiveness to the single-register fragment of FLTL, and the single-register fragment of TPTL is strictly less expressive than the two-register fragment of TPTL. Our EF games provide a very general and intuitive tool to prove results concerning the expressive power of quantitative logics, which is of independent interest. We would like to mention that a similar quantitative extension of EF games has been developed to investigate relative expressiveness of some fragments of the real-time version of MTL over *finite timed words* [Pandya and Shah 2011]. However, the proof of Theorem 1 in Pandya and Shah [2011] relies on the fact that there is an integer bound on the timestamps of a finite timed word to deal with the potentially infinite number of equivalence classes of MTL formulas; it is not clear how this can be extended to *infinite* timed words, whereas the EF game we propose here is suitable for proving expressiveness results over infinite data words as well.

Our new expressiveness results for MTL may raise hope that one can use MTL as a specification language for one-counter machines. However, as a second main result, we prove that model checking one-counter machines against formulas of MTL is undecidable, both over finite and infinite data words. This undecidability result even holds for the strict fragment of MTL in which the only temporal modality that may be used is the *finally* modality. Using similar ideas as presented for FLTL in Demri et al. [2010], we further prove a purification lemma for model checking, which implies that model checking is undecidable even if no propositional variables are used in the MTL formulas. We remark that the undecidability proofs can also be adapted to one-counter machines without zero test (i.e., one-dimensional vector addition systems with states).

Finally, we prove that the SATs for MTL and TPTL are both undecidable, even for the unary fragments of the logics. For TPTL, this undecidability result even applies to the single-register fragment in which, as a temporal modality, only the *finally* modality may be used. The results concerning the SAT for MTL and TPTL are summarized later in Table 3.

The main insight of this article is that the full logics MTL and TPTL have a very limited use in the verification of one-counter machines and in specifying properties over data words over the non-negative integers in general. This adds an important piece to complete the picture about the decidability status of SATs for data-relevant extensions of temporal logics. It also opens new directions for future research: in the context of real-time systems, and in view of the undecidability (infinite timed words), respectively, the very high complexity (finite timed words) of model checking timed automata against formulas in MTL, a plethora of fragments of MTL for which the associated verification task is more feasible than for full MTL have been studied [Alur et al. 1996; Bouyer et al. 2007; Lazic et al. 2016; Ouaknine and Worrell 2006b]. It is interesting to study whether similar results can be obtained for MTL and TPTL when they are evaluated over data words. A first step into this direction was recently established for the *flat* fragment of FLTL [Bollig et al. 2017; Lechner et al. 2016].

2 PRELIMINARIES

2.1 One-Counter Machines

A *one-counter machine* is a non-deterministic finite-state machine extended with a single counter that takes values in the non-negative integers. Formally, a one-counter machine is a tuple $\mathcal{A} = (Q, q_{\text{in}}, E, \lambda)$, where Q is a finite set of control states, q_{in} is the initial control state, $E \subseteq Q \times Q$ is a finite set of edges, and $\lambda : E \to \text{Op}$ is a function that assigns to each edge an operation from the set $\text{Op} = \{\text{zero}\} \cup \{\text{add}(a) \mid a \in \mathbb{Z}\}$. We use the operation zero to test whether the current value of the counter is equal to zero, and we use add(a) for adding a to the current value of the counter, provided that the value of the counter remains non-negative. A *configuration* of $\mathcal{A}$ is a pair (q, c), where $q \in Q$ is a control state and $c \in \mathbb{N}$ is the current value of the counter. We define a transition relation $\to$ over the set of all configurations by $(q, c) \to (q', c')$ if, and only if, there is an edge $(q, q') \in E$ with $\lambda(q, q') = \text{op}$, and if op = zero, then $c = c' = 0$, and if op = add(a) for some $a \in \mathbb{Z}$, then $c' = c + a$. A *computation* of $\mathcal{A}$ is a finite or infinite sequence $(q_0, c_0)(q_1, c_1)(q_2, c_2) \dots$ over $Q \times \mathbb{N}$ such that $(q_0, c_0) = (q_{\text{in}}, 0)$ and $(q_j, c_j) \to (q_{j+1}, c_{j+1})$ for all $j \geq 0$.

We are interested in model checking one-counter machines. The *infinitary existential model-checking problem for a logic* $\mathcal{L}$ asks, given a one-counter machine $\mathcal{A}$ and a logical formula $\varphi \in \mathcal{L}$, whether there exists some infinite computation ρ of $\mathcal{A}$ such that $\rho \models_{\mathcal{L}} \varphi$. If the answer to this question is positive, then we write $\mathcal{A} \models^{\omega}_{\mathcal{L}} \varphi$. The *finitary existential model-checking problem* is defined in an analogous manner for finite computations of $\mathcal{A}$, and we write $\mathcal{A} \models^{*}_{\mathcal{L}} \varphi$ in case of a positive answer.

2.2 Data Words

Let $\mathbb{P}$ be a countably infinite set of *atomic propositions*. A *word* over $\mathbb{P}$ is a non-empty finite or infinite sequence $p_0 p_1 p_2 \dots$, where $p_i \in \mathbb{P}$ for all $i \in \mathbb{N}$. A *data word* over $\mathbb{P}$ is a non-empty finite or infinite sequence $(p_0, d_0)(p_1, d_1) \dots$, where $(p_i, d_i) \in (\mathbb{P} \times \mathbb{D})$ for all $i \in \mathbb{N}$, and $\mathbb{D}$ is an infinite data domain. Since in this article we see data words as behaviours of one-counter machines, we assume that $\mathbb{D} = \mathbb{N}$ for the rest of the article. A data word is *monotonic* (*strictly monotonic*), if $d_i \leq d_{i+1}$ ($d_i < d_{i+1}$) for all $i \in \mathbb{N}$. We say that a data word is *pure* if it is a pure sequence of non-negative integers. We use $(\mathbb{P} \times \mathbb{N})^+$ and $(\mathbb{P} \times \mathbb{N})^{\omega}$, respectively, to denote the set of finite and infinite, respectively, data words over $\mathbb{P}$. We use $|w|$ to denote the *length* of a data word w—that is, the number of all pairs (p_i, d_i) in w. If w is infinite, then $|w| = +\infty$.

2.3 Linear Temporal Logic

Given a countably infinite set $\mathbb{P}$ of propositions, the set of formulas of LTL is built from $\mathbb{P}$ by Boolean connectives and the *until* modality U using the following grammar,

$$\varphi ::= \text{true} \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \mathrm{U} \varphi,$$

where $p \in \mathbb{P}$. Formulas of LTL are interpreted over *words*. Let $w = p_0 p_1 p_2 \dots$ be a word over $\mathbb{P}$, and let i be a position in w. We define the *satisfaction relation for LTL* inductively as follows:

- $(w, i) \models_{\mathrm{LTL}}$ true.
- $(w, i) \models_{\mathrm{LTL}} p$ if, and only if, $p = p_i$.
- $(w, i) \models_{\mathrm{LTL}} \neg\varphi$ if, and only if, $(w, i) \not\models_{\mathrm{LTL}} \varphi$.
- $(w, i) \models_{\mathrm{LTL}} \varphi_1 \wedge \varphi_2$ if, and only if, $(w, i) \models_{\mathrm{LTL}} \varphi_1$ and $(w, i) \models_{\mathrm{LTL}} \varphi_2$.
- $(w, i) \models_{\mathrm{LTL}} \varphi_1 \mathrm{U} \varphi_2$ if, and only if, there exists a position $j > i$ in w such that $(w, j) \models_{\mathrm{LTL}} \varphi_2$, and $(w, k) \models_{\mathrm{LTL}} \varphi_1$ for all positions k with $i < k < j$.

We say that a word *satisfies* an LTL formula φ, written $w \models_{\mathrm{LTL}} \varphi$, if $(w, 0) \models_{\mathrm{LTL}} \varphi$.

We use the following standard abbreviations:

$$\begin{aligned} \text{false} &:= \neg\text{true} & \mathrm{F}\varphi &:= \text{true}\mathrm{U}\varphi \\ \varphi_1 \vee \varphi_2 &:= \neg(\neg\varphi_1 \wedge \neg\varphi_2) & \mathrm{G}\varphi &:= \neg\mathrm{F}\neg\varphi \\ \varphi_1 \rightarrow \varphi_2 &:= \neg\varphi_1 \vee \varphi_2 & \mathrm{X}\varphi &:= \text{false}\,\mathrm{U}\,\varphi \end{aligned}$$

The modalities X (*next*), F (*finally*), and G (*globally*), respectively, are *unary* operators, which refer to the *next* position, *some future* position, and *all future* positions, respectively.

The finitary (infinitary, respectively) SAT for LTL is to decide, given an LTL formula φ, whether there exists a finite word (infinite word, respectively) w over $\mathbb{P}$ such that $w \models_{\mathrm{LTL}} \varphi$. We say that a formula φ is (finitary or infinitary) *satisfiable* if there exists a (finite or infinite) word w such that $w \models_{\mathrm{LTL}} \varphi$.

2.4 Relative Expressiveness

Let $\mathcal{L}$ be a logic, and let $\mathbb{C}$ be a class of data words. We say that $\mathbb{C}$ is *definable* by an $\mathcal{L}$ formula φ if for every data word w, $w \in \mathbb{C}$ if, and only if, $w \models_{\mathcal{L}} \varphi$. Two $\mathcal{L}$ formulas φ and ψ are *equivalent* over $\mathbb{C}$ if for every data word $w \in \mathbb{C}$ we have $w \models_{\mathcal{L}} \varphi$ if, and only if, $w \models_{\mathcal{L}} \psi$. We say that φ and ψ are equivalent, written $\varphi \equiv \psi$, if they are equivalent over the set of *all* data words.

Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two logics. We say that an $\mathcal{L}_1$ formula φ is *definable in* $\mathcal{L}_2$ if there is an $\mathcal{L}_2$ formula ψ such that $\varphi \equiv \psi$. We say that $\mathcal{L}_2$ is *at least as expressive as* $\mathcal{L}_1$, written $\mathcal{L}_1 \preccurlyeq \mathcal{L}_2$, if every $\mathcal{L}_1$ formula is definable in $\mathcal{L}_2$. $\mathcal{L}_2$ is *strictly more expressive than* $\mathcal{L}_1$ if $\mathcal{L}_2$ is at least as expressive as $\mathcal{L}_1$ and, additionally, there is an $\mathcal{L}_2$ formula that is not definable in $\mathcal{L}_1$. Further, $\mathcal{L}_1$ and $\mathcal{L}_2$ are *equally expressive* if $\mathcal{L}_1 \preccurlyeq \mathcal{L}_2$ and $\mathcal{L}_2 \preccurlyeq \mathcal{L}_1$. $\mathcal{L}_1$ and $\mathcal{L}_2$ are *incomparable* if neither $\mathcal{L}_1 \preccurlyeq \mathcal{L}_2$ nor $\mathcal{L}_2 \preccurlyeq \mathcal{L}_1$.

Example 2.1. In the preceding definition for LTL , we use the *strict semantics* for the *until* modality. One may also use a *non-strict semantics* of U, defined as follows:

- $(w, i) \models_{\mathrm{LTL}} \varphi_1 \mathrm{U} \varphi_2$ if, and only if, there exists a position $j \geq i$ in w such that $(w, j) \models_{\mathrm{LTL}} \varphi_2$, and $(w, k) \models_{\mathrm{LTL}} \varphi_1$ for all positions k with $i \leq k < j$.

Note, however, that in our definition, the strictness of the *until* modality is essential to derive the semantics of the *next* modality. It is well known that LTL with a strict semantics is *equally expressive* as LTL with a non-strict semantics and the *next* modality explicitly given in the syntax.

Example 2.2. We define the *until rank* of an LTL formula inductively in a natural way—that is, by "counting" the maximum number of nested until modalities (see Section 4.1 for a precise definition). Accordingly, we define $\mathrm{LTL}_k := \{\varphi \in \mathrm{LTL} \mid \text{the until rank of } \varphi \text{ is less than or equal to } k\}$ to be the fragment of LTL in which all formulas have an until rank not greater than k. In Etessami and Wilke [2000], it is proved that for every $k \geq 1$, the logic LTL_k is strictly more expressive than LTL_{k-1}. The proof uses an EF game for LTL, which is the base of our quantitative version of an EF game, presented in Section 4.

2.5 Minsky Machines

Our undecidability proofs are reductions from undecidable problems for *Minsky machines.* A *Minsky machine* $\mathcal{M}$ consists of a finite set of instructions $\{I_1, \ldots, I_n\}$ operating on two counters denoted by C_1 and C_2. Each instruction I_j is of one of the following forms:

- *Increment*: $C_i := C_i + 1;$ go to S_j;
- *Zero test/decrement*: If $C_i = 0$ go to S_j^0; else $C_i := C_i - 1;$ go to S_j^1;
- *Halt*: halt;

where $i \in \{1, 2\}$ and $S_j, S_j^0, S_j^1 \subseteq \{I_1, \ldots, I_n\}$. Without loss of generality, we assume that I_n is the only halting instruction.

A *configuration* of $\mathcal{M}$ is a triple $(J, c, d) \in \{I_1, \ldots, I_n\} \times \mathbb{N} \times \mathbb{N}$, where J indicates the current instruction, and c and d are the current values of the counters C_1 and C_2, respectively. We define a transition relation over the set of all configurations of $\mathcal{M}$, denoted by $\Rightarrow$, as follows: $(I_j, c_0, c_1) \Rightarrow (I_k, c_0', c_1')$ if, and only if, there exists $i \in \{0, 1\}$ such that one of the following conditions holds:

- I_j is an increment instruction for C_{i+1}, $c_i' = c_i + 1$, $c_{1-i}' = c_{1-i}$, and $I_k \in S_j$.
- I_j is a zero test/decrement instruction for C_{i+1}, $c_i = c_i' = 0$, $c_{1-i}' = c_{1-i}$, and $I_k \in S_j^0$.
- I_j is a zero test/decrement instruction for C_{i+1}, $c_i' = c_i - 1 \geq 0$, $c_{1-i}' = c_{1-i}$, and $I_k \in S_j^1$.

A *computation* of $\mathcal{M}$ is a finite or infinite sequence $(\gamma_i)_{i \geq 0}$ of configurations such that $\gamma_0 = (I_1, 0, 0)$ and $\gamma_i \Rightarrow \gamma_{i+1}$ for every $i \geq 0$. A computation is called *halting* if it contains some configuration containing I_n. An infinite computation is called *recurring* if it contains infinitely many configurations containing I_1. The *halting problem* for Minsky machines asks, given a Minsky machine $\mathcal{M}$, whether there exists a halting computation of $\mathcal{M}$. The *recurrent state problem* for Minsky machines asks, given a Minsky maching $\mathcal{M}$, whether there exists a recurring computation of $\mathcal{M}$. Both problems are undecidable: the halting problem is Σ_1^0-complete [Minsky 1961], and the recurrent state problem is Σ_1^1-complete [Alur and Henzinger 1994].

3 MTL AND TPTL

Metric Temporal Logic. Let $\mathbb{P}$ be a countably infinite set of *atomic propositions.* MTL extends LTL in that the *until* modality is annotated with some interval over $\mathbb{Z}$. More precisely, the set of MTL formulas is defined by the following grammar:

$$\varphi ::= \mathrm{true} \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \mathrm{U}_I \varphi,$$

where $p \in \mathbb{P}$ and $I \subseteq \mathbb{Z}$ is an interval with endpoints in $\mathbb{Z} \cup \{-\infty, +\infty\}$.

Formulas in MTL are interpreted over data words. Let $w = (p_0, d_0)(p_1, d_1) \ldots$ be a data word over $\mathbb{P}$, and let i be a position in w. We define the *satisfaction relation for MTL*, denoted by $\models_{\mathrm{MTL}}$, inductively as follows:

- $(w, i) \models_{\mathrm{MTL}}$ true.
- $(w, i) \models_{\mathrm{MTL}} p$ if, and only if, $p = p_i$.

- $(w, i) \models_{\mathrm{MTL}} \neg\varphi$ if, and only if, $(w, i) \not\models_{\mathrm{MTL}} \varphi$.
- $(w, i) \models_{\mathrm{MTL}} \varphi_1 \wedge \varphi_2$ if, and only if, $(w, i) \models_{\mathrm{MTL}} \varphi_1$ and $(w, i) \models_{\mathrm{MTL}} \varphi_2$.
- $(w, i) \models_{\mathrm{MTL}} \varphi_1 \mathrm{U}_I \varphi_2$ if, and only if, there exists a position $j > i$ in w such that $(w, j) \models_{\mathrm{MTL}} \varphi_2$, $d_j - d_i \in I$, and for all positions k with $i < k < j$, $(w, k) \models_{\mathrm{MTL}} \varphi_1$.

We say that a data word *satisfies* an MTL formula φ, written $w \models_{\mathrm{MTL}} \varphi$, if $(w, 0) \models_{\mathrm{MTL}} \varphi$. We use the same syntactic abbreviations as for LTL, where every temporal operator is annotated with an interval over $\mathbb{Z}$—for example, $\mathrm{F}_I\varphi := \mathrm{true}\mathrm{U}_I\varphi$ and $\mathrm{X}_I\varphi := \mathrm{false}\mathrm{U}_I\varphi$. We may also use pseudo-arithmetic expressions of the form $x \sim c$, where $\sim \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{Z}$, to annotate the temporal modalities. For instance, we may write $\mathrm{F}_{=2}p$ as an alternative to $\mathrm{F}_{[2,2]}p$. If $I = \mathbb{Z}$, then we may omit the interval I on U_I.

The finitary and infinitary SAT for MTL is defined as for LTL.

Some of our undecidability results already hold for fragments of MTL, which we define in the following. We write MTL(F, X) to denote the set of MTL formulas in which the only temporal modalities allowed are F and X; likewise, we write MTL(F) to denote the fragment of MTL(F, X) in which even the usage of the X modality is not allowed. We sometimes refer to MTL(F, X) as the *unary* fragment of MTL. We write pMTL to denote the set of *pure* MTL formulas in which no propositional variables from $\mathbb{P}$ are used.

We define the *size* of an MTL formula φ, denoted by $|\varphi|$, as the number of symbols occurring in φ. We assume that all integer constants occurring in φ are encoded in binary.

Timed Propositional Temporal Logic. Let $\mathbb{P}$ be a countably infinite set of *atomic propositions.* Let V be a countably infinite set of *register variables.* Formulas of TPTL are built by the following grammar:

$$\varphi ::= \mathrm{true} \mid p \mid x \sim c \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \mathrm{U} \varphi \mid x.\varphi,$$

where $p \in \mathbb{P}$, $x \in V$, $c \in \mathbb{Z}$, and $\sim \in \{<, \leq, =, \geq, >\}$. We may also use formulas of the form $x \in I$ as abbreviation for conjunctions of constraints—for example, we may write $x \in [a, b]$ for $x \geq a \wedge x \leq b$.

A *register valuation* ν is a function from V to $\mathbb{N}$. Given a register valuation ν, a data value $d \in \mathbb{Z}$, and a variable $x \in V$, we define the register valuation $\nu[x \mapsto d]$ by $(\nu[x \mapsto d])(y) = \nu(y)$ for every $y \in V\backslash\{x\}$, and $(\nu[x \mapsto d])(x) = d$. Let $w = (p_0, d_0)(p_1, d_1)\cdots$ be a data word, let ν be a register valuation, and let i be a position in w. The satisfaction relation for TPTL, denoted by $\models_{\mathrm{TPTL}}$, is defined inductively as follows:

- $(w, i, \nu) \models_{\mathrm{TPTL}} \mathrm{true}$.
- $(w, i, \nu) \models_{\mathrm{TPTL}} p$ if, and only if, $p = p_i$.
- $(w, i, \nu) \models_{\mathrm{TPTL}} \neg\varphi$ if, and only if, $(w, i, \nu) \not\models_{\mathrm{TPTL}} \varphi$.
- $(w, i, \nu) \models_{\mathrm{TPTL}} \varphi_1 \wedge \varphi_2$ if, and only if, $(w, i, \nu) \models_{\mathrm{TPTL}} \varphi_1$ and $(w, i, \nu) \models_{\mathrm{TPTL}} \varphi_2$.
- $(w, i, \nu) \models_{\mathrm{TPTL}} \varphi_1 \mathrm{U} \varphi_2$ if, and only if, there exists a position $j > i$ in w such that $(w, j, \nu) \models_{\mathrm{TPTL}} \varphi_2$, and for all positions k with $i < k < j$, $(w, k, \nu) \models_{\mathrm{TPTL}} \varphi_1$.
- $(w, i, \nu) \models_{\mathrm{TPTL}} x \sim c$ if, and only if, $d_i - \nu(x) \sim c$.
- $(w, i, \nu) \models_{\mathrm{TPTL}} x.\varphi$ if, and only if, $(w, i, \nu[x \mapsto d_i]) \models_{\mathrm{TPTL}} \varphi$.

Intuitively, $x.\varphi$, means that x is *updated* to the data value at the current position of the data word, and atomic formulas of the form $x \sim c$, called *register tests*, require that, compared to the last time that x was updated, the data value has increased or decreased by a value satisfying $\sim c$. We say that a data word w satisfies a TPTL formula φ, written $w \models_{\mathrm{TPTL}} \varphi$, if $(w, 0, \mathbf{0}_w) \models_{\mathrm{TPTL}} \varphi$, where $\mathbf{0}_w$ denotes the valuation that maps all register variables to the initial data value d_0 of w.

The finitary and infinitary SAT for TPTL is defined as for LTL. We use the same syntactic abbreviations as for LTL. We define the fragments TPTL(F, X), TPTL(F), and pTPTL like the corresponding fragments of MTL. Additionally, we define FLTL to be the subset of TPTL formulas φ where all register tests are of the form $x = 0$. Further, for every $r \in \mathbb{N}$, we define the r-register fragment of TPTL, denoted by TPTL^r, as the fragment of TPTL in which at most r different register variables occur; similarly for fragments of TPTL.

4 RELATIVE EXPRESSIVENESS OF MTL AND TPTL

In this section, we investigate the relative expressiveness of MTL and TPTL when they are evaluated over data words. The following result is folklore.

Proposition 4.1. *TPTL^1 is at least as expressive as MTL.*

The proof of this proposition is based on the easy observation that an MTL formula of the form $\varphi \text{U}_{[a,b]} \psi$ can be transformed into the equivalent TPTL^1 formula $x.(\varphi \text{U}(\psi \wedge a \leq x \wedge x \leq b))$. It is, however, not so clear whether every TPTL formula can be translated into an equivalent MTL formula. It turns out that the answer to this question depends on the domain of the data words: for monotonic data words, Alur and Henzinger [1993] proved that MTL and TPTL are equally expressive. For *timed words* (i.e., monotonic data words over the non-negative real numbers), Bouyer et al. [2010] showed that TPTL is strictly more expressive than MTL.

The main result of this section is that, for data words, TPTL is strictly more expressive than MTL. More specifically, we prove that the following simple class of pure data words

$$\mathbb{E}_2 := \{w \in \mathbb{N}^\omega \mid w = d_0\, d_1\, d_2 \dots \text{ such that } d_0 = d_2\} \tag{1}$$

cannot be defined in MTL, whereas it can be defined in TPTL^1 (actually even in FLTL^1), namely by the formula

$$x.\text{XX}(x = 0). \tag{2}$$

As a main tool for proving this result, we introduce a quantitative version of EF games for LTL [Etessami and Wilke 2000], tailored for MTL.

In the remainder of this section, we fix a *finite* set $\mathbb{P}$ of propositional variables. If no confusion can arise, we omit the label in $\models_{\text{MTL}}$ and simply write $\models$ for the satisfaction relation for MTL.

Let $w = (p_0, d_0)(p_1, d_1)\dots$ be a data word, and let i be a position in w. We define $w[i:]$ to be the suffix of w starting in position i—that is, $w[i:] := (p_i, d_i)(p_{i+1}, d_{i+1})\dots$.

Let $w = (p_0, d_0)(p_1, d_1)(p_2, d_2)\dots$ be a finite or infinite data word, and let $r \in \mathbb{N}$. We define

$$w_{+r} := (p_0, d_0 + r)(p_1, d_1 + r)(p_2, d_2 + r)\dots.$$

If additionally w is finite, then we define the finite data word w^k_{+r} (where $k \in \mathbb{N}$) and the infinite data word w^ω_{+r}, respectively, by

$$w^k_{+r} := w w_{+r} w_{+2r} \dots w_{+(k-1)r} \quad \text{and} \quad w^\omega_{+r} := w w_{+r} w_{+2r} w_{+3r} \dots.$$

4.1 The EF Game for MTL

First we give the definition of the *until rank* of an MTL formula. The until rank of an MTL formula φ, denoted by $\text{rank}(\varphi)$, is defined inductively on the structure of φ:

- If φ is true or $p \in \mathbb{P}$, then $\text{rank}(\varphi) := 0$.
- If φ is $\neg\varphi_1$, then $\text{rank}(\varphi) := \text{rank}(\varphi_1)$.
- If φ is $\varphi_1 \wedge \varphi_2$, then $\text{rank}(\varphi) := \max\{\text{rank}(\varphi_1), \text{rank}(\varphi_2)\}$.
- If φ is $\varphi_1 \text{U}_I \varphi_2$, then $\text{rank}(\varphi) := \max\{\text{rank}(\varphi_1), \text{rank}(\varphi_2)\} + 1$.

Let $S \subseteq \mathbb{Z}$, and let $k \in \mathbb{N}$. We define

$$\begin{aligned}\mathrm{MTL}^S &:= \{\varphi \in \mathrm{MTL} \mid \text{all constants occurring in } \varphi \text{ are in } S\},\\ \mathrm{MTL}_k &:= \{\varphi \in \mathrm{MTL} \mid \mathrm{rank}(\varphi) \leq k\},\\ \mathrm{MTL}_k^S &:= \mathrm{MTL}^S \cap \mathrm{MTL}_k.\end{aligned}$$

Note that $\mathrm{MTL} = \bigcup_{k\in\mathbb{N}} \bigcup_{S\subseteq\mathbb{Z}} \mathrm{MTL}_k^S$. The following lemma is important for the proofs of Theorems 4.3 and 4.4.

Lemma 4.2. *For every finite set $S \subseteq \mathbb{Z}$ and every $k \in \mathbb{N}$, there are only finitely many formulas in MTL_k^S up to equivalence.*

Proof. Let $S \subseteq \mathbb{Z}$ be finite. We prove the lemma by induction on k. For the induction base, note that MTL_0^S is the set of all propositional formulas over $\mathbb{P}$, which is finite up to equivalence as $\mathbb{P}$ is finite. For the induction step, suppose the claim holds for k. The number of distinct intervals I that can be used to annotate the *until* modality in formulas in MTL_{k+1}^S is finite. Hence, there are only finitely many formulas up to equivalence in the set

$$\left\{\varphi_1 \, \mathrm{U}_I \, \varphi_2 \mid \varphi_1, \varphi_2 \in \mathrm{MTL}_k^S\right\} \cup \mathrm{MTL}_k^S. \tag{3}$$

The formulas obtained by using Boolean connectives on the formulas in (3) are also finitely many up to equivalence. Hence, the claim holds for $k + 1$. □

Let $S \subseteq \mathbb{Z}$ and $k \in \mathbb{N}$. Let w_0, w_1 be two data words, and let $i_0, i_1 \geq 0$ be two positions in w_0 and w_1, respectively. We say that $w_0[i_0:]$ and $w_1[i_1:]$ are *MTL_k^S-equivalent*, written $(w_0, i_0) \equiv_k^S (w_1, i_1)$, if, for every $\varphi \in \mathrm{MTL}_k^S$, $(w_0, i_0) \models \varphi$ if, and only if, $(w_1, i_1) \models \varphi$. We write $w_0 \equiv_k^S w_1$ if $(w_0, 0) \equiv_k^S (w_1, 0)$.

Now let $S \subseteq \mathbb{Z}$ be a finite set, and let $c_1, \ldots, c_n$ be the sequence of all numbers in S such that $c_i < c_{i+1}$ for all $i \in \{1, \ldots, n-1\}$. We define an equivalence relation $\simeq^S$ over $\mathbb{Z}$ as follows: for every $a, b \in \mathbb{Z}$, we have $a \simeq^S b$ if, and only if, $a = b$, or both a and b belong to one (but the same) of the intervals $(-\infty, c_1)$, (c_i, c_{i+1}) for some $i \in \{1, \ldots, n-1\}$, or $(c_n, +\infty)$. Note that for $S = \emptyset$, all numbers belong to the interval $\mathbb{Z}$.

Let w_0, w_1 be two data words. In the following, we use $p_{i,j}$ and $d_{i,j}$, where $i \in \{0, 1\}$ and j is a position in w_i, to denote the proposition and the data value, respectively, at position j of data word w_i.

Next, we define the EF game for MTL. The EF game is played by two players, called *Spoiler* and *Duplicator*, on w_0 and w_1, over a finite set $S \subseteq \mathbb{Z}$, and in k rounds. A game configuration is a pair (i_0, i_1), where i_0 is a position in w_0 and i_1 is a position in w_1. In each round of the game, if the current game configuration is (i_0, i_1):

- Spoiler starts the round by picking a data word w_l for some $l \in \{0, 1\}$ and a position $j_l > i_l$ in the data word w_l.
- Duplicator picks a position $j_{1-l} > i_{1-l}$ in the other data word w_{1-l} such that
  —if $j_l = i_l + 1$, then $j_{1-l} = i_{1-l} + 1$, and
  —$(d_{0,j_0} - d_{0,i_0}) \simeq^S (d_{1,j_1} - d_{1,i_1})$.
- Then Spoiler chooses between one of the following two options:
  —Finishing the current round with configuration (j_0, j_1).
  —Picking another position $i_{1-l} < j'_{1-l} < j_{1-l}$ in w_{1-l}. In this case, Duplicator responds with a position $i_l < j'_l < j_l$ in w_l. Then the current round finishes with configuration (j'_0, j'_1).

We use $\mathbf{G}_k^S(w_0, i_0, w_1, i_1)$ to denote the k-round EF game for MTL over the finite set $S \subseteq \mathbb{Z}$ starting from position i_0 in w_0, and from position i_1 in w_1.

The *winning condition for Duplicator* is defined inductively on the number of rounds of the game. We say that Duplicator wins $\mathbf{G}_0^S(w_0, i_0, w_1, i_1)$ if $p_{0,i_0} = p_{1,i_1}$. Duplicator wins $\mathbf{G}_{k+1}^S(w_0, i_0, w_1, i_1)$ if she wins $\mathbf{G}_0^S(w_0, i_0, w_1, i_1)$ and at least one of the following conditions holds:

- i_0 is the last position of w_0, and i_1 is the last position of w_1—that is, Spoiler has no position to pick and thus cannot play according to the rules;
- for every choice of moves of Spoiler in the first round, Duplicator can respond according to the preceding rules, and she wins $\mathbf{G}_k^S(w_0, n_0, w_1, n_1)$, where the first round has finished with configuration (n_0, n_1).

If Duplicator cannot win the game, we say that Spoiler wins the game. We write $(w_0, i_0) \sim_k^S (w_1, i_1)$ if Duplicator wins $\mathbf{G}_k^S(w_0, i_0, w_1, i_1)$. It follows easily that if $(w_0, i_0) \sim_k^S (w_1, i_1)$, then for all $0 \le m < k$, we also have $(w_0, i_0) \sim_m^S (w_1, i_1)$. We will write $w_0 \sim_k^S w_1$ if $(w_0, 0) \sim_k^S (w_1, 0)$.

Next, we establish a crucial correspondence between satisfaction of formulas in MTL_k^S and Duplicator winning the k-round EF game on S.

Theorem 4.3. *$(w_0, i_0) \equiv_k^S (w_1, i_1)$ if, and only if, $(w_0, i_0) \sim_k^S (w_1, i_1)$.*

Proof. The proof is by induction on k. The induction base $k = 0$ is trivial. So assume that the claim holds for $k \in \mathbb{N}$. We show that it also holds for $k + 1$.

(only if) Assume that $(w_0, i_0) \equiv_{k+1}^S (w_1, i_1)$. We prove $(w_0, i_0) \sim_{k+1}^S (w_1, i_1)$. Without loss of generality, assume that Spoiler picks w_0 and a position $i_0' > i_0$ in w_0. For each $i_0 < j \le i_0'$, define

$$\varphi_j' := \bigwedge \left\{ \varphi \in \mathrm{MTL}_k^S \mid (w_0, j) \models \varphi \right\}.$$

By Lemma 4.2, there exists an MTL_k^S formula φ_j that is equivalent to φ_j'. Define $d := d_{0,i_0'} - d_{0,i_0}$ and the MTL_{k+1}^S formula

$$\varphi := \left(\bigvee_{i_0 < j < i_0'} \varphi_j \right) \mathrm{U}_I \, \varphi_{i_0'},$$

where

$$I := \begin{cases} [d, d] & \text{if } d \in S, \\ (-\infty, a) & \text{if } d < a \text{ and } a \text{ is the smallest number in } S, \\ (b, +\infty) & \text{if } d > b \text{ and } b \text{ is the largest number in } S, \\ (a, b) & \text{if } a, b \in S, a < d < b \text{ and there is no } c \in S \text{ such that } a < c < b, \\ \mathbb{Z} & \text{if } S = \emptyset. \end{cases}$$

Clearly, $(w_0, i_0) \models \varphi$. By assumption, we have $(w_1, i_1) \models \varphi$. Hence, there exists $i_1' > i_1$ such that $(w_1, i_1') \models \varphi_{i_0'}$, $d_{1,i_1'} - d_{1,i_1} \in I$, and for all $i_1 < i_1'' < i_1'$, $(w_1, i_1'') \models \bigvee_{i_0 < j < i_0'} \varphi_j$. Let Duplicator respond with position i_1' in w_1. There are now two cases to consider. First, assume that Spoiler finishes the current round with configuration (i_0', i_1'). By $(w_0, i_0') \models \varphi_{i_0'}$, $(w_1, i_1') \models \varphi_{i_0'}$, and by definition of $\varphi_{i_0'}$ ($\varphi_{i_0'}$ is equivalent to the conjunction of all MTL_k^S formulas that hold in position i_0' in w_0), we obtain $(w_0, i_0') \equiv_k^S (w_1, i_1')$. By induction hypothesis, $(w_0, i_0') \sim_k^S (w_1, i_1')$. Second, assume that Spoiler picks a position $i_1 < i_1'' < i_1'$ in w_1. By $(w_1, i_1'') \models \bigvee_{i_0 < j < i_0'} \varphi_j$, there exists a position $i_0 < i_0'' < i_0'$ such that $(w_1, i_1'') \models \varphi_{i_0''}$. Let Duplicator respond with position i_0'' in w_0. The next round starts from configuration (i_0'', i_1''). Similarly to the other case, we can conclude from the definition of $\varphi_{i_0''}$ that $(w_0, i_0'') \equiv_k^S (w_1, i_1'')$. By induction hypothesis, we have $(w_0, i_0'') \sim_k^S (w_1, i_1'')$. Hence, in both cases, Duplicator wins the remaining k rounds. Finally, from $(w_0, i_0) \equiv_{k+1}^S (w_1, i_1)$, we obtain $p_{0,i_0} = p_{1,i_1}$. Hence, Duplicator wins $\mathbf{G}_{k+1}^S(w_0, i_0, w_1, i_1)$. Thus, $(w_0, i_0) \sim_{k+1}^S (w_1, i_1)$.

(if) Assume that $(w_0, i_0) \sim^S_{k+1} (w_1, i_1)$. We prove that $(w_0, i_0) \equiv^S_{k+1} (w_1, i_1)$. Let $\varphi \in \text{MTL}^S_{k+1}$. There are several cases to consider, and we prove the most interesting case here. The other cases are similar but simpler.

Let φ be of the form $\varphi_1 \text{U}_I \varphi_2$, where $\varphi_1, \varphi_2 \in \text{MTL}^S_k$, let $(w_0, i_0) \models \varphi$, and let Spoiler pick position i_0' in w_0 such that $(w_0, i_0') \models \varphi_2$, $d_{0,i_0'} - d_{0,i_0} \in I$, and $(w, i_0'') \models \varphi_1$ for all $i_0 < i_0'' < i_0'$. Since Duplicator can win the game, she can pick some position $i_1' > i_1$ in w_1 such that $(d_{0,i_0'} - d_{0,i_0}) \simeq^S (d_{1,i_1'} - d_{1,i_1})$. Recall that Spoiler has two options, and by assumption, Duplicator has a winning strategy for both cases: if Spoiler finishes the current round, it ends with configuration (i_0', i_1'), then Duplicator can win the k-round game starting from (i_0', i_1')—for instance, $(w_0, i_0') \sim^S_k (w_1, i_1')$. By induction hypothesis, $(w_0, i_0') \equiv^S_k (w_1, i_1')$. Thus, $(w_1, i_1') \models \varphi_2$. Otherwise, for instance, Spoiler picks a position $i_1 < i_1'' < i_1'$ in w_1, then Duplicator can respond with a position $i_0 < i_0'' < i_0'$ so that she can win the k-round game starting from (i_0'', i_1'')—for instance, $(w_0, i_0'') \sim^S_k (w_1, i_1'')$. By induction hypothesis, $(w_0, i_0'') \equiv^S_k (w_1, i_1'')$, and hence $(w_1, i_1'') \models \varphi_1$. Since Spoiler can pick any arbitrary position between i_1 and i_1', we obtain $(w_1, i_1'') \models \varphi_1$ for every $i_1 < i_1'' < i_1'$. Adding $(d_{0,i_0'} - d_{0,i_0}) \simeq^S (d_{1,i_1'} - d_{1,i_1})$ and $(w_1, i_1') \models \varphi_2$, we have $(w_1, i_1) \models \varphi_1 \text{U}_I \varphi_2$. Hence, $(w_0, i_0) \equiv^S_{k+1} (w_1, i_1)$. □

Theorem 4.4. *Let $\mathbb{C}$ be a class of data words. The following two statements are equivalent:*

(1) $\mathbb{C}$ is not definable in MTL.
(2) For every finite set $S \subseteq \mathbb{Z}$ and every $k \in \mathbb{N}$, there exist two data words $w_0 \in \mathbb{C}$ and $w_1 \notin \mathbb{C}$ such that $w_0 \sim^S_k w_1$.

Proof. ***(1. ⇒ 2.)*** We prove the contraposition: assume that the second claim does not hold. We prove that $\mathbb{C}$ is definable in MTL. If the second claim does not hold, then there exist a finite set $S \subseteq \mathbb{Z}$ and $k \in \mathbb{N}$ such that $w_0 \in \mathbb{C}$ implies $w_1 \in \mathbb{C}$ if $w_0 \sim^S_k w_1$ for every pair of data words w_0, w_1. By Theorem 4.3,

$$w_0 \in \mathbb{C} \text{ implies } w_1 \in \mathbb{C} \text{ if } w_0 \equiv^S_k w_1. \tag{4}$$

For a data word w, we define

$$\varphi'_w := \bigwedge \left\{\varphi \in \text{MTL}^S_k \mid w \models \varphi\right\}.$$

We further define

$$\psi' := \bigvee_{w \in \mathbb{C}} \varphi'_w.$$

By Lemma 4.2, there exist MTL^S_k formulas φ_w and ψ, respectively, such that $\varphi_w \equiv \varphi'_w$ and $\psi \equiv \psi'$.

We prove that $w \models \psi$ if, and only if, $w \in \mathbb{C}$, for all data words w. Note that this proves that $\mathbb{C}$ is definable in MTL. The direction from right to left is trivial. For the other direction, assume that $w \models \psi$. Then there exists a data word u such that $u \in \mathbb{C}$ and $w \models \varphi_u$. This implies that $w \equiv^S_k u$. From (4) and $u \in \mathbb{C}$, we obtain $w \in \mathbb{C}$.

(2. ⇒ 1.) Towards contradiction, suppose that $\mathbb{C}$ is definable in MTL. Hence, there exist some finite set $S \subseteq \mathbb{Z}$, $k \in \mathbb{N}$, and $\psi \in \text{MTL}^S_k$ such that

$$w \in \mathbb{C} \text{ if, and only if, } w \models \psi, \tag{5}$$

for all data words w. By assumption, there exist two data words w_0, w_1 such that $w_0 \in \mathbb{C}$, $w_1 \notin \mathbb{C}$, and $w_0 \sim^S_k w_1$. By (5), $w_0 \models \psi$ and $w_1 \not\models \psi$. By Theorem 4.3, $w_0 \equiv^S_k w_1$, and thus $w_1 \models \psi$, a contradiction. □

Before we come to our first main result, we prove two auxiliary lemmas that also serve as a warm-up for using the EF game.

Table 1. Six Possible Cases of Moves of Spoiler and Duplicator in the First Round of the k-Round Game on Data Words w_0 and w_1 from the Proof of Lemma 4.6

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
Move 1	$(\underline{1}, 1)$ or $(1, \underline{1})$	$(\underline{i}, i+1)$ $2 \le i < k$	$(1, \underline{2})$	$(i-1, \underline{i})$ $3 \le i \le k$	$(\underline{i}, i+1)$ $k \le i$	$(i-1, \underline{i})$ $k < i$
Move 2	Not possible	$(1, \underline{1})$ or $(j-1, \underline{j})$ $1 < j < i+1$	Not possible	$(\underline{j}, j+1)$ $1 \le j < i-1$	$(1, \underline{1})$ or $(j-1, \underline{j})$ $1 < j < i+1$	$(\underline{j}, j+1)$ $1 \le j < i-1$

We use (i_0, i_1) to denote that i_0 is a position in w_0 and i_1 is a position in w_1. We underline i_j to denote that Spoiler picks position i_j in w_j, and Duplicator responds with i_{1-j} in w_{1-j}.

Lemma 4.5. *Let $S \subseteq \mathbb{Z}$ be a finite set, and let w_0, w_1 be two data words such that $|w_0| = |w_1|$. If for every $i \ge 0$, $p_{0,i} = p_{1,i}$ and for every $j > i \ge 0$, $(d_{0,j} - d_{0,i}) \simeq^S (d_{1,j} - d_{1,i})$, then $w_0 \sim_k^S w_1$ for every $k \in \mathbb{N}$.*

Proof. The proof is straightforward: in each round, Duplicator can always respond with the same position that Spoiler picks in the other data word. □

Intuitively, the preceding lemma states that, if we are only allowed to use the constants from S, it is not the precise data values of a data word that matter, but in which interval I the differences of data values lie, where I can only be defined using the constants from S.

Lemma 4.6. *Let w be a finite or infinite pure data word, let $S \subseteq \mathbb{Z}$ be a finite set, and let $r \in \mathbb{N}$ be such that r is greater than the maximum value in S. Then $d_{+r}^k \cdot w_{+m} \sim_k^S d_{+r}^{k+1} \cdot w_{+m}$ for every $d \in \mathbb{N}$, $k \ge 1$ and $m > d + (k+1)r$.*

Proof. Let $w_0 = d_{+r}^k \cdot w_{+m}$, and let $w_1 = d_{+r}^{k+1} \cdot w_{+m}$—that is,

$$w_0 = (d,\ d+r,\ d+2r,\ d+3r,\ \ldots, d+(k-1)r) \cdot w_{+m}$$
$$w_1 = (d,\ d+r,\ d+2r,\ d+3r,\ \ldots, d+(k-1)r,\ d+kr) \cdot w_{+m}.$$

We explain the winning strategy of Duplicator for the first round. There are several cases, which are summarized in Table 1:

(1) Assume that Spoiler picks position 1 in w_0 or in w_1. Duplicator picks the same position in the respective other word. There cannot be a second move of Spoiler. Duplicator wins this round since $d_{0,1} - d_{0,0} = d_{1,1} - d_{1,0}$.
(2) Assume that Spoiler picks in w_0 a position $2 \le i < k$. Then Duplicator picks position $i+1$ in w_1. Note that $(d_{0,i} - d_{0,0}) \simeq^S (d_{1,i+1} - d_{1,0})$, and thus this move is according to the rules. If Spoiler takes as a second move position 1 in word w_1, then Duplicator responds with the same position in w_0; otherwise, if Spoiler takes as a second move a position $1 < j < i+1$ in w_1, Duplicator can respond with position $j-1$ in w_0.
(3) Assume that Spoiler picks position 2 in w_1. Then Duplicator can pick position 1 in w_0. By the choice of r, we again have $(d_{0,1} - d_{0,0}) \simeq^S (d_{1,2} - d_{1,0})$, and thus this move is according to the rules. Spoiler cannot take a second move in this case.
(4) Assume that Spoiler picks in w_1 a position $3 \le i \le k$. Duplicator can respond by picking position $i-1$ in w_0. Again, this response is according to the rules—that is, we have $(d_{0,i-1} - d_{0,0}) \simeq^S (d_{1,i} - d_{1,0})$. If Spoiler takes as a second move a position $1 \le j < i-1$ in w_0, then Duplicator can respond by picking position $j+1$ in w_1.
(5) Assume that Spoiler picks in w_0 a position $i \ge k$. Then Duplicator can follow the same strategy as in case (2).

Table 2. Six Possible Cases of Moves of Spoiler and Duplicator in the First Round of the k-Round Game on Data Words w_0 and w_1 from the Proof of Theorem 4.7

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
Move 1	$(\underline{1}, 1)$ or $(1, \underline{1})$	$(\underline{2}, 1)$	$(\underline{i}, i-1)$ $3 \le i \le k+3$	$(i+1, \underline{i})$ $2 \le i \le k+2$	$(\underline{i}, i-1)$ $k+4 \le i$	$(i+1, \underline{i})$ $k+3 \le i$
Move 2	Not possible	Not possible	$(j+1, \underline{j})$ $1 \le j \le i-2$	$(\underline{1}, 1)$ or $(\underline{j}, j-1)$ $2 \le j \le i$	$(j+1, \underline{j})$ $1 \le j \le i-2$	$(\underline{1}, 1)$ or $(\underline{j}, j-1)$ $2 \le j \le i$

We use (i_0, i_1) to denote that i_0 is a position in w_0 and i_1 is a position in w_1. We underline i_j to denote that Spoiler picks position i_j in w_j, and Duplicator responds with i_{1-j} in w_{1-j}.

(6) Assume that Spoiler picks in w_1 a position $i > k$. Then Duplicator can follow the same strategy as in case (4).

Hence, Duplicator can win the first round of the game. If $k = 1$, we are done. So assume that $k > 1$. In this case, note that after the first round, there are only two cases:

- The new configuration in the $(k-1)$-round game is $(1, 1)$. In this case, we use inductive reasoning to argue that Duplicator can win $\mathbf{G}^S_{k-1}(w_0, 1, w_1, 1)$.
- The new configuration is $(i, i+1)$ for some $1 \le i \le k$. Note that we have $(d_{0,n} - d_{0,j}) \simeq^S (d_{1,n+1} - d_{1,j+1})$ for all $n > j \ge i$ (for $n = k$ and $j = k-1$, this holds by the conditions on m). By Lemma 4.5, Duplicator can win $\mathbf{G}^S_{k-1}(w_0, i, w_1, i+1)$. □

Next, we prove the main result of this section.

Theorem 4.7. *The class $\mathbb{E}_2$ of data words defined in (1) is not definable in MTL.*

Proof. For every finite set $S \subseteq \mathbb{Z}$ and every $k \in \mathbb{N}$, we define two pure data words w_0 and w_1 such that $w_0 \notin \mathbb{E}_2$, $w_1 \in \mathbb{E}_2$, and $w_0 \sim^S_k w_1$. By Theorem 4.4, $\mathbb{E}_2$ is not definable in MTL.

Let $S \subseteq \mathbb{Z}$ be a finite set, let $k \in \mathbb{N}$, let $r > 0$ be such that all numbers in S are contained in $(-r, +r)$, and let $s > 2r$. Intuitively, we choose r in such a way that a jump of magnitude $\pm r$ between two data values cannot be detected by a formula in MTL^S, as all numbers in S are contained in $(-r, +r)$. We define two finite pure data words $v_0 := (s)(s-2r)^{k+3}_{+r}$ and $v_1 := (s)(s-r)^{k+2}_{+r}$—that is,

$$v_0 = s,\ s-2r,\ s-r,\ s,\ s+r,\ s+2r \ldots s+kr \qquad v_1 = s,\ s-r,\ s,\ s+r,\ s+2r \ldots s+kr. \quad (6)$$

Let w be a finite or infinite pure data word, and define $m \in \mathbb{N}$ such that $m > s + (k+1)r$; define

$$w_0 := v_0 \cdot w_{+m} \qquad w_1 := v_1 \cdot w_{+m}. \quad (7)$$

Clearly, $w_0 \notin \mathbb{E}_2$ and $w_1 \in \mathbb{E}_2$. We show that Duplicator can win $\mathbf{G}^S_k(w_0, 0, w_1, 0)$ for every $k \in \mathbb{N}$. The proof for $k = 0$ is obvious as there are no propositions. So let us assume that $k \ge 1$. We explain the winning strategy for Duplicator in the first round. There are six cases, which are summarized in Table 2:

(1) Spoiler picks position 1 in w_0 or w_1, and Duplicator responds with position 1 in the other data word. We have $(d_{0,1} - d_{0,0}) \simeq^S (d_{1,1} - d_{1,0})$ since $(d_{0,1} - d_{0,0}) = -2r$ and $(d_{1,1} - d_{1,0}) = -r$.
(2) Spoiler picks position 2 in w_0, and Duplicator responds with position 1 in w_1. We have $(d_{0,2} - d_{0,0}) \simeq^S (d_{1,1} - d_{1,0})$ since $(d_{0,2} - d_{0,0}) = (d_{1,1} - d_{1,0}) = -r$.
(3) Spoiler picks some position $3 \le i \le 3 + k$ in w_0, and Duplicator responds with position $i - 1$ in w_1. We have $(d_{0,i} - d_{0,0}) \simeq^S (d_{1,i-1} - d_{1,0})$ since $(d_{0,i} - d_{0,0}) = (d_{1,i-1} - d_{1,0}) = (i-3)r$. In the next move, if Spoiler picks some position $1 \le j \le i-2$ in w_1, Duplicator responds with position $j + 1$ in w_0.

(4) Spoiler picks a position $2 \leq i \leq 2+k$ in w_1, and Duplicator can respond with position $i+1$ in w_0. We have $(d_{0,i+1} - d_{0,0}) \simeq^S (d_{1,i} - d_{1,0})$ since $(d_{0,i+1} - d_{0,0}) = (d_{1,i} - d_{1,0}) = (i-2)r$. In the next move, if Spoiler picks position 1 in w_0, Duplicator responds with position 1 in w_1. If Spoiler picks a position $2 \leq j \leq i$ in w_0, Duplicator can respond with position $j-1$ in w_1.

(5) Assume that Spoiler picks in w_0 a position $i \geq k+4$. Then Duplicator can follow the same strategy as in case (2).

(6) Assume that Spoiler picks in w_1 a position $i \geq k+3$. Then Duplicator can follow the same strategy as in case (4).

After the first round, the new game configuration is either $(1,1)$ or $(i+1,i)$ for some $i \geq 1$. In the former case, we have

$$w_0[1:] = ((s-2r)^{k+3}_{+r}) \cdot w_{+m} \qquad w_1[1:] = ((s-r)^{k+2}_{+r}) \cdot w_{+m}.$$

By Lemma 4.6, we have

$$((s-2r)^{k+2}_{+r}) \cdot w_{+m} \sim^S_{k-1} w_0[1:],$$

and by Lemma 4.5,

$$((s-2r)^{k+2}_{+r}) \cdot w_{+m} \sim^S_{k-1} w_1[1:].$$

This implies that $w_0[1:] \sim^S_{k-1} w_1[1:]$. Hence, Duplicator wins $\mathbf{G}^S_{k-1}(w_0, 1, w_1, 1)$. If the new configuration is $(i+1,i)$ for some $i \geq 1$, it is obvious that Duplicator can win the game, as the two words are equal. Hence, if w is finite (infinite, respectively), we have proved that $\mathbb{E}_2$ is not definable in MTL over finite (infinite, respectively) pure data words. □

Recall from (2) that $\mathbb{E}_2$ is definable in TPTL^1. Using this, Theorem 4.7, and Proposition 4.1, we obtain the following.

Corollary 4.8. TPTL^1 *is strictly more expressive than* MTL.

Remark 4.1. In Bouyer et al. [2010], it is proved that the TPTL formula

$$x.\mathrm{F}(b \wedge \mathrm{F}(c \wedge x \leq 2))$$

is not definable in MTL over *timed words*—that is, monotonic data words over the non-negative real numbers. Using EF games for MTL, one can prove that this formula is neither definable in MTL over infinite data words: for every finite set $S \subseteq \mathbb{Z}$, let $r > 2$ be such that all numbers in S are contained in $(-r, +r)$, and let $s > 3r$. We define two infinite data words,

$$w_0 := (a,s)(c,s-3r)(b,s-2r)(c,s-r)((b,s+r)(c,s+2r))^{\omega}_{+2r},$$
$$w_1 := (a,s)(c,s-2r)(b,s-r)((c,s+r)(b,s+2r))^{\omega}_{+2r},$$

where a, b, c are atomic propositions. Clearly, $w_0 \models_{\mathrm{TPTL}} x.\mathrm{F}(b \wedge \mathrm{F}(c \wedge x \leq 2))$ and $w_1 \not\models_{\mathrm{TPTL}} x.\mathrm{F}(b \wedge \mathrm{F}(c \wedge x \leq 2))$. We leave it to the reader to verify that Duplicator can win the game $\mathbf{G}^S_k(w_0, 0, w_1, 0)$ for every $k \in \mathbb{N}$.

4.2 The MTL Definability Decision Problem

In the last section, we proved that TPTL is strictly more expressive than MTL. A natural problem that arises is the *MTL definability decision problem*: given a TPTL formula φ, is φ definable in MTL? We prove that this problem is undecidable by combining Theorem 4.7 and the undecidability of the SAT for TPTL [Alur and Henzinger 1994].

Theorem 4.9. *The problem whether a* TPTL *formula is definable in* MTL *is undecidable.*

Proof. We prove the claim for the class of infinite data words by a reduction from the recurrent state problem for Minsky machines. The proof for the class of finite data words can be done analogously using the Halting problem for Minsky machines. Let $\mathcal{M}$ be a Minsky machine. We construct a TPTL formula $\psi_\mathcal{M}$ such that $\psi_\mathcal{M}$ is definable in MTL if, and only if, $\mathcal{M}$ is a negative instance of the recurrent state problem.

Let $\varphi_{\text{recurrent}}$ be a pure TPTL formula such that $\mathcal{M}$ is a positive instance of the recurrent state problem if, and only if, $\varphi_{\text{recurrent}}$ is satisfiable; such a formula exists due to Theorem 5 in Alur and Henzinger [1994]. Define the TPTL formula

$$\psi_\mathcal{M} := x.\mathrm{XX}(x = 0) \wedge \mathrm{F}\,\varphi_{\text{recurrent}}.$$

By definition, $\psi_\mathcal{M}$ is definable in MTL if, and only if, the class $\mathbb{E}_\mathcal{M} := \{w \in \mathbb{N}^\omega \mid w \models_{\text{TPTL}} \psi_\mathcal{M}\}$ is definable in MTL. Let $\mathbb{E}_2$ be as in (1), and define $\mathbb{F}_\mathcal{M} = \{w \in \mathbb{N}^\omega \mid w \models_{\text{TPTL}} \varphi_{\text{recurrent}}\}$. Clearly, $\mathbb{E}_\mathcal{M} = \mathbb{E}_2 \cap \{uw \in \mathbb{N}^\omega \mid w \in \mathbb{F}_\mathcal{M}, |u| \geq 1\}$.

We distinguish two cases. First, assume that $\mathcal{M}$ is a negative instance of the recurrent state problem. Then $\varphi_{\text{recurrent}}$ is not satisfiable, so $\mathbb{F}_\mathcal{M} = \emptyset$ and thus $\mathbb{E}_\mathcal{M} = \emptyset$. Then $\mathbb{E}_\mathcal{M}$ is definable in MTL by the formula false. Second, assume that $\mathcal{M}$ is a positive instance of the recurrent state problem. We show that $\mathbb{E}_\mathcal{M}$ is not definable in MTL. Let $S \subseteq \mathbb{Z}$ be finite, and let $k \in \mathbb{N}$. Further pick $r > 0$ such that all numbers in S are contained in $(-r, +r)$, and let $s > 2r$. Define two finite pure data words v_0 and v_1 as in (6), for instance, by

$$v_0 = s,\ s-2r,\ s-r,\ s,\ s+r,\ s+2r \ldots s+kr \qquad v_1 = s,\ s-r,\ s,\ s+r,\ s+2r \ldots s+kr.$$

By assumption, there exists some pure infinite data word w such that $w \in \mathbb{F}_\mathcal{M}$. Let $m > s + (k+1)r$. Lemma 4.5, implies that $w_{+m} \in \mathbb{F}_\mathcal{M}$. Set

$$w_0 := v_0 \cdot w_{+m} \qquad w_1 := v_1 \cdot w_{+m},$$

and note that w_0 and w_1 are exactly as in (7). We can hence use the same lines of arguments to prove $w_0 \sim_k^S w_1$. Additionally, we clearly have $w_0 \notin \mathbb{E}_2$ and $w_1 \in \mathbb{E}_2$. Hence, $w_0 \notin \mathbb{E}_\mathcal{M}$ and $w_1 \in \mathbb{E}_\mathcal{M}$. By Theorem 4.4, this implies that $\mathbb{E}_\mathcal{M}$ is not definable in MTL. □

Remark 4.2. In Figueira and Segoufin [2009], it is proved that for every *counter automaton with incrementing errors* $\mathcal{A}$, there exists a formula $\varphi_\mathcal{A}$ in $\mathrm{FLTL}^1(\mathrm{F})$ such that $\mathcal{A}$ is a positive instance of the non-emptiness problem if, and only if, $\varphi_\mathcal{A}$ is satisfiable. Recall from Demri and Lazic [2009] and Figueira et al. [2011] that the non-emptiness problem for counter automata with incrementing errors is Ackermann-complete. We may thus use a proof method similar to the proof of Theorem 4.9 to show that the problem of whether a given FLTL formula φ is definable in MTL is Ackermann-hard. It is, however, an open problem whether this problem is decidable or not.

4.3 Further Applications of the EF Game for MTL

In Etessami and Wilke [2000], it is proved that the until rank hierarchy is strict for LTL. We prove that the same holds for MTL.

Proposition 4.10. *For every $k \in \mathbb{N}$, MTL_{k+1} is strictly more expressive than MTL_k.*

Proof. We define $\varphi_1 := (p \wedge \mathrm{X}\,p)$ for some $p \in \mathbb{P}$. For every $k \geq 1$, we define $\varphi_{k+1} := (p \wedge \mathrm{X}\,\varphi_k)$. Note that for every $k \geq 1$, $\varphi_k \in \mathrm{MTL}_k$. We prove that φ_k is not definable in MTL_{k-1}. By $\mathrm{MTL}_{k-1} = \bigcup_{\substack{S \subseteq \mathbb{Z} \\ S \text{ is finite}}} \mathrm{MTL}_{k-1}^S$, it is sufficient to prove that φ_k is not definable in MTL_{k-1}^S for every finite set $S \subseteq \mathbb{Z}$. Let $S \subseteq \mathbb{Z}$ be a finite set, and let $r > 0$ be such that all numbers in S are strictly smaller than r. Define two finite data words w_0, w_1, and an infinite data word w by

$$w_0 := (p, 0)_{+r}^{k+1} \qquad w_1 := (p, 0)_{+r}^{k} \qquad w := (q, (k+1)r)_{+r}^{\omega},$$

where $q \in \mathbb{P}$ satisfies $p \neq q$. One can use similar lines of arguments as in Lemma 4.6 to prove that $w_0 \sim_{k-1}^S w_1$ (for the finite word case) and that $(w_0 \cdot w) \sim_{k-1}^S (w_1 \cdot w)$ (for the infinite word case). This implies that φ_k is not definable in MTL_{k-1}^S. □

Recall from Example 2.1 that LTL with the strict semantics for the *until* modality has the same expressive power as LTL with the non-strict semantics and the *next* modality explicitly given in the syntax. For *monotonic* data words, one can conclude from the results in Alur and Henzinger [1993] that this also holds for MTL with the strict semantics as defined in Section 3 and for MTL with a non-strict semantics and the *next* modality explicitly given in the syntax, denoted by $\mathrm{MTL}(\mathrm{U}_{\mathrm{weak}}, \mathrm{X})$. For MTL interpreted over data words, it is easy to prove that for every $\mathrm{MTL}(\mathrm{U}_{\mathrm{weak}}, \mathrm{X})$ formula, there exists an equivalent MTL formula. We show here the quite surprising result that the reverse, however, does not hold.

Proposition 4.11. MTL *is strictly more expressive than* $\mathrm{MTL}(\mathrm{U}_{\mathrm{weak}}, \mathrm{X})$.

Sketch. For the proof, we use a variant of the EF game tailored to capture the non-strict semantics of U. In this variant of the EF game, roughly speaking, if the current round of the game starts in configuration (i_0, i_1), then Spoiler may pick a position $j_l \geq i_l$ in word w_l for some $l \in \{0, 1\}$ (instead of $j_l > i_l$). Using this game, one can prove that the MTL formula $\mathrm{F}_{=0}\mathrm{true}$ cannot be defined in $\mathrm{MTL}(\mathrm{U}_{\mathrm{weak}}, \mathrm{X})$. For this, let $S \subseteq \mathbb{Z}$ be a finite set, and let $s, r > 0$ be such that all numbers in S are contained in $(-r, +r)$ and $s > r$. Define the two infinite pure data words w_0 and w_1 by

$$w_0 := (s)(s + r, s)_{+2r}^{\omega} \qquad w_1 := (s)(s + 2r, s + r)_{+2r}^{\omega}.$$

Clearly, $w_0 \models \mathrm{F}_{=0}\mathrm{true}$ and $w_1 \not\models \mathrm{F}_{=0}\mathrm{true}$. We leave it to the reader to prove $w_0 \sim_k^S w_1$ for every $k \in \mathbb{N}$. □

The following proposition actually already holds for the corresponding fragments of LTL.

Proposition 4.12. $\mathrm{MTL}(\mathrm{F}, \mathrm{X})$ *is strictly more expressive than* $\mathrm{MTL}(\mathrm{F})$.

Sketch. For the proof, we use a variant of the EF game tailored for $\mathrm{MTL}(\mathrm{F})$, where, roughly speaking, Spoiler does not have a second move in the current round. Using this game, one can prove that the MTL formula $\mathrm{X}p$ cannot be defined in $\mathrm{MTL}(\mathrm{F})$. We leave the proof to the interested reader (hint: use $w_1 = (qp)^{\omega}$ and $w_2 = q(qp)^{\omega}$). □

4.4 The EF Game for TPTL

In Corollary 4.8, we proved that TPTL^1 is strictly more expressive than MTL. Recall that the class $\mathbb{E}_2$ defined in (1), which we used to separate TPTL^1 from MTL, can be defined by the formula $x.\mathrm{XX}(x = 0)$. Observe that this formula is in FLTL^1 on the one hand and in $\mathrm{TPTL}^1(\mathrm{F}, \mathrm{X})$ on the other hand. It is thus natural to investigate the relative expressiveness between MTL and FLTL, respectively, between MTL and $\mathrm{TPTL}(\mathrm{F}, \mathrm{X})$. For this, we define the EF game on TPTL.

The until rank of a TPTL formula φ, denoted by $\mathrm{rank}(\varphi)$, is defined as for MTL; we additionally define $\mathrm{rank}(x \sim c) := 0$ and $\mathrm{rank}(x.\varphi) := \mathrm{rank}(\varphi)$. Let $S \subseteq \mathbb{Z}$, and let $k \in \mathbb{N}$. We define

$$\mathrm{TPTL}^S := \{\varphi \in \mathrm{TPTL} \mid c \in S \text{ for every constraint } x \sim c \text{ in } \varphi\},$$
$$\mathrm{TPTL}_k := \{\varphi \in \mathrm{TPTL} \mid \mathrm{rank}(\varphi) \leq k\}.$$

Let $r \in \mathbb{N}$. Recall that we use TPTL^r to denote the fragment of TPTL that uses at most r register variables. We define

$$\mathrm{TPTL}^{r,S} := \mathrm{TPTL}^r \cap \mathrm{TPTL}^S,$$
$$\mathrm{TPTL}_k^{r,S} := \mathrm{TPTL}^r \cap \mathrm{TPTL}^S \cap \mathrm{TPTL}_k.$$

The proof for the following lemma can be done analogously to the proof of Lemma 4.2.

Lemma 4.13. *For every finite set $S \subseteq \mathbb{Z}$, every $r \in \mathbb{N}$, and every $k \in \mathbb{N}$, there are only finitely many formulas in* $\mathrm{TPTL}_k^{r,S}$ *up to equivalence.*

Let $S \subseteq \mathbb{Z}$ be finite, $r \in \mathbb{N}$, and $k \in \mathbb{N}$. Let w_0, w_1 be two data words, let $i_0, i_1 \geq 0$ be two positions in w_0 and w_1, respectively, and let ν_0, ν_1 be two register valuations. We say that $(w_0[i_0:], \nu_0)$ and $(w_1[i_1:], \nu_1)$ are $\mathrm{TPTL}_k^{r,S}$-equivalent, written $(w_0, i_0, \nu_0) \equiv_k^{r,S} (w_1, i_1, \nu_1)$, if for every $\varphi \in \mathrm{TPTL}_k^{r,S}$, $(w_0, i_0, \nu_0) \models_{\mathrm{TPTL}} \varphi$ if, and only if, $(w_1, i_1, \nu_1) \models_{\mathrm{TPTL}} \varphi$. We write $w_0 \equiv_k^{r,S} w_1$ if $(w_0, 0, \mathbf{0}_{w_0}) \equiv_k^{r,S} (w_1, 0, \mathbf{0}_{w_1})$ (recall that $\mathbf{0}_{w_i}$ denotes the register valuation assigning to every register variable the initial data value $d_{i,0}$).

Let ν be a register valuation over register variables $\{x_1, \ldots, x_r\}$, let $Y \subseteq \{x_1, \ldots, x_r\}$, and let $d \in \mathbb{N}$. We define the register valuation $\nu[Y \mapsto d]$ by $\nu[Y \mapsto d](x) = d$ if $x \in Y$, and $\nu[Y \mapsto d](x) = \nu(x)$ otherwise.

The EF game for TPTL is played by Spoiler and Duplicator, on two data words w_0 and w_1, on a finite set of register variables $\{x_1, \ldots, x_r\}$, and over a finite set $S \subseteq \mathbb{Z}$. A game configuration is a tuple (i_0, ν_0, i_1, ν_1), where ν_0, ν_1 are two register valuations, and $i_0, i_1 \geq 0$ are positions in w_0 and w_1, respectively. In each round of the game, if the current game configuration is (i_0, ν_0, i_1, ν_1):

- Spoiler picks a subset $Y \subseteq \{x_1, \ldots, x_r\}$ and sets $\nu_l' = \nu_l[Y \mapsto d_{l,i_l}]$ for all $l \in \{0, 1\}$. Informally, we say that Spoiler *freezes* the register variables $x_1, \ldots, x_r$.
- Spoiler picks a data word w_l for $l \in \{0, 1\}$ and a position $j_l > i_l$ in the data word w_l.
- Duplicator responds with some position $j_{1-l} > i_{1-l}$ in data word w_{1-l} satisfying $j_{1-l} = i_{1-l} + 1$ whenever $j_l = i_l + 1$.
- Then Spoiler chooses one of the following two options:
  - Spoiler finishes the current round; the new configuration is $(j_0, \nu_0', j_1, \nu_1')$.
  - Spoiler picks a new position $i_{1-l} < j_{1-l}' < j_{1-l}$ in w_{1-l}. Then Duplicator responds with a position $i_l < j_l' < j_l$ in w_l, and the current round is finished. The new configuration is $(j_0', \nu_0', j_1', \nu_1')$.

We use $\mathrm{TG}_k^{r,S}(w_0, i_0, \nu_0, w_1, i_1, \nu_1)$ to denote the k-round EF game for TPTL starting from the positions i_0, i_1 with the register valuations ν_0, ν_1 in data words w_0 and w_1, respectively, over the register variables set $\{x_1, \ldots, x_r\}$ and finite set S.

The *winning condition for Duplicator* is defined inductively. Duplicator wins $\mathrm{TG}_0^{r,S}(w_0, i_0, \nu_0, w_1, i_1, \nu_1)$ if

- $p_{0,i_0} = p_{1,i_1}$, and
- for all constraints $x \sim c$ with $x \in \{x_1, \ldots, x_r\}$ and $c \in S$, $(w_0, i_0, \nu_0) \models_{\mathrm{TPTL}} x \sim c$ if, and only if, $(w_1, i_1, \nu_1) \models_{\mathrm{TPTL}} x \sim c$.

Duplicator wins $\mathrm{TG}_{k+1}^{r,S}(w_0, i_0, \nu_0, w_1, i_1, \nu_1)$ if she wins $\mathrm{TG}_0^{r,S}(w_0, i_0, \nu_0, w_1, i_1, \nu_1)$ and at least one of the following conditions is satisfied:

- i_0 is the last position of w_0, and i_1 is the last position of w_1
- for every choice of moves of Spoiler in the first round, Duplicator can respond according to the rules and wins $\mathrm{TG}_k^{r,S}(w_0, n_0, \nu_0', w_1, n_1, \nu_1')$.

In the latter case, $(n_0, \nu_0', n_1, \nu_1')$ is the new configuration after the first round. If Duplicator cannot win the game, we say that Spoiler wins the game. We write $(w_0, i_0, \nu_0) \sim_k^{r,S} (w_1, i_1, \nu_1)$ if Duplicator wins $\mathrm{TG}_k^{r,S}(w_0, i_0, \nu_0, w_1, i_1, \nu_1)$. We write $w_0 \sim_k^{r,S} w_1$ if $(w_0, 0, \mathbf{0}_{w_0}) \sim_k^{r,S} (w_1, 0, \mathbf{0}_{w_1})$.

The proofs of the following two theorems are analogous to the proofs of Theorems 4.3 and 4.4, respectively.

Theorem 4.14. $(w_0, i_0, \nu_0) \equiv_k^{r,S} (w_1, i_1, \nu_1)$ *if, and only if,* $(w_0, i_0, \nu_0) \sim_k^{r,S} (w_1, i_1, \nu_1)$.

Theorem 4.15. *Let $\mathbb{C}$ be a class of data words. For every finite set $S \subseteq \mathbb{Z}$, every $r \in \mathbb{N}$, and every $k \in \mathbb{N}$, the following two statements are equivalent:*

(1) $\mathbb{C}$ is not definable in $\mathrm{TPTL}_k^{r,S}$.
(2) There exist two data words $w_0 \in \mathbb{C}$ and $w_1 \notin \mathbb{C}$ such that $w_0 \sim_k^{r,S} w_1$.

We are now ready to answer the question about the relative expressiveness of MTL and TPTL(F, X) and FLTL. We remark that for proving the following two results, one has to slightly change the definition of EF games for TPTL to suit to the fragments FLTL and TPTL(F, X) and such that an analogous version of Theorem 4.14 holds. The following result is not very surprising, as with FLTL, one can only compare the values of register variables with the current data value for (in)equality.

Proposition 4.16. *The* MTL *formula* $\mathrm{F}_{=1}\mathrm{true}$ *cannot be defined in* FLTL.

Sketch. Define the pure data words w_0 and w_1 by

$$w_0 := (s, s+1)_{+0}^{\omega} \qquad w_1 := (s, s+2)_{+0}^{\omega}.$$

We leave it to the reader to prove that $w_0 \sim_k^{r,S} w_1$ for every $r, k \in \mathbb{N}$. □

Proposition 4.17. *The* MTL *formula $\neg a\mathrm{U}b$ cannot be defined in* TPTL(F, X).

Sketch. This result already holds for LTL and unary LTL(F, X), and it can be proved using the words $w_0 = c(ba)^{\omega}$ and $w_1 = c(ab)^{\omega}$. □

As a corollary from the preceding results, we obtain the following.

Corollary 4.18.

(1) MTL and FLTL are incomparable.
(2) MTL and TPTL(F, X) are incomparable.
(3) FLTL and TPTL(F, X) are incomparable.

The following lemma is the TPTL-analogon to Lemma 4.5. The proof can be done in a similar way.

Lemma 4.19. *Let $S \subseteq \mathbb{Z}$ be finite, and let $r \in \mathbb{N}$. Let w_0, w_1 be two data words such that $|w_0| = |w_1|$, for every $i \geq 0, p_{0,i} = p_{1,i}$, and for every $j > i$, $(d_{0,j} - d_{0,i}) \simeq^S (d_{1,j} - d_{1,i})$. Then for every two register valuations ν_0 and ν_1, if, for every $i \geq 0$ and every constraint $x \sim c$ with $x \in \{x_1, \ldots, x_r\}$ and $c \in S$, we have $(w_0, i, \nu_0) \models_{\mathrm{TPTL}} x \sim c$ if, and only if, $(w_1, i, \nu_1) \models_{\mathrm{TPTL}} x \sim c$, then $(w_0, 0, \nu_0) \sim_k^{r,S} (w_1, 0, \nu_1)$.*

In the following, we prove that for TPTL, the number of used register variables has an impact on the expressive power of the logic.

Theorem 4.20. TPTL^2 *is strictly more expressive than* TPTL^1.

Proof. We show that the TPTL^2 formula

$$\varphi = x_1.\mathrm{X}(x_1 > 0 \wedge x_2.\mathrm{F}(x_1 > 0 \wedge x_2 < 0))$$

is not definable in TPTL^1. Intuitively, φ expresses that there exists a future position for which the data value is strictly between the first and the second data value. It is sufficient to show that φ is not definable in $\mathrm{TPTL}_k^{1,S}$ for every finite set $S \subseteq \mathbb{Z}$ and $k \in \mathbb{N}$.

We first prove that φ is not definable in TPTL^1 over infinite data words. Let $S \subseteq \mathbb{Z}$ be finite, and let $k \in \mathbb{N}$. Let $s, r > 0$ be such that all numbers in S are contained in $(-r, +r)$ and $s - kr > 0$. We define two pure infinite data words w_0 and w_1 by

$$w_0 := (s, s+2r)(s-kr)_{+r}^{k+2}(s+3r)_{+r}^{\omega} \qquad w_1 := (s, s+2r)(s-kr)_{+r}^{k+1}(s+3r)_{+r}^{\omega}.$$

Clearly, $w_0 \models_{\text{TPTL}} \varphi$ and $w_1 \not\models_{\text{TPTL}} \varphi$. We show that $w_0 \sim_k^{1,S} w_1$, which by Theorem 4.15 implies that φ is not definable in $\text{TPTL}_k^{1,S}$.

The claim clearly holds for $k = 0$. Let $k \geq 1$. The initial game configuration is $(0, s, 0, s)$. In the first round, if Spoiler picks some position $i \geq 1$ in w_l, where $l \in \{0, 1\}$, then Duplicator can respond with the same position i in w_{1-l}. In the next move, if Spoiler picks a position $0 < j < i$ in w_{1-l}, Duplicator can respond with the same position j in w_l. After the first round, the new game configuration is either (i, s, i, s) for some $i \geq 2$ or $(1, s, 1, s)$. In the former case, we obtain $(w_0, i, s) \sim_{k-1}^{1,S} (w_1, i, s)$ by Lemma 4.19. In the second case, we investigate the possible choices of Spoiler in the next round:

(1) Assume that Spoiler does not freeze x_1 to the data value $s + 2r$ in the current position. Then Duplicator can respond with the same position that Spoiler picks in the other data word. By Lemma 4.19, Duplicator can win the remaining rounds.
(2) Assume that Spoiler freezes x_1 to the data value $s + 2r$—that is, the configuration becomes $(1, s+2r, 1, s+2r)$. We show that $(w_0, 1, s+2r) \sim_{k-1}^{1,S} (w_1, 1, s+2r)$:
   (a) Spoiler picks position 2 in w_l for some $l \in \{0, 1\}$. Then Duplicator can respond with the same position in w_{1-l}. We show that Duplicator wins the remaining $k - 2$ rounds from the new configuration $(2, s+2r, 2, s+2r)$. This is equivalent to showing that Duplicator wins $\text{TG}_h^{1,S}(w_0', 0, s+2r, w_1', 0, s+2r)$, where $h = k - 2$ and

   $$w_0' := (s-(h+2)r)_{+r}^{h+4}(s+3r)_{+r}^{\omega} \qquad w_1' := (s-(h+2)r)_{+r}^{h+3}(s+3r)_{+r}^{\omega}.$$

   By induction on h, we obtain $(w_0', 0, s+2r) \sim_h^{1,S} (w_1', 0, s+2r)$.
   (b) Spoiler picks some position $i \geq 3$ in w_0. Then Duplicator can respond with position $i - 1$ in w_1. In the next move, if Spoiler picks a position $1 < j < i - 1$ in w_1, then Duplicator can respond with position $j + 1$ in w_0. After this round, the new configuration is $(i, s + 2r, i - 1, s + 2r)$ for some $i \geq 3$. By Lemma 4.19, we can infer that $(w_0, i, s+2r) \sim_{k-2}^{1,S} (w_1, i-1, s+2r)$.
   (c) Spoiler picks some position $i \geq 3$ in w_1. Then Duplicator can respond with position $i + 1$ in w_0. In the next move, if Spoiler picks position 2 in w_0, Duplicator can also respond with position 2 in w_1. If Spoiler picks some position $2 < j \leq i$ in w_0, then Duplicator can respond with position $j - 1$ in w_1. After this round, the new configuration is either $(2, s+2r, 2, s+2r)$, then by (a) we have $(w_0, 2, s+2r) \sim_{k-2}^{1,S} (w_1, 2, s+2r)$, or $(i+1, s+2r, i, s+2r)$ $(i \geq 2)$, then by Lemma 4.19, we have $(w_0, i+1, s+2r) \sim_{k-2}^{1,S} (w_1, i, s+2r)$.

To prove that φ is not definable in TPTL^1 over finite data words, let $S \subseteq \mathbb{Z}$ be a finite set, let $k \in \mathbb{N}$, and let $s, r > 0$ be defined as earlier. We define

$$w_0 := (s, s+2r)(s-kr)_{+r}^{k+2}(s+3r)_{+r}^{k} \qquad w_1 := (s, s+2r)(s-kr)_{+r}^{k+1}(s+3r)_{+r}^{k}.$$

Using similar arguments as for the infinite case, one may prove $w_0 \sim_k^{1,S} w_1$. □

It is open whether the hierarchy of relative expressiveness based on the number of allowed registers is strict—for instance, whether TPTL^{r+1} is strictly more expressive than TPTL^r for every

$r \in \mathbb{N}$. For the *path-checking problem* for TPTL (i.e., the problem of deciding whether $w \models_{\text{TPTL}} \varphi$ for a given TPTL formula and a given data word w), we recently proved that there is a difference between TPTL^1 and TPTL^2, but no difference between TPTL^r and TPTL^{r+1} for all $r \geq 2$: the path-checking problem for TPTL^1 is PTIME-complete, whereas the path-checking problem for TPTL^r is PSPACE-complete for all $r \geq 2$ [Feng et al. 2017].

We would like to conclude this section with the following result on the strict until hierarchy for TPTL. It can be proved analogously to the corresponding result for MTL, stated in Proposition 4.10.

Proposition 4.21. *For every $k \in \mathbb{N}$, TPTL$_{k+1}$ is strictly more expressive than TPTL$_k$.*

5 MODEL CHECKING ONE-COUNTER MACHINES

In this section, we investigate the model-checking problem for one-counter machines. Recall from Demri et al. [2010] that the model-checking problem for FLTL is undecidable both for finite and infinite data words, even if the FLTL formula only uses one register and only unary modalities. FLTL^1 is a fragment of TPTL^1, so one can infer the same results for TPTL^1. Motivated by the expressiveness results from the previous section, and some promising decidability results for the setting of *monotonic* data words [Laroussinie et al. 2002; Ouaknine and Worrell 2007; Schmitz and Schnoebelen 2011], we would like to study the model-checking problem for one-counter machines and MTL.

As a main result, we will prove that model checking one-counter machines against formulas in MTL is undecidable. This even holds for the fragment of MTL in which the only allowed modality is the finally modality.

Theorem 5.1. *Finitary model checking for* MTL(F) *is Σ_1^0-hard, and infinitary model checking for* MTL(F) *is Σ_1^1-hard.*

Proof. For Σ_1^0-hardness of the finitary model-checking problem for MTL, we reduce from the halting problem for Minsky machines: given a Minsky machine $\mathcal{M}$, we will construct a one-counter machine $\mathcal{A}$ and an MTL formula φ such that $\mathcal{M}$ is a positive instance of the halting problem if, and only if, $\mathcal{A} \models^*_{\text{MTL}} \varphi$.

Let $\mathcal{M} = \{I_1, \ldots, I_n\}$ be a Minsky machine. Recall that I_n is the (only) halting instruction. One can easily see that, without loss of generality, we may assume that $\mathcal{M}$ satisfies the following two conditions:

(i) The first instruction I_1 is an increment instruction on C_1.
(ii) If $\mathcal{M}$ has a halting computation, then the last instruction before the halting instruction I_n is an increment instruction.

These two conditions will be crucial for the correctness of the reduction.

The idea of the reduction is to encode computations of $\mathcal{M}$ by data words over the set of atomic propositions $\mathbb{P} = \mathbb{P}_{\text{inc}} \cup \mathbb{P}_{\text{zero}} \cup \mathbb{P}_{\text{dec}} \cup \mathbb{P}_{\text{halt}} \cup \mathbb{P}_{\text{dum}}$, where

- $\mathbb{P}_{\text{inc}} = \{\langle I_j, \text{inc}_i, I_k\rangle \mid I_j : C_i := C_i + 1;\ \text{go to } S_k;\}$
- $\mathbb{P}_{\text{zero}} = \{\langle I_j, \text{ifzero}_i, I_k\rangle \mid I_j : \text{If } C_i = 0 \text{ go to } S_k^0;\ \text{else } C_i := C_i - 1;\ \text{go to } S_m^1;\}$
- $\mathbb{P}_{\text{dec}} = \{\langle I_j, \text{dec}_i, I_m\rangle \mid I_j : \text{If } C_i = 0 \text{ go to } S_k^0;\ \text{else } C_i := C_i - 1;\ \text{go to } S_m^1;\}$
- $\mathbb{P}_{\text{halt}} = \{\langle I_n, \text{halt}\rangle\}$,

and $\mathbb{P}_{\text{dum}}$ is a set of dummy propositions defined in the following. A zero test/decrement instruction I_j gives rise to two kinds of propositions in $\mathbb{P}$: one proposition $\langle I_j, \text{ifzero}_i, I_k\rangle$, where $I_k \in S_j^0$, and one proposition $\langle I_j, \text{dec}_i, I_k\rangle$, , where $I_k \in S_j^1$. When encoding a computation of $\mathcal{M}$ as a data word over $\mathbb{P}$, we will use the first one to represent a successful zero test and the second one for a

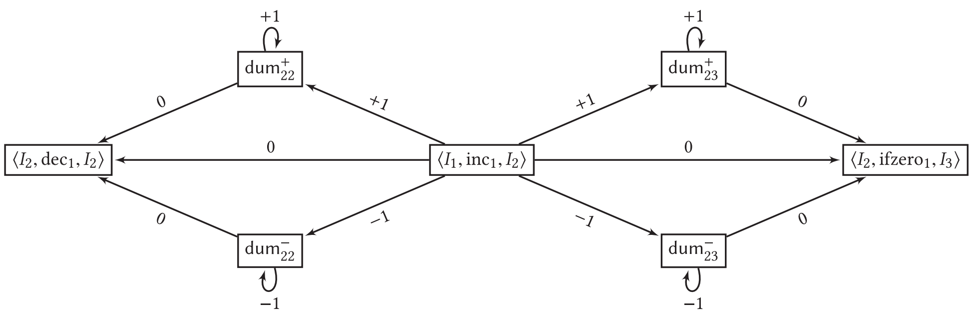
+1
dum$^+_{22}$
+1
dum$^+_{23}$
0
+1
+1
0
$\langle I_2, \text{dec}_1, I_2\rangle$
0
$\langle I_1, \text{inc}_1, I_2\rangle$
0
$\langle I_2, \text{ifzero}_1, I_3\rangle$
0
−1
−1
0
dum$^-_{22}$
dum$^-_{23}$
−1
−1

Fig. 1. Part of the one-counter machine $\mathcal{A}$ used for simulating a Minsky machine $\mathcal{M}$ with instructions like in (8).

successful decrement. For instance, assume that I_1 and I_2 are of the form

$$I_1 : C_1 := C_1 + 1;\ \text{go to } S_2; \quad I_2 : \text{If } C_1 = 0 \text{ go to } S_2^0;\ \text{else } C_1 := C_1 - 1;\ \text{go to } S_2^1; \tag{8}$$

where $S_2 = S_2^1 = \{I_2\}$ and $S_2^0 = \{I_3\}$. Then the computation

$$(I_1, 0, 0) \Rightarrow (I_2, 1, 0) \Rightarrow (I_2, 0, 0) \Rightarrow (I_3, 0, 0) \Rightarrow \ldots$$

will be encoded by the data word

$$(\langle I_1, \text{inc}_1, I_2\rangle, d_1)\,(\langle I_2, \text{dec}_1, I_2\rangle, d_2)\,(\langle I_2, \text{ifzero}_1, I_3\rangle, d_3)\ldots.$$

We explain in the following the conditions on the sequence of data values $d_1, d_2, d_3, \ldots$. Here, we only mention that the difference between two consecutive data values is not bounded *a priori*. We use the *dummy symbols* in $\mathbb{P}_{\text{dum}}$ to bridge any distances different from 0 between two consecutive positions in the data word. For instance, for bridging the distance of 2 between the two data values in the data word

$$(\langle I_3, \text{inc}_2, I_4\rangle, 3)\,(\langle I_4, \text{dec}_1, I_1\rangle, 1),$$

we add two dum$^-$ propositions:

$$(\langle I_3, \text{inc}_2, I_4\rangle, 3)\,(\text{dum}^-, 2)\,(\text{dum}^-, 1)(\langle I_4, \text{dec}_1, I_1\rangle, 1).$$

For technical reasons, we introduce two dummy symbols for each proposition in $\mathbb{P}_{\text{inc}} \cup \mathbb{P}_{\text{zero}} \cup \mathbb{P}_{\text{dec}}$, formally

$$\mathbb{P}_{\text{dum}} := \{\text{dum}^+_{jk}, \text{dum}^-_{jk} \mid \langle I_j, \cdot, I_k\rangle \in \mathbb{P}_{\text{inc}} \cup \mathbb{P}_{\text{zero}} \cup \mathbb{P}_{\text{dec}}\}.$$

The one-counter machine $\mathcal{A}$ is defined in such a way that it "generates" data words over $\mathbb{P}$ that correspond to computations of $\mathcal{M}$. More detailed, $\mathcal{A}$ generates a superset of such data words: every generated data word satisfies the *syntactical conditions* for being the encoding of a computation of $\mathcal{M}$. The *semantical conditions* are tested by the MTL formula φ defined in the following.

We define $\mathcal{A} = (Q, q_{\text{in}}, E, \lambda)$ as follows. Let $Q = \mathbb{P}$ and $q_{\text{in}} = \langle I_1, \text{inc}_1, I_2\rangle$. The transition relation E is the smallest set that contains for every pair $\langle I_i, \text{op}, I_j\rangle, \langle I_j, \text{op}', I_k\rangle \in \mathbb{P}$ the transitions

- $(\langle I_i, \text{op}, I_j\rangle, \langle I_j, \text{op}', I_k\rangle)$ with label 0,
- $(\langle I_i, \text{op}, I_j\rangle, \text{dum}^+_{ij})$ and $(\text{dum}^+_{ij}, \text{dum}^+_{ij})$ with label +1, $(\text{dum}^+_{ij}, \langle I_j, \text{op}', I_k\rangle)$ with label 0, and
- $(\langle I_i, \text{op}, I_j\rangle, \text{dum}^-_{ij})$ and $(\text{dum}^-_{ij}, \text{dum}^-_{ij})$ with label −1, and $(\text{dum}^-_{ij}, \langle I_j, \text{op}', I_k\rangle)$ with label 0.

For an example, we depict in Figure 1 the part of $\mathcal{A}$ corresponding to the two instructions given in (8).

Note that $\mathcal{A}$ does not impose any restrictions on the data values. It can be easily seen that all computations of $\mathcal{A}$ correspond to data words that encode all syntactical conditions for being a computation of $\mathcal{M}$.

Next, we define an MTL(F) formula φ such that all models of φ satisfy the semantical conditions for being a computation of $\mathcal{M}$. Specifically, we define φ such that it expresses conditions that ensure that no cheatings regarding zero tests and decrements can happen. The conditions heavily rely on restricted usage of data values, and we explain the idea in the following.

The general idea is to *uniquely match* each decrement of C_i in a data word w, shown by a proposition $(\cdot, \mathrm{dec}_i, \cdot)$, with a preceding increment of C_i, shown by $(\cdot, \mathrm{inc}_i, \cdot)$. The base idea is to require that every increment occurs in w with its own distinct data value. Then we say that a decrement on C_i at position j matches with a preceding increment on C_i at position $k < j$ if, and only if, it occurs in w with the same data value (i.e., $d_j = d_k$). We also require that there is no other decrement on C_i matching with the same increment—that is, the decrement at position j is the unique match with the increment at position k. This simple idea supports us in checking the correctness of the encoding of decrements and zero tests:

- For a decrement on C_i at position j to be correct, there must be some matching increment on C_i at some position $k < j$. This ensures that the current value of C_i is greater than zero.
- For a zero test on C_i at position k to be correct, for every increment on C_i at position $j < k$, there must be a decrement on C_i at some position $j < m < k$ that matches with the increment. This ensures that the current value of C_i is zero.

Before we explain in more detail the conditions for ensuring a correct encoding of decrements, we define some useful abbreviations. For this, let proj_2 denote the projection on the second component of a proposition in $\mathbb{P}$:

$$\mathrm{inc}(i) := \bigvee_{\substack{p \in \mathbb{P}_{\mathrm{inc}} \\ \mathrm{proj}_2(p) = \mathrm{inc}_i}} p \qquad \mathrm{dec}(i) := \bigvee_{\substack{p \in \mathbb{P}_{\mathrm{dec}} \\ \mathrm{proj}_2(p) = \mathrm{dec}_i}} p \qquad \mathrm{iz}(i) := \bigvee_{\substack{p \in \mathbb{P}_{\mathrm{zero}} \\ \mathrm{proj}_2(p) = \mathrm{ifzero}_i}} p$$
$$\mathrm{inc} := \mathrm{inc}(1) \vee \mathrm{inc}(2) \qquad \mathrm{dec} := \mathrm{dec}(1) \vee \mathrm{dec}(2) \qquad \mathrm{iz} := \mathrm{iz}(1) \vee \mathrm{iz}(2) \quad .$$

Let $w = (p_0, d_0)(p_1, d_1) \ldots (p_k, d_k)$ be a data word over $\mathbb{P}$. For all $0 \leq i \leq k$, if $p_i = (\cdot, \mathrm{inc}_j, \cdot)$ for some $j \in \{1, 2\}$, then we say that d_i *is the data value of some increment*; likewise for decrements and zero tests. Let

$$(p_{i_1}, d_{i_1})(p_{i_2}, d_{i_2}) \ldots (p_{i_\ell}, d_{i_\ell})$$

be the maximum subsequence of w such that $\mathrm{proj}(p_{i_j}) = \mathrm{inc}_m$ for some $m \in \{1, 2\}$, for every $j \in \{1, \ldots, \ell\}$. Then we say that $d_{i_1}, d_{i_2},, \ldots, d_{i_\ell}$ *is the data sequence of all increments.*

For defining φ, we start with one of our most important subformulas:

- The data sequence of all increments progresses by one:

$$\psi_1 := \mathrm{G}((\mathrm{inc} \wedge \mathrm{F}\,\mathrm{inc}) \rightarrow ((\neg \mathrm{F}_{\leq 0}\mathrm{inc}) \wedge (\mathrm{F}_{=1}\mathrm{inc}))).$$

Next, we explain the conditions that ensure a correct encoding of zero tests:

- For every increment on C_i for which there exists a future zero test on C_i, there exists some future matching decrement on C_i :

$$\psi_2 := \bigwedge_{i=1,2} \mathrm{G}((\mathrm{inc}(i) \wedge \mathrm{F}\,\mathrm{iz}(i)) \rightarrow \mathrm{F}_{=0}\mathrm{dec}(i)).$$

  Note that his alone does not express that the matching decrement takes place *before* the zero test. For this, we use the following trick:

- No data value of an increment is greater than the data value of some future zero test (if there is any):
$$\psi_3 := \mathrm{G}(\mathrm{inc} \rightarrow \neg\mathrm{F}_{<0}\mathrm{iz}).$$
- The data value of every zero test is smaller than the data value of every future increment:
$$\psi_4 := \mathrm{G}(\mathrm{iz} \rightarrow \neg\mathrm{F}_{\leq 0}\mathrm{inc}).$$
Recall that by (ii) there will always be a future increment in halting computations. Then ψ_1, ψ_3, and ψ_4 imply that the data value of every zero test is equal to the data value of the last preceding increment.
- The data value of every zero test on C_i is smaller than the data value of all future decrements on C_i:
$$\psi_5 := \bigwedge_{i=1,2} \mathrm{G}(\mathrm{iz}(i) \rightarrow \neg\mathrm{F}_{\leq 0}\mathrm{dec}(i)).$$
Recall that the data value of a zero test must be equal to the data value of the last preceding increment. Hence, every decrement after a zero test cannot be a match with an increment which happened before the zero test. Hence, the matching decrement that must exist by the preceding condition must occur before the zero test.

Next, we explain how to express that for every decrement there is a matching increment before. We cannot express this directly in MTL, because there are no past modalities that, from a current position k in w, allow us to look backwards to preceding positions $j < k$ in w. Instead, we define the following formulas:

- No data value is smaller than the data value of the first increment:
$$\psi_6 := \neg\mathrm{F}_{<0}\mathrm{true}.$$
Recall that by assumption, the first instruction of $\mathcal{M}$ is an increment; the first data value d_0 in w is thus assigned to an increment.
- The data value of every decrement is strictly smaller than the data value of every future increment:
$$\psi_7 := \mathrm{G}(\mathrm{dec} \rightarrow \neg\mathrm{F}_{\leq 0}\mathrm{inc}).$$
By (ii), the last instruction in $\mathcal{M}$ before the halting instruction is an increment. Hence, by ψ_1, the data value of every decrement must be smaller or equal to the data value of its last preceding increment. By ψ_6, the data value of every decrement can only be chosen from the data of one of the preceding increments. In other words, only increments can "produce" new data, and decrements can only choose a data value from the set of already produced data.
- The data value of each decrement is distinct from the data value of every other decrement:
$$\psi_8 := \mathrm{G}(\mathrm{dec} \rightarrow \neg\mathrm{F}_{=0}\mathrm{dec}).$$
In this way, we ensure that no two decrements are matched with the same increment.
- The data value of every increment on some counter is different from the data value of every future decrement on the other counter:
$$\psi_9 := \mathrm{G}[(\mathrm{inc}(1) \rightarrow \neg\mathrm{F}_{=0}\mathrm{dec}(2)) \wedge (\mathrm{inc}(2) \rightarrow \neg\mathrm{F}_{=0}\mathrm{dec}(1))].$$
In this way, we guarantee that matches between increments and decrements refer to the same counter.

We remark that due to the strict semantics for the *until* modality, a formula like $\mathsf{G}\phi$ enforces ϕ to hold everywhere along the data word, but only starting from the next position. For this reason, all conjuncts of the form $\mathsf{G}\phi$ should actually appear in the form $\phi \wedge \mathsf{G}\phi$.

Finally, φ is the conjunction of all of these formulas and the formula $\mathsf{F}\langle I_n, \text{halt}\rangle$. Note that indeed all formulas only contain unary modalities and no *next* modality.

In the following, we prove the correctness of the construction.

"Completeness". Let $\gamma = (J_0, c_0, c_0')(J_1, c_1, c_1') \dots (J_k, c_k, c_k')$ be a shortest halting computation of $\mathcal{M}$, where, by assumption (i), $J_0 = I_1$, $c_1 = 1$, and $J_k = I_n$. We show that there exists a finite computation ρ of $\mathscr{A}$ such that $\rho \models_{\text{MTL}} \varphi$. Let

$$u = \langle J_0, \text{op}_0, J_1\rangle \langle J_1, \text{op}_1, J_2\rangle \dots \langle J_{k-1}, \text{op}_{k-1}, J_k\rangle \langle J_k, \text{halt}\rangle$$

be the finite word over $\mathbb{P}\backslash\mathbb{P}_{\mathsf{dum}}$ such that for every $0 \leq i \leq k$,

- if J_i is an increment on C_j for some $j \in \{1, 2\}$, then $\text{op}_i = \text{inc}_j$;
- if J_i is a successful zero test on C_j for some $j \in \{1, 2\}$, then $\text{op}_i = \text{zero}_j$;
- if J_i is a successful decrement on C_j for some $j \in \{1, 2\}$, then $\text{op}_i = \text{dec}_j$.

Next, we assign to every letter $(J_i, \text{op}_i, J_{i+1})$ a data value d_i as follows:

(1) if $\text{op}_i = \text{inc}_j$ is the m-th increment on C_j in u, for some $j \in \{1, 2\}$, then set $d_i = m - 1$;
(2) for all $j \in \{1, 2\}$, if $\text{op}_i = \text{dec}_j$ is the m-th decrement on C_j in u, then set $d_i = d_\ell$, where $0 \leq \ell < k$ is such that op_ℓ is the m-th increment on C_j in u (note that by the fact that γ is a computation of $\mathcal{M}$, we have $\ell < i$);
(3) if $\text{op}_i = \text{zero}_j$ is a zero test on C_j in u, for some $j \in \{1, 2\}$, then set $d_i = d_\ell$, where $0 \leq \ell < i$ is the greatest index such that that $\text{op}_\ell = \text{inc}_{m'}$ is an increment on $C_{m'}$ in u, for some $m' \in \{1, 2\}$ (note that by assumption (i), such an index ℓ always exists); and
(4) add arbitrary data value to $\text{op}_i = \text{halt}$.

Note that in the resulting data word

$$w = (\langle J_0, \text{op}_0, J_1\rangle, d_0)\,(\langle J_1, \text{op}_1, J_2\rangle, d_1) \dots (\langle J_{k-1}, \text{op}_{k-1}, J_k\rangle, d_{k-1})\,(\langle J_k, \text{halt}\rangle, d_k)$$

over $\mathbb{P}$ the distance $d := d_i - d_{i+1}$ between two consecutive data values d_i and d_{i+1} may be non-zero. We "fill up" positive distances $d > 0$ by d dummy symbols dum^- and negative distances $d < 0$ by $|d|$ dummy symbols dum^+, w. More formally, let ρ be the data word obtained from w by applying the following two rules:

- If $d := d_i - d_{i+1} > 0$, then insert d dummy symbols dum^-_{ab} between positions i and $i + 1$; for every $1 \leq j \leq d$, the j-th inserted dummy symbol is assigned data value $d_i - j$.
- If $d := d_i - d_{i+1} < 0$, then insert $|d|$ dummy symbols dum^+_{ab} between positions i and $i + 1$; for every $1 \leq j \leq d$, the j-th inserted dummy symbol is assigned data value $d_i + j$.

Here, $a \in \{1, \dots, n\}$ is the index of the instruction $J_i = I_a$, and $b \in \{1, \dots, n\}$ is the index of the instruction $J_{i+1} = I_b$. Clearly, ρ is a finite computation of $\mathscr{A}$. It is easy to prove that $\rho \models_{\text{MTL}} \varphi$.

"Soundness". Let ρ be a finite computation of $\mathscr{A}$ such that $\rho \models_{\text{MTL}} \varphi$. We prove that there exists a halting computation of $\mathcal{M}$. By definition of $\mathscr{A}$, ρ induces a path that corresponds to a path in $\mathcal{M}$. We prove that this path is indeed a computation of $\mathcal{M}$ by proving that no cheatings on zero tests or decrements can happen.

No cheating on zero tests. We prove the following claim: for every $j \in \{1, 2\}$, for every position i in ρ, if $\text{op}_i = \text{zero}_j$, then the number B of positions $\ell \in \{0, \dots, i - 1\}$ with $\text{op}_\ell = \text{dec}_j$ is greater or equal to the number A of positions $m \in \{0, \dots, i - 1\}$ with $\text{op}_\ell = \text{inc}_j$. Towards contradiction, suppose that $A > B$. Let $i_1 < i_2 < \cdots < i_A$ be the sequence of indices in w such that $i_m < i$ and $\mathrm{op}_{i_m} = \mathrm{inc}_j$ for all $m \in \{1, \ldots, A\}$. Let $i_A \le i_{\mathrm{last}} < i$ be the last increment (not necessarily on C_j) before position i . Note that such an increment necessarily exists by assumption (i). Further, let $i_{\mathrm{next}} > i$ be the next increment (not necessarily on C_j) after position i. Note that such an increment exists by assumption (ii). By ψ_1, we have

$$d_{i_1} < d_{i_2} < \cdots < d_{i_A} \le d_{i_{\mathrm{last}}} < d_{i_{\mathrm{next}}} \quad \text{and} \quad d_{i_{\mathrm{next}}} = d_{i_{\mathrm{last}}} + 1.$$

By ψ_2, for each $m \in \{1, \ldots, A\}$, there exists some future position $j_m > i_m$ such that $\mathrm{op}_{j_m} = \mathrm{dec}_j$ and $d_{i_m} = d_{j_m}$. By assumption, one of these future decrements occurs only *after* position i, formally: there exists some position $m \in \{1, \ldots, A\}$ such that $i < j_m$, $\mathrm{op}_{j_m} = \mathrm{dec}_j$ and $d_{i_m} = d_{j_m}$. By ψ_3, $d_{i_{\mathrm{last}}} \le d_i$. By ψ_4, $d_i < d_{i_{\mathrm{next}}}$. Hence, $d_i = d_{i_{\mathrm{last}}}$. But by ψ_5, $d_i < d_{j_m}$, contradiction.

No cheating on zero tests. We prove that, for every position i in ρ and every $j \in \{1, 2\}$, the number A of all increments on counter C_j before position i is strictly greater than the number B of all decrements on counter C_j before position i. Towards contradiction, suppose that $A \le B$.

Let $i_1 < i_2 < \cdots < i_{A'}$ be the sequence of all positions $\ell < i$ such that $\mathrm{op}_{i_\ell} = \mathrm{inc}_m$ for some $m \in \{1, 2\}$. Note that by assumption (i), $i_1 = 0$. Let $i < i_{\mathrm{next}}$ be the first increment on some counter (not necessarily C_j) after position i. Note that by assumption (ii), such a position exists. By ψ_1, $d_{i_{\ell+1}} = d_{i_\ell} + 1$ for all $\ell \in \{1, \ldots, A'_1\}$, and $d_{i_{\mathrm{next}}} = d_{i_{A'}} + 1$.

Let $j_1 < j_2 < \cdots < j_B < j_{B+1} = i$ be the sequence of all positions until position i such that $\mathrm{op}_{j_\ell} = \mathrm{dec}_j$. By ψ_6, $d_{j_\ell} \ge d_{i_1}$ for all $\ell \in \{1, \ldots, B+1\}$. By ψ_7, $d_{j_\ell} < d_{i_{\mathrm{next}}}$ for all $\ell \in \{1, \ldots, B+1\}$. Hence, $d_{i_1} \le d_{j_\ell} < d_{i_{\mathrm{next}}}$ for all $\ell \in \{1, \ldots, B+1\}$. By ψ_8, $d_{j_\ell} \ne d_{j_{\ell'}}$ for all $\ell, \ell' \in \{1, \ldots, B+1\}$ with $\ell \ne \ell'$. Hence, $B+1$ out of the A' many data values in $\{d_{i_1}, \ldots, d_{i_{A'}}\}$ are used for d_{j_ℓ}. But by ψ_9, the value for d_{j_ℓ}, for every $\ell \in \{1, \ldots, B+1\}$ must come from one of the A many data values in $\{d_{i_1}, \ldots, d_{i_{A'}}\}$, a contradiction.

This finishes the proof for Σ^0_1-hardness of finitary model checking for MTL.

For obtaining Σ^1_1-hardness of infinitary model checking for MTL, we can easily adapt the proof by using a corresponding reduction from the Σ^1_1-complete recurrent state problem for Minsky machines using the formula $\mathrm{GF} \bigvee_{I_k \in S_j} \langle I_1, \mathrm{inc}_1, I_k \rangle$. □

We remark that the one-counter machine defined in the proof of Theorem 5.1 does not use zero tests on the counter—that is, the zero test capability of one-counter machines is not crucial to obtain undecidability. We can thus conclude that even for *one-dimensional vector addition systems with states*, the model-checking problem is undecidable. We further remark that the one-counter machine only uses constants $+1, -1$, and 0.

5.1 Purification

In the following, we show that the model-checking problem for one-counter machines and MTL is undecidable even if the MTL formula does not contain any propositional variables. For this, we prove the following purification lemma for MTL. We remark that the proof of this lemma is based on the ideas presented in the proof for a purification lemma for FLTL in Demri et al. [2010].

Lemma 5.2 (Purification for MTL). Given a one-counter machine $\mathcal{A}$ and an MTL formula φ, one can compute in space logarithmic in $|\mathcal{A}| + |\varphi|$ a one-counter machine $\mathcal{A}'$ and a pMTL formula φ' such that $\mathcal{A} \models^*_{\mathrm{MTL}} \varphi$ ($\mathcal{A} \models^\omega_{\mathrm{MTL}} \varphi$, respectively) if, and only if, $\mathcal{A}' \models^*_{\mathrm{MTL}} \varphi'$ ($\mathcal{A}' \models^\omega_{\mathrm{MTL}} \varphi'$, respectively). Moreover, φ only uses unary modalities if, and only if, φ' only uses unary modalities.

Proof. In the proof of Lemma 5 in Demri et al. [2010], it is explained how one can construct from a one-counter machine $\mathcal{A}$ with control states $Q = \{q_1, \ldots, q_n\}$ a one-counter machine $\mathcal{A}'$ such that every control state q_j can be identified in $\mathcal{A}'$ with a computation that follows a certain

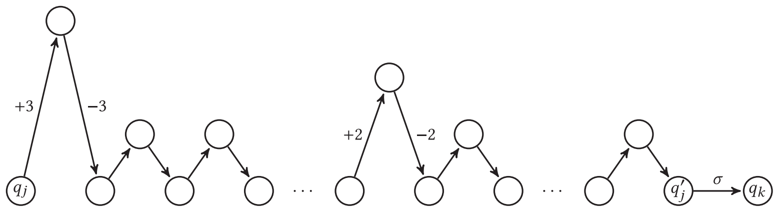

Fig. 2. Encoding an edge (q_j, q_k) with label σ by a pattern made of $n+1$ peaks and of length $2(n+1)+1$. We write $+c$ and $-c$, which, respectively, are short for add$(+c)$ and add$(-c)$. Up-going edges with no label represent edges with label add$(+1)$, and down-going edges with no label represent edges with label add(-1).

pattern about the changes of the counter value. This pattern follows the following rules (Figure 2). For every $j \in \{1, \ldots, n\}$, we define control states and edges between them such that for all $c \in \mathbb{N}$, the following conditions are satisfied:

- $(q_j, c) \to (p^j_1, c+3) \to (p^j_2, c)$,
- $(p^j_{2k}, c) \to (p^j_{2k+1}, c+1) \to (p^j_{2(k+1)}, c)$ for every $1 \le k < j$,
- $(p^j_{2j}, c) \to (p^j_{2j+1}, c+2) \to (p^j_{2(j+1)}, c)$, and
- $(p^j_{2j}, c) \to (p^j_{2k+1}, c+1) \to (p^j_{2(k+1)}, c)$ for every $j < k \le n$, where we set $q'_j = p^j_{2(n+1)}$.

Then, for every edge (q_j, q_k) with label σ, we define (q'_j, q_k) with label σ. This pattern is very useful for identifying control states in MTL without using propositional variables, as explained in the following:

- $\varphi_{\text{ctrlState}}$ holds in the current position of a data word if, and only if, the current position represents a control state of $\mathcal{A}'$ (i.e., at the beginning of a peak of height 3):
$$\varphi_{\text{ctrlState}} := \mathsf{X}_{=+3}\mathsf{X}_{=-3}\text{true}.$$
- For each $j \in \{1, \ldots, n\}$, we define a formula φ_j that holds in a position in a data word if, and only if, this position represents the control state q_j. For this, the current position should satisfy $\varphi_{\text{ctrlState}}$. The current position should also be followed by a peak of height 2 after $2j$ positions:
$$\varphi_{\text{ctrlState},j} := \varphi_{\text{ctrlState}} \wedge \mathsf{X}^{2i}(\mathsf{X}_{=+2}\mathsf{X}_{=-2}\text{true}).$$

We also have to translate the MTL formula φ to a pMTL formula φ'. This is done by a mapping $f : \text{MTL} \to \text{pMTL}$ defined by $f(q_j) = \varphi_{\text{ctrlState},j}$ for every $j \in \{1, \ldots, n\}$, $f(\neg\varphi) = \neg f(\varphi)$, $f(\varphi_1 \wedge \varphi_2) = f(\varphi_1) \wedge f(\varphi_2)$, and $f(\varphi_1 \mathsf{U}_I \varphi_2) = (\varphi_{\text{ctrlState}} \to f(\varphi_1))\mathsf{U}_I(\varphi_{\text{ctrlState}} \wedge f(\varphi_2))$. □

Note that in the proof of the purification lemma, we use the *next* modality. As a corollary from the purification lemma and Theorem 5.1 presented earlier, we yield the following.

Corollary 5.3. *Finitary model checking for* pMTL(F, X) *is* Σ^0_1*-hard, and infinitary model checking for* pMTL(F, X) *is* Σ^1_1*-hard.*

The exact decidability status for pMTL(F) is open.

6 THE SAT

In this section, we are interested in infinitary and finitary versions of the SAT for MTL and TPTL.

Alur and Henzinger [1994] proved more than 20 years ago that infinitary SAT for TPTL is undecidable and Σ^1_1-complete, even for the pure fragment of TPTL. The proof in the cited work is by a reduction of the recurrent state problem for Minsky machines. In the reduction, more than

Table 3. Complexity Results of Finitary and Infinitary SAT of TPTL and MTL

		U	F, X	F
Finitary	MTL	Σ^0_1-complete (Corollary 6.4)	Σ^0_1-complete (Corollary 6.4)	Open
	TPTL^1	Σ^0_1-complete (Theorem 6.5)	Σ^0_1-complete (Theorem 6.5)	Σ^0_1-complete (Theorem 6.5)
	FLTL^1	Ackermann-complete Demri and Lazic 2009]	Ackermann-complete [Demri and Lazic 2009]	Ackermann-complete Figueira and Legoufin 2009]
Infinitary	MTL	Σ^1_1-complete (Corollary 6.4)	Σ^1_1-complete (Corollary 6.4)	Open
	TPTL^1	Σ^1_1-complete [Alur and Henzinger 1994]	Σ^1_1-complete (Theorem 6.5)	Σ^1_1-complete (Theorem 6.5)
	FLTL^1	Π^0_1-complete [Demri and Lazic 2009]	Π^0_1-complete Demri and Lazic 2009]	Π^0_1-complete [Figueira and Segoufin 2009]

Note: The results in the first column concern the full logics, whereas the second and third columns contain the respective complexities for the unary fragment with and without the *next* modality. The lower bounds for the results shaded in grey are new and presented in this paper (the upper bounds follow from Theorem 6.1 respectively from Alur and Henzinger [1994]). The status of SAT for MTL(F) i.e., the unary fragment without the next modality is open.

one register variable is used. However, one can easily adapt the proof and strenghten the result to TPTL using only one register variable. We will later prove that finitary SAT for TPTL^1 is Σ^0_1-hard, even for some strict fragments of TPTL^1. For completeness, we prove in the following a Σ^0_1-upper bound for finitary SAT for TPTL^1.

THEOREM 6.1. *Finitary SAT for* TPTL^1 *is in* Σ^0_1.

PROOF. It is sufficient to show that the TPTL^1 formulas which are satisfied by a finite data word are recursively enumerable. The set $\text{TPTL}^1 \times (\mathbb{P} \times \mathbb{N})^+$ is recursively enumerable, as it is the Cartesian product of two recursively enumerable sets. We check the pairs (φ, w) in the enumeration one by one to see whether $w \models_{\text{TPTL}} \varphi$ or not. If $w \models_{\text{TPTL}} \varphi$, then we output φ; otherwise, we check the next pair. In this way, we can enumerate all finitely satisfiable formulas in TPTL^1. □

By Proposition 4.1, the upper bounds of SAT for TPTL^1 also apply to SAT for MTL. However, recall that TPTL^1 is strictly more expressive than MTL by Corollary 4.8. It is thus useful to consider the exact complexity of SAT for MTL in particular, as it is further known that finitary SAT for MTL over *timed words* is decidable, albeit with hyper-Ackermannian complexity [Ouaknine and Worrell 2007; Schmitz and Schnoebelen 2011], and SAT for MTL is EXPSPACE-complete for *monotonic* data words [Alur and Henzinger 1993].

However, we prove in the following that the model-checking problem for one-counter machines and MTL can be reduced to SAT for MTL. In the reduction, we use the *next* modality so that by the undecidability of the model-checking problem for MTL(F) (Theorem 5.1), we can infer the undecidability of SAT for MTL(F, X). This implies the undecidability of SAT for TPTL^1(F, X). Later, in Theorem 6.5, we go one step further and prove, by a novel reduction, that SAT for TPTL^1(F) is undecidable. We remark that this novel reduction does not work for MTL(F) (cf. Proposition 4.12). The results concerning the computational complexity of SAT for different fragments of TPTL and MTL are summarized in Table 3.

6.1 Lower Bounds for Unary MTL

We prove that the model-checking problem for one-counter machines and MTL can be reduced to SAT for MTL.

Proposition 6.2. *Given a one-counter machine $\mathcal{A}$ and an* MTL *formula φ, one can compute in space logarithmic in $|\mathcal{A}| + |\varphi|$ an* MTL *formula φ' such that $\mathcal{A} \models^*_{\mathrm{MTL}} \varphi$ ($\mathcal{A} \models^\omega_{\mathrm{MTL}} \varphi$, respectively) if, and only if, φ' is finitary (infinitary, respectively) satisfiable. Moreover, φ only uses unary modalities if, and only if, φ' only uses unary modalities.*

Proof. Let $\mathcal{A} = (Q, q_i, E, \lambda)$ be a one-counter machine, and let φ be an MTL formula over Q. First, we define an MTL formula $\varphi_{\mathcal{A}}$ over Q describing the computations of $\mathcal{A}$: $\varphi_{\mathcal{A}}$ is the conjunction of the following formulas:

- $\mathrm{G}_{<0}$false: all data values are at least as big as the initial value, which represents the initial data value of the counter (i.e., zero).
- For each $q \in Q$, we define
$$\mathrm{G}\left(\left(q \wedge \mathrm{X} \bigvee_{q' \in Q} q'\right) \rightarrow \bigvee_{\substack{q' \in Q, (q,q') \in E \\ \lambda(q,q') = \mathrm{zero}}} \mathrm{X}_{=0} q' \vee \bigvee_{\substack{q' \in Q, (q,q') \in E \\ \lambda(q,q') = \mathrm{add}(z)}} \mathrm{X}_{=z} q'\right).$$
This guarantees that the q' following q is indeed in relation with q as defined by the transition relation E.
- For each q with $(q, q') \in E$ and operation $\lambda(q, q') = \mathrm{zero}$, we define $\mathrm{G}_{>0}\neg(q \wedge \mathrm{X}_{=0} q')$. This excludes cheatings with zero test transitions.
- For each q with $(q, q') \in E$ and operation $\lambda(q, q') = \mathrm{add}(z)$ for some $z < 0$, we define $G_{<-z}\neg(q \wedge \mathrm{X}_{=-z} q')$. This excludes cheatings with decrements.

Clearly, $\mathcal{A} \models^*_{\mathrm{MTL}} \varphi$ if, and only if, $\varphi_{\mathcal{A}} \wedge \varphi$ is finitary satisfiable (and likewise for the infinitary case). □

Note that $\varphi_{\mathcal{A}}$ only uses unary modalities. The upper bound for the infinitary case in Alur and Henzinger [1994], Proposition 6.2, and Theorem 6.1 yield the upper bounds for the model-checking problem for TPTL^1.

Corollary 6.3. *Finitary model checking for* TPTL^1 *is in Σ^0_1; infinitary model checking for* TPTL^1 *is in Σ^1_1.*

Remark 6.1. We remark that one cannot use the reduction described previously to reduce the model-checking problem for one-counter machines and FLTL to SAT for FLTL: in FLTL, one cannot express the exact difference of the data values between two consecutive positions as simply as it can be done in MTL. Indeed, SAT for FLTL^1 is decidable [Demri and Lazic 2009], albeit with Ackermann complexity [Figueira et al. 2011], whereas the model-checking problem for FLTL^1 is undecidable [Demri et al. 2010].

As a corollary from Proposition 6.2 and Theorem 5.1, we obtain the following.

Corollary 6.4. *Finitary SAT for* MTL(F, X) *is Σ^0_1-hard, and infinitary SAT for* MTL(F, X) *is Σ^1_1-hard.*

Note that the undecidability result we give in Theorem 5.1 holds even for MTL(F) formulas that do not use the *next* modality. The use of X in the reduction from model checking to satisfiability in the proof of Proposition 6.2 seems to be unavoidable. In fact, the decidability status of MTL(F) without the *next* modality is open. The attentive reader may now observe that in the proof of Theorem 5.1, the syntactic conditions on data words encoding the computation of a Minsky machine (that are encoded by the one-counter machine) can easily be expressed by an MTL(F, X) formulas. However, it seems that also here the use of the *next* modality cannot be avoided.

6.2 Lower Bound for Unary TPTL^1 without the *Next* Modality

In the following, we will prove that SAT for TPTL^1(F) is undecidable.

THEOREM 6.5. *Finitary SAT for* TPTL^1(F) *is* Σ_1^0*-hard, and infinitary SAT for* TPTL^1(F) *is* Σ_1^1*-hard.*

PROOF. We present a novel reduction from the halting problem (the recurrent state problem, respectively) for Minsky machines. Let $\mathcal{M} = \{I_1, \ldots, I_n\}$ be a Minsky machine, where I_n is the single halting instruction. Define $\mathbb{P} = \{I_1, \ldots, I_n, C_1, C_2\}$. We define a TPTL$_1$(F) formula $\varphi_{\mathcal{M}}$ over $\mathbb{P}$ such that $\mathcal{M}$ has a halting computation, if, and only if, $\varphi_{\mathcal{M}}$ is satisfiable.

Let $\rho = (J_0, c_0, c_0')(J_1, c_1, c_1') \ldots (J_k, c_k, c_k')$ be a computation of $\mathcal{M}$. Let $d \in \mathbb{N}$ be an arbitrary data value. We encode ρ as a data word over $\mathbb{P}$ of the form

$$(J_0, d)(C_1, d{+}c_0)(C_2, d{+}c_0')(J_1, d{+}1)(C_1, d{+}1{+}c_1)$$
$$\times (C_2, d{+}1{+}c_1') \ldots (J_k, d{+}k)(C_1, d{+}k{+}c_k)(C_2, d{+}k{+}c_k').$$

For instance, for each $j \geq 0$, the j-th configuration (J_j, c_j, c_j') of ρ is encoded by the data word

$$(J_j, d{+}j)(C_1, d{+}j{+}c_j)(C_2, d{+}j{+}c_j').$$

The crucial point in this encoding is that the data sequence of all instruction symbols progresses by one. The data value d of J_0 serves as a reference value. In each of the TPTL^1(F) formulas defined in the following, we use this to determine when the encoding of a new configuration starts, even without using the *next* modality.

Next, we define TPTL^1(F) formulas describing the computations of $\mathcal{M}$. If possible, we give MTL(F) formulas, which, by Proposition 4.1, can be translated into equivalent TPTL^1(F) formulas. We explicitly give TPTL^1(F) formulas only if we are not able to find equivalent MTL(F) formulas. As an abbreviation, we will use

$$I := \bigvee_{1 \leq j \leq n} I_j.$$

(1) The data sequence of all instruction propositions progresses by one:

$$\mathrm{G}\left((I \wedge \mathrm{F}\, I) \rightarrow (\mathrm{F}_{=1} I \wedge \neg \mathrm{F}_{\leq 0} I)\right).$$

(2) For $i = 1, 2$, the data sequence of all C_i propositions is weakly monotonically increasing:

$$\mathrm{G}\left(C_i \rightarrow \neg \mathrm{F}_{<0} C_i\right).$$

Note that in our encoding, the difference between two consecutive occurrences of C_i is zero if, and only if, a decrementing instruction is encoded. The difference may, however, never be less than zero.

(3) For $i = 1, 2$, the data value of every C_i proposition is not smaller than the data value of any preceding instruction proposition:

$$\mathrm{G}\left(I \rightarrow \neg \mathrm{F}_{<0} C_i\right).$$

(4) Every instruction proposition should be immediately followed by a C_1 proposition, which itself should be immediately followed by a C_2 proposition. For avoiding the usage of the *next* modality, we take advantage of (1):

$$\mathrm{G}\left((I \wedge \mathrm{F} I) \rightarrow x.\, (\psi \wedge \eta)\right),$$

where

$$\psi := \mathrm{F}(C_1 \wedge \mathrm{F}(C_2 \wedge \mathrm{F}(I \wedge x = 1)))$$

expresses that for the instruction symbol with data value d', there is some future C_1, for which there is some future C_2, for which there is some future instruction symbol with data value $d' + 1$. By (1), we can infer that between two instruction symbols, there is thus indeed some C_1 followed by some C_2, and

$$\eta := \bigwedge_{i=1,2} \mathrm{G}\,(C_i \rightarrow \neg\mathrm{F}(C_i \wedge \mathrm{F}(I \wedge x = 1)))$$

expresses that between two instruction symbols, there cannot be more than one C_1 proposition and one C_2 proposition.

(5) The prefix of the data word should be of the form $(I_1, d)(C_1, d)(C_2, d)$:

$$I_1 \wedge \mathrm{F}_{=0}(C_1 \wedge \mathrm{F}_{=0}C_2).$$

By (4), we know that between I_1 with data value d and the next instruction symbol (let us say, J, with data value $d + 1$ by (1)), there is exactly one C_1 followed by exactly one C_2. By (3), the values of C_1 and C_2 must be at least d. Again by (3), the values of C_1 and C_2 after J must be at least $d + 1$. By (2), the data sequence of all C_i propositions must be monotonically increasing. Hence, the only chance to satisfy $\mathrm{F}_{=0}(C_1 \wedge \mathrm{F}_{=0}C_2)$ is to assign data value d to the C_1 and C_2 occurring between I_1 and J.

(6) The halting instruction I_n is followed by C_1 and C_2, in this order, and then the data word ends:

$$\mathrm{G}\left(I_n \rightarrow \left(\mathrm{F}(C_1 \wedge \mathrm{F}(C_2 \wedge \mathrm{G}(\neg\mathrm{true}))) \wedge \neg\mathrm{F}\,I \wedge \bigwedge_{i=1,2} \mathrm{G}(C_i \rightarrow \neg\mathrm{F}\,C_i)\right)\right)$$

(7) Assume that I_i is of the form

$$I_i : C_1 := C_1 + 1;\ \text{go to } S_j;$$

in addition, assume that the data word starting in the current position is of the form

$$(I_i, d')(C_1, e)(C_2, f)\ (J, d' + 1)(C_1, e')(C_2, f').$$

Following the idea for the encoding, the goal is to define a $\mathrm{TPTL}^1(\mathrm{F})$ formula that guarantees $J = I_j$, $e' = e + 2$, and $f' = f + 1$. We define

$$\mathrm{G}(I_i \rightarrow x.(\psi_j \wedge \eta_j)),$$

where

$$\psi_j := \mathrm{F}(C_1 \wedge \mathrm{F}(C_1 \wedge x = 2) \wedge \neg\mathrm{F}(C_1 \wedge x < 2) \wedge \mathrm{F}(I_j \wedge x = 1)),$$

and

$$\eta_j := \mathrm{F}(C_2 \wedge \mathrm{F}(C_2 \wedge x = 1) \wedge \neg\mathrm{F}(C_2 \wedge x < 1) \wedge \mathrm{F}(I_j \wedge x = 1)).$$

(8) Assume that I_i is of the form

$$I_i : \text{If } C_1 = 0 \text{ go to } S_j^0;\ \text{else } C_1 := C_1 - 1;\ \text{go to } S_k^1;$$

in addition, assume that the data word starting in the current position is of the form

$$(I_i, d')(C_1, e)(C_2, f)\ (J, d' + 1)(C_1, e')(C_2, f').$$

Following the encoding, the goal is to define a TPTL[1](F) formula that guarantees that (i) $e' = e + 1$, $f' = f + 1$, and $J = I_j$ if $e = d'$ (successful zero test), and (ii) $e' = e$, $f' = f + 1$, and $J = I_k$ if $e \neq d'$. We define

$$\mathrm{G}\,(I_i \rightarrow x.((\mathrm{F}(C_1 \wedge x = 0) \wedge \psi_{\text{zero}}) \vee (\neg \mathrm{F}(C_1 \wedge x = 0) \wedge \psi_{\text{dec}}))),$$

where ψ_{zero} and ψ_{dec}, respectively, express the valid encoding for the case that the zero test is successful and not successful. The formulas ψ_{zero} and ψ_{dec} can be defined similarly to ψ_j and η_j in (7).

(9) Instructions for the second counter can be encoded analogously.

(10) For the reduction from the halting problem (the recurrent state problem, respectively), the data word should contain the symbol for the halting instruction I_n (for instruction I_1 infinitely often, respectively):

$$\mathrm{F}\,I_n \qquad (\mathrm{GF}\,I_1, \text{respectively}).$$

We remark that, due to the strict semantics of the temporal modalities, some of the formulas using the modality globally (G) have to be additionally stated for the initial position of the data word but have been omitted here for brevity. We define $\varphi_{\mathcal{M}}$ to be the conjunction of the formulas defined earlier. Clearly, $w \models_{\text{TPTL}} \varphi_{\mathcal{M}}$ if, and only if, w encodes a halting (recurring, respectively) computation of $\mathcal{M}$, for every data word w over $\mathbb{P}$. □

6.3 Results for the Pure Fragments

Last but not least, we prove that SAT for MTL and TPTL is undecidable even if no propositional variables are used. The proof idea is very similar to that of the purification lemma in Section 5: we encode each configuration by a sequence of data values that form a certain pattern about the changes of the data values ("peaks").

Theorem 6.6. *Finitary SAT for* pMTL *is Σ_1^0-hard, and infinitary SAT for* pMTL *is Σ_1^1-hard.*

Proof. We reduce the halting problem, respectively the recurrent state problem, for Minsky machines to SAT for MTL. Let $\mathcal{M}$ be a Minsky machine with a set $\{I_1, \ldots, I_n\}$ of instructions. We define a pMTL formula $\varphi_{\mathcal{M}}$ that is satisfiable if, and only if, $\mathcal{M}$ has a halting computation, respectively, has a recurring computation.

Let $d \in \mathbb{N}$ be an arbitrary data value. Each configuration of the form (I_i, c_1, c_2) is encoded by a sequence of $2n + 6$ data values. The beginning of the encoding of every configuration is marked by the pair $(d, d + 3)$. The next $2n$ positions encode the index i of the current instruction I_i as follows: $(d, d + 1)^{i-1}(d, d + 2)(d, d + 1)^{n-i}$. Finally, the last four positions are used to encode the current values of the two counters: $(d, d + 4 + c_1)(d, d + 4 + c_2)$. By gluing together the encodings of configurations, we can encode a whole computation of $\mathcal{M}$. For an example, consider the prefix of a computation of the form

$$(I_1, 0, 0)(I_3, 1, 0) \tag{9}$$

for some Minsky machine with $n = 4$ instructions. Let $d = 0$. Then the following data word encodes (9):

$$0, 3, 0, 2, 0, 1, 0, 1, 0, 1, 0, 4, 0, 4, \quad 0, 3, 0, 1, 0, 1, 0, 2, 0, 1, 0, 5, 0, 4.$$

The pMTL formula $\varphi_{\mathcal{M}}$ is the conjunction of the formulas defined in the following. The pair $(d, d + 3)$ marks the beginning of a configuration; we thus define

$$\varphi_{\text{conf}} := \mathrm{X}_{=3}\text{true}.$$

To identify the index of the current instruction, we define for each $i \in \{1, \ldots, n\}$ a formula

$$\varphi_{I_i} := \mathrm{X}_{=3}\mathrm{X}_{=-3}\,(\mathrm{X}_{=1}\mathrm{X}_{=-1})^{i-1}\,\mathrm{X}_{=2}\mathrm{X}_{=-2}\,(\mathrm{X}_{=1}\mathrm{X}_{=-1})^{n-i}\,\mathrm{true}.$$

The next formula guarantees that the data word is of the form explained previously:

$$\mathrm{G}\left(\bigvee_{i\in\{1,\ldots,n\}} \varphi_{I_i} \wedge \bigwedge_{i\in\{1,\ldots,n-1\}}\left(\varphi_{I_i} \rightarrow \mathrm{X}^{2n+6} \bigvee_{i\in\{1,\ldots,n\}} \varphi_{I_i}\right)\right).$$

For encoding the initial configuration $(I_1, 0, 0)$, we define

$$\varphi_{I_1} \wedge \mathrm{X}^{2+2n}(\mathrm{X}_{=+4}\mathrm{X}_{=-4}\mathrm{X}_{=+4}\mathrm{X}_{=-4}\mathrm{true}).$$

Next, assume that I_j is an increment instruction of the form `C₁ := C₁+1; goto Sⱼ`. We encode the correct semantics of the application of I_j as follows:

$$\mathrm{G}\left(\varphi_{I_j} \rightarrow \left(\left(\mathrm{X}^{2n+6} \bigvee_{I_k\in S_j} \varphi_{I_k}\right) \wedge \varphi_{\mathrm{inc}} \wedge \varphi_{\mathrm{nochange}}\right)\right),$$

where

$$\varphi_{\mathrm{inc}} := \mathrm{X}^{2n+3}(\neg\mathrm{X}^3\varphi_{\mathrm{conf}}\mathrm{U}_{=1}\mathrm{X}^3\varphi_{\mathrm{conf}}) \quad \text{and} \quad \varphi_{\mathrm{nochange}} := \mathrm{X}^{2n+5}(\neg\mathrm{X}\varphi_{\mathrm{conf}}\mathrm{U}_{=0}\mathrm{X}\varphi_{\mathrm{conf}}).$$

A zero-test/decrement instruction can be encoded in a similar way. Testing whether the value of C_1 is zero in (e.g., I_j) can be done by $\varphi_{I_j} \wedge \mathrm{X}^{2+2n}\mathrm{X}_{=4}\mathrm{true}$. Reaching the halting instruction I_n is encoded by

$$\mathrm{F}\varphi_{I_n}.$$

The reduction to infinitary SAT for pMTL can be done similarly and is left to the reader. □

In the proof of the former theorem, the *until* modality is only used in the formulas that encode an instruction (e.g., see the subformula φ_{inc} of φ_j). Using TPTL^1, we can avoid the *until* modality and strengthen the result as follows.

THEOREM 6.7. *For* pTPTL^1(F, X)*, finitary SAT is Σ^0_1-hard, and infinitary SAT is Σ^1_1-hard.*

PROOF. We adapt the proof of Theorem 6.6 by replacing the formulas using the *until* modality by formulas in TPTL^1(F, X). For instance, the formula encoding the preceding increment instruction I_j can be replaced by

$$\mathrm{G}\left(\varphi_{I_j} \rightarrow \left(\left(x.\mathrm{X}^{2n+6}\left(\bigvee_{I_k\in S_j} \varphi_{I_k} \wedge x = 0\right)\right) \wedge \varphi'_{\mathrm{inc}} \wedge \varphi'_{\mathrm{nochange}}\right)\right),$$

where

$$\varphi'_{\mathrm{inc}} := \mathrm{X}^{2n+3}x.\left(\mathrm{X}^{2n+6} \wedge x = 1\right) \quad \text{and} \quad \varphi'_{\mathrm{nochange}} := \mathrm{X}^{2n+5}x.\left(\mathrm{X}^{2n+6} \wedge x = 0\right)$$ □

7 CONCLUSION AND OPEN PROBLEMS

The main open problem of this article is the decidability status of the SAT for the fragment MTL(F). Although the *next* modality can be avoided in reductions for showing undecidability for TPTL^1(F), it seems to be fundamental in all reductions we have looked at for showing undecidability of the unary fragment of MTL. At the same time, it seems surprising that the only absence of the *next* modality could be sufficient to change the decidability status.

REFERENCES

Luca Aceto and Anna Ingólfsdóttir (Eds.). 2006. *Proceedings of Foundations of Software Science and Computation Structures, 9th International Conference, FOSSACS 2006, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25-31, 2006.* Lecture Notes in Computer Science, Vol. 3921. Springer.

Rajeev Alur and David L. Dill. 1994. A theory of timed automata. *Theoretical Computer Science* 126, 2 (1994), 183–235.

Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. 1996. The benefits of relaxing punctuality. *Journal of the ACM* 43, 1 (1996), 116–146.

Rajeev Alur and Thomas A. Henzinger. 1993. Real-time logics: Complexity and expressiveness. *Information and Computation* 104, 1 (1993), 35–77.

Rajeev Alur and Thomas A. Henzinger. 1994. A really temporal logic. *Journal of the ACM* 41, 1 (1994), 181–204.

Mikołaj Bojańczyk, Anca Muscholl, Thomas Schwentick, Luc Segoufin, and Claire David. 2006. Two-variable logic on words with data. In *Proceedings of the 21st IEEE Symposium on Logic in Computer Science (LICS'06).* Los Alamitos, CA, 7–16. DOI: https://doi.org/10.1109/LICS.2006.51

Benedikt Bollig, Karin Quaas, and Arnaud Sangnier. 2017. The complexity of flat Freeze LTL. In *Proceedings of the 28th International Conference on Concurrency Theory (CONCUR'17).* Article 33, 16 pages. DOI: https://doi.org/10.4230/LIPIcs.CONCUR.2017.33

Ahmed Bouajjani, Javier Esparza, and Oded Maler. 1997. Reachability analysis of pushdown automata: Application to model-checking. In *Proceedings of the Concurrency Theory (CONCUR'97).* Lecture Notes in Computer Science, Vol. 1243. Springer, 135–150.

Ahmed Bouajjani, Peter Habermehl, Yan Jurski, and Mihaela Sighireanu. 2007. Rewriting systems with data. In *Proceedings of the Fundamentals of Computation Theory (FCT'07).* Lecture Notes in Computer Science, Vol. 4639. Springer, 1–22. DOI: https://doi.org/10.1007/978-3-540-74240-1_1

Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. 2010. On the expressiveness of TPTL and MTL. *Information and Computation* 208, 2 (2010), 97–116.

Patricia Bouyer, Kim Guldstrand Larsen, and Nicolas Markey. 2008. Model checking one-clock priced timed automata. *Logical Methods in Computer Science* 4, 2 (2008), 108–122.

Patricia Bouyer, Nicolas Markey, Joël Ouaknine, and James Worrell. 2007. The cost of punctuality. In *Proceedings of the 22nd IEEE Symposium on Logic in Computer Science (LICS'07).* Los Alamitos, CA, 109–120. DOI: https://doi.org/10.1109/LICS.2007.49

Patricia Bouyer, Antoine Petit, and Denis Thérien. 2003. An algebraic approach to data languages and timed languages. *Information and Computation* 182, 2 (2003), 137–162. DOI: https://doi.org/10.1016/S0890-5401(03)00038-5

Cristiana Chitic and Daniela Rosu. 2004. On validation of XML streams using finite state machines. In *Proceedings of the 7thInternational Workshop on the Web and Databases (WebDB'04), Colocated with ACM SIGMOD/PODS 2004.* 85–90. http://webdb2004.cs.columbia.edu/papers/6-2.pdf.

Stéphane Demri and Ranko Lazic. 2009. LTL with the freeze quantifier and register automata. *ACM Transactions on Computational Logic* 10, 3 (2009), Article 16.

Stéphane Demri, Ranko Lazic, and David Nowak. 2007. On the freeze quantifier in Constraint LTL: Decidability and complexity. *Information and Computation* 205, 1 (2007), 2–24.

Stéphane Demri, Ranko Lazic, and Arnaud Sangnier. 2008. Model checking Freeze LTL over one-counter automata. In *Proceedings of the Foundations on Software Science and Computational Structures (FoSSACS'08).* Lecture Notes in Computer Science, Vol. 4962. Springer, 490–504.

Stéphane Demri, Ranko Lazic, and Arnaud Sangnier. 2010. Model checking memoryful linear-time logics over one-counter automata. *Theoretical Computer Science* 411, 22–24 (2010), 2298–2316. DOI: https://doi.org/10.1016/j.tcs.2010.02.021

Stéphane Demri and Arnaud Sangnier. 2010. When model-checking Freeze LTL over counter machines becomes decidable. In *Proceedings of the Foundations of Software Science and Computational Structures (FoSSaCS'10).* Lecture Notes in Computer Science, Vol. 6014. Springer, 176–190.

Javier Esparza. 1996. Decidability and complexity of Petri net problems—An introduction. In *Proceedings of the Lectures on Petri Nets I: Basic Models (ACPN'96).* Lecture Notes in Computer Science, Vol. 1491. Springer, 374–428.

Kousha Etessami and Thomas Wilke. 2000. An until hierarchy and other applications of an Ehrenfeucht-Fraïssé game for temporal logic. *Information and Computation* 160, 1–2 (2000), 88–108. DOI: https://doi.org/10.1006/inco.1999.2846

Shiguang Feng, Markus Lohrey, and Karin Quaas. 2017. Path checking for MTL and TPTL over data words. *Logical Methods in Computer Science* 13, 3 (2017), 1–34. DOI: https://doi.org/10.23638/LMCS-13(3:19)2017

Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. 2011. Ackermannian and primitive-recursive bounds with Dickson's lemma. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science (LICS'11).* IEEE, Los Alamitos, CA, 269–278. DOI: https://doi.org/10.1109/LICS.2011.39

Diego Figueira and Luc Segoufin. 2009. Future-looking logics on data words and trees. In *Proceedings of the Mathematical Foundations of Computer Science (MFCS'09).* Lecture Notes in Computer Science, Vol. 5734. Springer, 331–343. DOI: https://doi.org/10.1007/978-3-642-03816-7_29

Stefan Göller, Christoph Haase, Joël Ouaknine, and James Worrell. 2010. Model checking succinct and parametric one-counter automata. In *Proceedings of the Automata, Languages and Programming (ICALP'10)*. Lectures Notes in Computer Science, Vol. 6199. Springer, 575–586.

Stefan Göller, Christoph Haase, Joël Ouaknine, and James Worrell. 2012. Branching-time model checking of parametric one-counter automata. In *Proceedings of the Foundations of Software Science and Computational Structures (FoSSaCS'12)*. Lecture Notes in Computer Science, Vol. 7213. Springer, 406–420. DOI: https://doi.org/10.1007/978-3-642-28729-9_27

Stefan Göller and Markus Lohrey. 2013. Branching-time model checking of one-counter processes and timed automata. *SIAM Journal on Computing* 42, 3 (2013), 884–923. DOI: https://doi.org/10.1137/120876435

Christoph Haase, Joël Ouaknine, and James Worrell. 2016. Relating reachability problems in timed and counter automata. *Fundamenta Informaticae* 143, 3–4 (2016), 317–338. DOI: https://doi.org/10.3233/FI-2016-1316

Piotr Hofman, Slawomir Lasota, Richard Mayr, and Patrick Totzke. 2016. Simulation problems over one-counter nets. *Logical Methods in Computer Science* 12, 1 (2016), 1–46. DOI: https://doi.org/10.2168/LMCS-12(1:6)2016

Michael Kaminski and Nissim Francez. 1994. Finite-memory automata. *Theoretical Computer Science* 134, 2 (1994), 329–363. DOI: https://doi.org/10.1016/0304-3975(94)90242-9

Ron Koymans. 1990. Specifying real-time properties with metric temporal logic. *Real-Time Systems* 2, 4 (1990), 255–299.

Pascal Lafourcade, Denis Lugiez, and Ralf Treinen. 2005. Intruder deduction for *AC*-like equational theories with homomorphisms. In *Proceedings of the Term Rewriting Techniques and Applications (RTA'05)*. Lecture Notes in Computer Science, Vol. 3467. Springer, 308–322. DOI: https://doi.org/10.1007/978-3-540-32033-3_23

François Laroussinie, Nicolas Markey, and Ph. Schnoebelen. 2002. On model checking durational Kripke structures. In *Proceedings of the Foundations of Software Science and Computation Structures (FoSSaCS'02)*. Lecture Notes in Computer Science, Vol. 2303. Springer, 264–279.

Ranko Lazic, Joël Ouaknine, and James Worrell. 2016. Zeno, Hercules, and the Hydra: Safety metric temporal logic is Ackermann-complete. *ACM Transactions on Computational Logic* 17, 3 (2016), Article 16, 27 pages. DOI: https://doi.org/10.1145/2874774

Antonia Lechner, Richard Mayr, Joël Ouaknine, Amaury Pouly, and James Worrell. 2016. Model checking flat Freeze LTL on one-counter automata. In *Proceedings of the 27th International Conference on Concurrency Theory (CONCUR'16)*. Article 29, 14 pages. DOI: https://doi.org/10.4230/LIPIcs.CONCUR.2016.29

Alexei Lisitsa and Igor Potapov. 2005. Temporal logic with predicate lambda-abstraction. In *Proceedings of the 12th International Symposium on Temporal Representation and Reasoning (TIME'05)*. IEEE, Los Alamitos, CA, 147–155. DOI: https://doi.org/10.1109/TIME.2005.34

Marvin L. Minsky. 1961. Recursive unsolvability of Post's problem of "Tag" and other topics in theory of Turing machines. *Annals of Mathematics* 74, 3 (Nov. 1961), 437–455.

Frank Neven, Thomas Schwentick, and Victor Vianu. 2004. Finite state machines for strings over infinite alphabets. *ACM Transactions on Computational Logic* 5, 3 (2004), 403–435. DOI: https://doi.org/10.1145/1013560.1013562

Joël Ouaknine and James Worrell. 2006a. On metric temporal logic and faulty Turing machines. In *Proceedings of the 9th European Joint Conference on Foundations of Software Science and Computation Structures (FOSSACS'06)*. 217–230.

Joël Ouaknine and James Worrell. 2006b. Safety metric temporal logic is fully decidable. In *Proceedings of the Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*. Lecture Notes in Computer Science, Vol. 3920. Springer, 411–425. DOI: https://doi.org/10.1007/11691372_27

Joël Ouaknine and James Worrell. 2007. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Computer Science* 3, 1 (2007), 1–27.

Paritosh K. Pandya and Simoni S. Shah. 2011. On expressive powers of timed logics: Comparing boundedness, non-punctuality, and deterministic freezing. In *Proceedings of the Concurrency Theory (CONCUR'11)*. Lecture Notes in Computer Science, Vol. 6901. Springer, 60–75. DOI: https://doi.org/10.1007/978-3-642-23217-6_5

Hiroshi Sakamoto and Daisuke Ikeda. 2000. Intractability of decision problems for finite-memory automata. *Theoretical Computer Science* 231, 2 (2000), 297–308. DOI: https://doi.org/10.1016/S0304-3975(99)00105-X

Sylvain Schmitz and Philippe Schnoebelen. 2011. Multiply-recursive upper bounds with Higman's lemma. In *Proceedings of the Automata, Languages and Programming (ICALP'11)*. Lecture Notes in Computer Science, Vol. 6756. Springer, 441–452. DOI: https://doi.org/10.1007/978-3-642-22012-8_35

Olivier Serre. 2006. Parity games played on transition graphs of one-counter processes. In *Proceedings of the Foundations of Software Science and Computation (FoSSaCS'06)*. Lecture Notes in Computer Science, Vol. 3921. Springer, 337–351. DOI: https://doi.org/10.1007/11690634_23

Received July 2016; revised March 2018; accepted September 2019