Satisfiability for MTL and TPTL over Non-monotonic Data Words

Claudia Carapelle*, Shiguang Feng*, Oliver Fernández Gil*, and Karin Quaas**

Institut für Informatik, Universität Leipzig,
04109 Leipzig, Germany

Abstract. In the context of real-time systems, Metric Temporal Logic (MTL) and Timed Propositional Temporal Logic (TPTL) are prominent and widely used extensions of Linear Temporal Logic. In this paper, we examine the possibility of using MTL and TPTL to specify properties about classes of non-monotonic data languages over the natural numbers. Words in this class may model the behaviour of, *e.g.*, one-counter machines. We proved, however, that the satisfiability problem for many reasonably expressive fragments of MTL and TPTL is undecidable, and thus the use of these logics is rather limited. On the positive side we prove that satisfiability for the existential fragment of TPTL is NP-complete.

1 Introduction

Recently, verification and analysis of sets of *data words* have gained a lot of interest [6, 18, 11, 10, 3–5]. A data word is a sequence over $\Sigma \times D$, where Σ is a finite set of labels, and D is a (potentially infinite) set of *data values*. In this paper, we consider data words as behavioural models of one-counter machines. In this regard, we let the data domain be the set of natural numbers. Note that the sequence of data values of a word may be non-monotonic.

For reasoning about data words, one may use extensions of linear temporal logic (LTL). One of these is FreezeLTL, which extends LTL with a *freeze quantifier* that stores the current data value in a register variable. The registers can be used to test for equality of data values at different positions of a data word. In spite of this limited access to data values, the satisfiability problem for FreezeLTL is undecidable [10]. However, over *finite* data words, and if the logic is restricted to a single register, then the satisfiability problem is decidable, albeit not primitive recursive [10]. This lower bound has been confirmed for satisfiability for the fragment of FreezeLTL where the only temporal modality is the *finally* modality [12].

Originally, the freeze quantifier was introduced in *Timed Propositional Temporal Logic* (TPTL, for short) [2]. With TPTL, in addition to FreezeLTL, one can compare data values of a data word using linear inequations of the form, *e.g.*, $x - y \leq c$. Another widely used logic in the context of real-time systems is *Metric*

* The author is supported by *Deutsche Forschungsgemeinschaft (DFG).*, GRK 1763
** The author is supported by *DFG*, project QU 316/1-1.

Temporal Logic (MTL, for short). MTL extends LTL by constraining the temporal operators with intervals of the non-negative reals. Both logics, however, have not gained much attention in the specification of *non-monotonic* data words, albeit they can express many interesting properties.

FreezeLTL is a fragment of TPTL, and thus it is clear that one cannot find better decidability results for TPTL than for FreezeLTL. In fact it is well known that the satisfiability problem for TPTL over non-monotonic data words is undecidable [2]. In the context of monotonic data words over the natural numbers, MTL and TPTL are equally expressive, and the satisfiability problem for both logics is EXPSPACE-complete [1, 2]. However, over timed words, TPTL has been proved strictly more expressive than MTL [7], and while satisfiability for both logics is undecidable over infinite timed words [1, 15], there is a difference in the finite words case: TPTL has an undecidable satisfiability problem [1], while satisfiability for MTL is decidable (but not primitive recursive) [16]. We recently proved in [8] that also for *non-monotonic* data words over the natural numbers TPTL is strictly more expressive than MTL, and indeed, there are properties which can be expressed in FreezeLTL, but cannot be expressed in MTL. Hence there was the possibility that MTL would have a better complexity for the satisfiability problem.

However, as a main result we prove that the satisfiability problem for MTL over non-monotonic finite data words is undecidable. This is even the case if we do not allow for propositional variables.

We then investigate the *unary fragments* of MTL and TPTL where the only allowed temporal modalities are unary. We show that the satisfiability problem over finite data words is undecidable for both logics, and for TPTL it is undecidable even if we restrict the formulae to contain at most one register variable and no next modality. This is opposed to the decidability result for FreezeLTL with one register variable evaluated over finite data words [10].

After that we consider another syntactic restriction of the logics, namely we restrict the negation operator to propositional variables, which results in what we call the *positive* fragments of our logics. This excludes the globally modality, which is used in most of the undecidability proofs. However, we prove that this restriction does not lead to any changes in the results for the satisfiability problem compared to the full logics.

Last but not least, we prove that for the *unary positive* fragment (called *existential* fragment in [7]), the satisfiability problem for TPTL is NP-complete.

The main insight of this paper is that both MTL and TPTL have a very limited use in specifying properties over non-monotonic data languages. This adds an important piece to complete the picture about decidability of satisfiability problems for data-relevant extensions of temporal logics.

2 Preliminaries

We use $\mathbb{Z}$ and $\mathbb{N}$ to denote the set of integers and the non-negative integers, respectively. We let P be a finite set of propositional variables.

A *data word* over P is a finite or infinite sequence $(P_0, d_0)(P_1, d_1)\dots$ of pairs in $2^{\mathsf{P}} \times \mathbb{N}$. We use $(2^{\mathsf{P}} \times \mathbb{N})^*$ and $(2^{\mathsf{P}} \times \mathbb{N})^\omega$, respectively, to denote the set of finite and infinite, respectively, data words over P. The *length* of a data word w is denoted by $|w|$, where we set $|w| = \infty$ if w is an infinite data word.

A *two-counter machine* $\mathcal{M}$ is a finite sequence $(\mathcal{I}_j)_{j=1}^n$ of instructions operating on two counters denoted by C_1 and C_2, where $\mathcal{I}_j$ is one of the following instructions (with $i \in \{1, 2\}$ and $j, k, m \in \{1, ..., n\}$):

increment	$\mathcal{I}_j : C_i := C_i + 1;$ `go to` $\mathcal{I}_k$
zero test/decrement	$\mathcal{I}_j :$ `if` $C_i = 0$ `then go to` $\mathcal{I}_k$ `else` $C_i := C_i - 1;$ `go to` $\mathcal{I}_m$
halt	$\mathcal{I}_j :$ `halt`

A *configuration* of a two-counter machine $\mathcal{M}$ is a triple $\gamma = (J, c, d) \in \{\mathcal{I}_1, ..., \mathcal{I}_n\} \times \mathbb{N} \times \mathbb{N}$, where J indicates the current instruction, and c and d are the current values of the counters C_1 and C_2, respectively. A *computation* of $\mathcal{M}$ is a finite or infinite sequence $(\gamma_i)_{i \geq 0}$ of configurations, such that $\gamma_0 = (\mathcal{I}_1, 0, 0)$ and γ_{i+1} is the result of executing the instruction $\mathcal{I}_i$ on γ_i for each $i \geq 0$. Without loss of generality, we assume that $\mathcal{I}_n$ is the only instruction of the form `halt`. The *halting problem* for two-counter machines asks, given a two-counter machine $\mathcal{M}$, whether the (unique) computation of $\mathcal{M}$ reaches a configuration with instruction $\mathcal{I}_n$, *i.e.*, the halting instruction. This problem is Σ_1^0-complete [14]. The *recurrent state problem* for two-counter machines asks, given a two-counter machine $\mathcal{M}$, whether the (unique) computation of $\mathcal{M}$ visits instruction $\mathcal{I}_1$ infinitely often. This problem is Σ_1^1-complete [2]. We will use reductions of these problems to show lower bounds of the satisfiability problem for some fragments of MTL and TPTL .

3 Extensions of Linear Temporal Logic

3.1 Metric Temporal Logic

The set of formulae of MTL is built up from P by boolean connectives and a constraining version of the *until* modality:

$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathsf{U}_I \varphi_2$$

where $p \in \mathsf{P}$ and $I \subseteq \mathbb{Z}$ is an interval with endpoints in $\mathbb{Z} \cup \{-\infty, +\infty\}$. We use pseudo-arithmetic expressions to denote intervals, like, *e.g.*, ≥ 1 to denote $[1, \infty)$. If $I = \mathbb{Z}$, then we may omit the annotation I on U_I.

Formulae in MTL are interpreted over data words. Let $w = (P_0, d_0)(P_1, d_1)\dots$ be a data word, and let $i \leq |w|$. We define the *satisfaction relation for* MTL, denoted by $\models_{\mathsf{MTL}}$, inductively as follows:

$$(w, i) \models_{\mathsf{MTL}} p \Leftrightarrow p \in P_i, \quad (w, i) \models_{\mathsf{MTL}} \neg\varphi \Leftrightarrow (w, i) \not\models_{\mathsf{MTL}} \varphi,$$
$$(w, i) \models_{\mathsf{MTL}} \varphi_1 \wedge \varphi_2 \Leftrightarrow (w, i) \models_{\mathsf{MTL}} \varphi_1 \text{ and } (w, i) \models_{\mathsf{MTL}} \varphi_2,$$
$$(w, i) \models_{\mathsf{MTL}} \varphi_1 \mathsf{U}_I \varphi_2 \Leftrightarrow \exists j. i < j \leq |w| : (w, j) \models_{\mathsf{MTL}} \varphi_2 \text{ and } d_j - d_i \in I, \text{ and }$$
$$\forall k. i < k < j : (w, k) \models_{\mathsf{MTL}} \varphi_1.$$

We say that a data word *satisfies* an MTL formula φ, written $w \models_{\mathsf{MTL}} \varphi$, if $(w, 0) \models_{\mathsf{MTL}} \varphi$. We use the following syntactical abbreviations: $\varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\varphi_1 \rightarrow \varphi_2 := \neg\varphi_1 \vee \varphi_2$, $\mathtt{true} := p \vee \neg p$, $\mathtt{false} := \neg\mathtt{true}$, $\mathsf{X}_I\varphi := \mathtt{false}\mathsf{U}_I\varphi$, $\mathsf{F}_I\varphi := \mathtt{true}\mathsf{U}_I\varphi$, $\mathsf{G}_I\varphi := \neg\mathsf{F}_I\neg\varphi$. Note that the use of the *strict* semantics for the until modality is essential to derive the next modality.

We define the *length* of a formula ψ, denoted by $|\psi|$, as the number of symbols occurring in ψ where all integer constants in ψ are given in a binary encoding.

In the following, we define some fragments of MTL. A $\mathsf{unaryMTL}$ formula is built from propositional variables, using the boolean connectives, and the unary temporal modalities X and F. We use $\mathsf{positiveMTL}$ to denote the subset of MTL where negation is restricted to propositional variables. A $\mathsf{posUnaryMTL}$ formula is a $\mathsf{positiveMTL}$ formula where the only allowed modalities are the F and X modalities.

3.2 Timed Propositional Temporal Logic

Next we define formulae of TPTL. For this, let X be a countable set of *register variables.* Formulae in TPTL are defined by the following grammar:

$$\varphi ::= p \mid x \sim c \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1\mathsf{U}\varphi_2 \mid x.\varphi$$

where $p \in \mathsf{P}$, $x \in X$, $c \in \mathbb{Z}$, and $\sim \in \{<, \leq, =, \geq, >\}$.

Formulae in TPTL are interpreted over data words. A *register valuation* ν is a function from X to $\mathbb{N}$. Let $w = (P_0, d_0)(P_1, d_1)\ldots$ be a data word, let ν be a register valuation, and let $i \in \mathbb{N}$. The satisfaction relation for TPTL, denoted by $\models_{\mathsf{TPTL}}$, is inductively defined in a similar way as for MTL; we only give the definitions for the new formulae:

$$\begin{aligned}(w,i,\nu) \models_{\mathsf{TPTL}} \varphi_1\mathsf{U}\varphi_2 &\Leftrightarrow \exists i.i < j \leq |w|.(w,j,\nu) \models_{\mathsf{TPTL}} \varphi_2,\\ &\qquad \text{and } \forall k.i<k<j.(w,k,\nu) \models_{\mathsf{TPTL}} \varphi_1,\\ (w,i,\nu) \models_{\mathsf{TPTL}} x \sim c &\Leftrightarrow d_i - \nu(x) \sim c,\\ (w,i,\nu) \models_{\mathsf{TPTL}} x.\varphi &\Leftrightarrow (w, \nu[x \mapsto d_i], i) \models_{\mathsf{TPTL}} \varphi.\end{aligned}$$

Here, $\nu[x \mapsto d_i]$ is the valuation that agrees with ν on all $y \in X\backslash\{x\}$, and maps x to d_i. We say that a data word w satisfies a TPTL formula φ, written $w \models_{\mathsf{TPTL}} \varphi$, if $(w, 0, \bar{0}) \models_{\mathsf{TPTL}} \varphi$, where $\bar{0}$ denotes the valuation that maps all variables to the initial data value of the word, *i.e.* to d_0.

We use the same syntactical abbreviations as for MTL. The *length* of TPTL formulae is also defined as for MTL formulae. We define the fragments $\mathsf{unaryTPTL}$, $\mathsf{positiveTPTL}$, and $\mathsf{posUnaryTPTL}$ like the corresponding fragments in MTL. Additionally, we define $\mathsf{FreezeLTL}$ to be the subset of TPTL formulae φ where $\sim c$ is of the form $= 0$ whenever φ contains the subformula $x \sim c$. Given $n \geq 0$ and a TPTL fragment $\mathcal{L}$, we use $\mathcal{L}^n$ to denote the subset of $\mathcal{L}$ that corresponds to the set of $\mathcal{L}$ formulae that use at most n different register variables.

4 The Satisfiability Problem

In this paper, we are interested in infinitary and finitary versions of the *satisfiability problem* (SAT, for short): given a formula φ in a logic $\mathcal{L}$, is there some infinite (finite, respectively) data word w with $w \models_{\mathcal{L}} \varphi$?

In the table below, we summarize the complexity results for the satisfiability problem for different fragments of TPTL and MTL. The results shaded in grey are new and presented in this paper.

		full	unary	unary without X	positive	posUnary
Finitary	MTL	Σ^0_1-cpl.	Σ^0_1-cpl.	?	Σ^0_1-cpl.	NP-cpl.
	TPTL^1	Σ^0_1-cpl.	Σ^0_1-cpl.	Σ^0_1-cpl.	Σ^0_1-cpl.	NP-cpl.
	FreezeLTL^1	not pr. rec. [10]	not pr. rec. [10]	not pr. rec. [12]	not pr. rec.	NP-cpl.
Infinitary	MTL	Σ^1_1-cpl.	Σ^1_1-cpl.	?	Σ^0_1-cpl.	NP-cpl.
	TPTL^1	Σ^1_1-cpl. [2]	Σ^1_1-cpl.	Σ^1_1-cpl.	Σ^0_1-cpl.	NP-cpl.
	FreezeLTL^1	Π^0_1-cpl. [10]	Π^0_1-cpl. [10]	Π^0_1-cpl. [12]	not pr. rec.	NP-cpl.

5 Results for Full and Unary Fragments

Alur and Henzinger proved already 20 years ago that infinitary SAT for TPTL is undecidable, even if one does not allow for propositional variables [2]. The proof in the cited paper is by reduction of the recurrent state problem for two-counter machines. In the reduction more than one register variable is used, however, one can easily adapt the proof and strenghten the result.

Theorem 1. *For* TPTL^1*, finitary SAT is Σ^0_1-complete, even for the fragment that does not allow propositional variables.*

Proof. Σ^0_1-hardness of finitary SAT can be proved in a similar way to Σ^1_1-hardness of infinitary SAT using a reduction of the halting problem for two-counter machines. It remains to show that finitary SAT is in Σ^0_1. For this, we note that given a TPTL^1 formula ψ and a finite data word w with $w \models_{\mathsf{TPTL}} \psi$, there exists a finite data word w' such that: $|w'| = |w| = n$, $w' \models_{\mathsf{TPTL}} \psi$ and the data values occurring in w' are bounded by $n * \max(|\ min_c\ |, |\ max_c\ |)$, where min_c and max_c are the smallest and greatest constants occurring in ψ. Based on this, the satisfiability of ψ can be characterized by a Σ^0_1 sentence $\exists n \phi(n)$ where $\phi(n)$ has only bounded quantifiers. Any data word can be encoded by using a unary predicate for each propositional variable p in ψ and a binary predicate to encode pairs of position numbers and associated data values. The variable n represents the length of a finite data word and $\phi(n)$ expresses whether ψ is satisfied by a data word of length n. □

It is well known that every formula in MTL can effectively be translated into a TPTL^1 formula defining the same language of data words. Hence the upper bounds of SAT for TPTL^1 also apply to SAT for MTL. However, we have recently proved that TPTL^1 over non-monotonic data words is *strictly more expressive* than MTL [8]. It is thus natural to consider the exact complexity of SAT for MTL, in particular, as it is further known that finitary SAT for MTL is decidable (albeit with non-primitive recursive complexity) for *timed words* [16], and $\mathsf{EXPSPACE}$-complete for *monotonic* data words [1]. However, we prove that the undecidability of SAT for TPTL^1 also applies to MTL, even if we do not allow for propositional variables.

Theorem 2. *For* MTL*, finitary SAT is* Σ_1^0*-complete, and infinitary SAT is* Σ_1^1*-complete, even for the fragment that does not allow propositional variables.*

Proof. (Sketch) The upper bounds follow from Th. 1 and the result by Alur and Henzinger [2]. For the lower bounds of finitary, respectively, infinitary SAT, we reduce the halting problem, respectively the recurrent state problem for two-counter machines to SAT for MTL: given a two-counter machine $\mathcal{M}$, we define an MTL formula $\varphi_{\mathcal{M}}$ that is satisfiable if, and only if, $\mathcal{M}$ reaches the halting instruction, respectively visits the first instruction infinitely often. In order to avoid the usage of propositional variables, we use an idea similar to the one in the proof of Lemma 2 in [11]. Each instruction $\mathcal{I}_i$ is encoded by a sequence of data values, starting with value 3, which is followed by n positions with data values in $\{1, 2\}$. The value 2 occurs at the i^{th} position after the value 3. After n positions, there is a position for encoding the value of C_1 plus 4, and after that there is a further position for encoding the value of C_2 plus 4. The first position with data value 3 can be used to identify an instruction of $\mathcal{M}$. For example, the configuration $(\mathcal{I}_3, 1, 2)$ of a two-counter machine with 4 instructions is encoded by $s\ s{+}3\ s\ s{+}1\ s\ s{+}1\ s\ s{+}2\ s\ s{+}1\ s\ s{+}4\ s\ s{+}5\ s$. Here, $s \in \mathbb{N}$ is some arbitrary initial data value. □

In [10], non-primitive recursive complexity for finitary SAT for $\mathsf{unaryFreezeLTL}^1$ is proved. This result was strengthened to SAT for $\mathsf{unaryFreezeLTL}^1$ without the X modality [12]. Unfortunately, if we extend $\mathsf{unaryFreezeLTL}^1$ to $\mathsf{unaryTPTL}^1$, we already obtain undecidability of SAT. We also prove undecidability of SAT for $\mathsf{unaryMTL}$, however, it is an open problem whether undecidability also holds for the $\mathsf{unaryMTL}$ fragment in which the X modality is not allowed.

Theorem 3. *For* $\mathsf{unaryMTL}$*, finitary SAT is* Σ_1^0*-complete, and infinitary SAT is* Σ_1^1*-complete. For* $\mathsf{unaryTPTL}^1$*, this is the case even if we do not allow for the* X *modality.*

Proof. The upper bounds follow from Theorems 1 and 2. For the lower bounds, we reduce the halting problem for two-counter machines to SAT for $\mathsf{unaryMTL}$ and to SAT for $\mathsf{unaryTPTL}^1$ without the X modality, respectively. Let $\mathcal{M}$ be a two-counter machine with n instructions. Recall that $\mathcal{I}_n$ is the only halting instruction. Define $\mathsf{P} = \{\#, \mathcal{I}_1, \ldots, \mathcal{I}_n, r_1, r_2\}$. We define a formula $\varphi_{\mathcal{M}}$ over P that is satisfiable if, and only if, $\mathcal{M}$ has a halting computation.

Let $\gamma = (J_0, c_0, d_0)(J_1, c_1, d_1)\dots$ be a computation of $\mathcal{M}$, where $(J_0, c_0, d_0) = (\mathcal{I}_1, 0, 0)$. . We encode γ as a data word over 2^{P} as follows:

$$(\{\#\}, s)(\{J_0, r_1\}, s + c_0)(\{J_0, r_2\}, s + d_0)(\{\#\}, s + 1)(\{J_1, r_1\}, s + 1 + c_1)\dots$$

i.e., for each $i \geq 0$, the i^{th} configuration of π is encoded by the data word

$$(\{\#\}, s + i)(\{J_i, r_1\}, s + i + c_i)(\{J_i, r_2\}, s + i + d_i).$$

Here $s \in \mathbb{N}$ is again an arbitrary initial data value.

The crucial point in this encoding is that the sequence of data values at positions where # holds is strictly monotonically increasing by exactly 1. In all of the $\mathsf{unaryTPTL}^1$ formulae (and some of the $\mathsf{unaryMTL}$ formulae, respectively) defined below, we can exploit this fact and determine when the encoding of a new configuration starts without using the X modality.

Next we define some $\mathsf{unaryMTL}$ formulae. The conjunction of all these formulae is satisfied by a data word w, if and only if, w encodes a *halting* computation of $\mathcal{M}$. Recall that every $\mathsf{unaryMTL}$ formula without the X modality can effectively be translated into a $\mathsf{unaryTPTL}^1$ formula without the X modality. For the reduction to SAT for $\mathsf{unaryTPTL}^1$ without the X modality, we give extra formulae only in the case that the $\mathsf{unaryMTL}$ formula uses the X modality. We further remark that, due to the strict semantics of our logics, some of the formulae using the G modality have to be additionally stated for the initial position of the data word, but have been omitted here due to lack of space. We start by defining the auxiliary formula $\varphi_{\mathsf{idz}} = \bigvee_{i\in\{1,\dots,n-1\}} \mathcal{I}_i$ (*i.e.*, the disjunction of all instructions without the halting instruction $\mathcal{I}_n$).

(1) At each position in the data word, exactly one of the following subsets of P must occur: $\{\#\}$, $\{\mathcal{I}_i, r_1\}$ and $\{\mathcal{I}_i, r_2\}$, for some $i \in \{1, \dots, n\}$. No other propositional variables are allowed. This can be expressed in $\mathsf{unaryMTL}$ without the X modality in a straightforward way.

(2) There are two consecutive positions in the data word where $\mathcal{I}_n$ holds. The data word ends after the second occurrence of $\mathcal{I}_n$.
   - $\mathsf{F}(\mathcal{I}_n \wedge \mathsf{F}(\mathcal{I}_n))$ (There are two different positions where $\mathcal{I}_n$ holds.)
   - $\mathsf{G}(\mathcal{I}_n \to \mathsf{G}(\mathcal{I}_n \to \mathsf{G}\mathtt{false}))$ (After the second occurrence of $\mathcal{I}_n$ the data word ends.)
   - $\mathsf{G}(\mathcal{I}_n \to \neg\mathsf{F}(\# \vee r_1 \vee \varphi_{\mathsf{idz}}))$ (After the first occurrence of $\mathcal{I}_n$, the symbols $\#, r_1, \mathcal{I}_1, \dots, \mathcal{I}_{n-1}$ should never occur again. Hence, by (1) the only propositional variables that may occur after the first occurrence of $\mathcal{I}_n$ are $\mathcal{I}_n$ and r_2. This implies that the two occurrences of $\mathcal{I}_n$ (whose existences are guaranteed by the first formula, are consecutive.)

The following formulae express important conditions on the data values.

(3) The data values in the positions where # holds are strictly monotonic and increase progressively by exactly 1.
   - $\mathsf{G}(\# \to \neg\mathsf{F}_{\leq 0}\#)$ (The data value at a position where # holds can never be smaller than or equal to the data value at a preceding position where # holds.)

  - $\mathsf{G}((\# \wedge \mathsf{F}\varphi_{\mathsf{idz}}) \rightarrow \mathsf{F}_{=1}\#)$ (If after the # symbol with data value $d_\#$ an instruction different from $\mathcal{I}_n$ is occurring, *i.e.*, by (2) we have not reached the last configuration, then there will finally be a further # with data value $d_\# + 1$. This and the first formula imply that the *next* occurrence of # has data value $d_\# + 1$.)

(4) The data values at positions where r_1 holds are weakly monotonic. (Similarly for r_2.)
  - $\mathsf{G}(r_1 \rightarrow \neg\mathsf{F}_{<0} r_1)$

(5) The data values at positions where # holds serve as a reference value for 0. Hence the data values at positions where r_1 hold should always be greater than or equal to this value. (Similarly for r_2.)
  - $\mathsf{G}(\# \rightarrow \neg\mathsf{F}_{<0} r_1)$.

Using these conditions, we can express the remaining details of the structure of a data word encoding a computation of $\mathcal{M}$.

(6) The data word should start with the prefix $(\#, s)(\{\mathcal{I}_1, r_1\}, s)(\{\mathcal{I}_1, r_2\}, s)$. The unaryMTL formula is of the form $\# \wedge X_{=0}(\mathcal{I}_1 \wedge r_1 \wedge \mathsf{X}_{=0}(\mathcal{I}_1 \wedge r_2))$. In order to express this condition in unaryTPTL^1 without the X modality, we have to consider more elaborate formulae:
  - $\# \wedge x.\mathsf{F}(\mathcal{I}_1 \wedge r_1 \wedge x = 0 \wedge \mathsf{F}(\mathcal{I}_1 \wedge r_2 \wedge x = 0))$ (After the first # with data value $d_\#$, there will be some $\{\mathcal{I}_1, r_1\}$ and data value $d_\#$, followed by some $\{\mathcal{I}_1, r_2\}$ with data value $d_\#$.)
  - For $i = 1, 2$, we define $\# \wedge x.\mathsf{G}(r_i \rightarrow \neg\mathsf{F}(r_i \wedge \mathsf{F}(\# \wedge x = 1)))$ (After the first # with data value $d_\#$, there cannot be two different r_i before another # with data value $d_\# + 1$ occurs.)

  The second formula and (3) express that after the first # there is at most one r_i before the next # occurs. By (5), each r_i following symbol # must have data value at least as big as that for #. Hence, the r_i with data value $d_\#$ whose existence is enforced by the first formula must occur before the second occurrence of #.

(7) In the remaining data word, the symbol # is followed by $\{r_1, \mathcal{I}_j\}$, which is followed by $\{r_2, \mathcal{I}_j\}$ for some $j \in \{1, \ldots, n\}$. This is repeated until the end of the word. In unaryMTL, this can be expressed by the formula $\mathsf{G}[\# \rightarrow \bigvee_{1 \leq i \leq n}(\mathsf{X}(\mathcal{I}_i \wedge r_1) \wedge \mathsf{XX}(\mathcal{I}_i \wedge r_2 \wedge (\mathsf{X}\# \vee \mathsf{G}\texttt{false})))]$. In unaryTPTL^1 without X modality, we define
  - $\mathsf{G}[\# \wedge \mathsf{F}\# \rightarrow x. \bigvee_{1 \leq i \leq n-1} \mathsf{F}(\mathcal{I}_i \wedge r_1 \wedge \mathsf{F}(\mathcal{I}_i \wedge r_2 \wedge \mathsf{F}(\# \wedge x = 1)))]$
    (Together with (3) this guarantees that after #, the symbol r_1 followed by symbol r_2 occur *before the next* occurrence of #.)
  - $\mathsf{G}[\# \wedge \neg\mathsf{F}\# \rightarrow \neg\mathsf{F}\varphi_{\mathsf{idz}} \wedge \mathsf{F}(\mathcal{I}_n \wedge r_1 \wedge \mathsf{F}(\mathcal{I}_n \wedge r_2))]$

  And for $i = 1, 2$, we define
  - $\mathsf{G}[\# \wedge \mathsf{F}\# \rightarrow x.\mathsf{G}(r_i \rightarrow \neg\mathsf{F}(r_i \wedge \mathsf{F}(\# \wedge x = 1)))]$
    (There cannot be two different r_i between two #.)

(8) We define the correct encoding of an increment instruction of the form $\mathcal{I}_j : C_1 := C_1 + 1;$ `go to` $\mathcal{I}_k$. Note that incrementing the first counter by 1 corresponds to incrementing the data value of r_1 by exactly 2. The value

of the second counter should not be changed, and this corresponds to incrementing the data value of r_2 by exactly 1. With unaryMTL, this can be expressed by the following formulae:

- $\mathsf{G}\langle(\mathcal{I}_j \wedge r_1) \rightarrow (\neg \mathsf{F}_{<2} r_1 \wedge \mathsf{F}_{=2} r_1)\rangle$ (Together with (4) this implies that the data value at the next occurrence of r_1 is incremented by 2.)
- $\mathsf{G}\langle(\mathcal{I}_j \wedge r_2) \rightarrow (\neg \mathsf{F}_{<1} r_2 \wedge \mathsf{F}_{=1} r_2)\rangle$ (Similarly, this and (4) imply that the data value at the next occurrence of r_2 is incremented by 1.)
- For $i = 1, 2$, define $\mathsf{G}\langle(\mathcal{I}_j \wedge r_i) \rightarrow \mathsf{XXX}(\mathcal{I}_k \wedge r_i)\rangle$.

For unaryTPTL^1 without X modality, we define

- $\mathsf{G}\langle(\mathcal{I}_j \wedge r_1) \rightarrow (x.\mathsf{G}(r_1 \rightarrow x \geq 2) \wedge x.\mathsf{F}(r_1 \wedge x = 2))\rangle$
- $\mathsf{G}\langle(\mathcal{I}_j \wedge r_2) \rightarrow (x.\mathsf{G}(r_2 \rightarrow x \geq 1) \wedge x.\mathsf{F}(r_2 \wedge x = 1))\rangle$
- $\mathsf{G}\langle\# \rightarrow x.\mathsf{G}((\mathcal{I}_j \wedge r_1 \wedge \mathsf{F}(\# \wedge x = 1)) \rightarrow (\varphi_1 \vee \varphi_2))\rangle$, where
  - $\varphi_1 = \mathsf{F}(\# \wedge x = 1 \wedge \mathsf{F}(\mathcal{I}_k \wedge \mathsf{F}(\# \wedge x = 2)))$, and
  - $\varphi_2 = \mathsf{F}(\# \wedge x = 1 \wedge \neg \mathsf{F}\# \wedge \mathsf{F}\mathcal{I}_k)$.

(9) We define the correct encoding of instructions of the form $\mathcal{I}_j$: `if` $C_1 = 0$ `then go to` $\mathcal{I}_k$ `else` $C_1 := C_1 - 1$; `go to` $\mathcal{I}_m$: Recall that the data value at # serves as a reference value for 0. A successful zero test of the first counter (and no change in the value of the first counter) can thus be defined in unaryMTL as follows:

- $\mathsf{G}((\# \wedge \mathsf{F}_{=0}(\mathcal{I}_j \wedge r_1)) \rightarrow \mathsf{F}_{=1}(\mathcal{I}_k \wedge r_1))$ (Note that (3) to (5) guarantee that the position where $\mathcal{I}_k \wedge r_1$ holds with data value incremented by 1 is directly after the next #.)

The negative zero test and decrement instruction is similar. Note that decrementing the value of the first counter corresponds to not changing the data value at r_1.

- $\mathsf{G}\langle(\# \wedge \mathsf{X}_{>0}(\mathcal{I}_j \wedge r_1)) \rightarrow \mathsf{X}(\mathcal{I}_j \wedge r_1 \wedge \mathsf{F}_{=0} r_1 \wedge \mathsf{XXX}\mathcal{I}_m)\rangle$

We further can use the same formulae as in (8) to express that the value of the other counter does not change. The unaryTPTL^1 formula without the X modality for expressing the negative zero test is a bit more elaborate. Again, (3) to (5) are crucial for ensuring that the correct positions in the data word are defined.

- $\mathsf{G}\langle\varphi_1 \rightarrow (\varphi_2 \wedge \varphi_3)\rangle$, where
  - $\varphi_1 = \# \wedge x.\mathsf{F}(\mathcal{I}_j \wedge r_1 \wedge \mathsf{F}(\# \wedge x = 1)) \wedge x.\mathsf{G}(\mathcal{I}_j \wedge r_1 \rightarrow x > 0)$, and
  - $\varphi_2 = x.\mathsf{F}\langle\mathcal{I}_j \wedge r_1 \wedge \mathsf{F}(\# \wedge x = 1) \wedge x.\mathsf{F}(r_1 \wedge x = 0)\rangle$, and
  - $\varphi_3 = x.\mathsf{F}\langle\# \wedge x = 1 \wedge \mathsf{F}(\mathcal{I}_m \wedge (\mathsf{F}(\# \wedge x = 2) \vee \neg \mathsf{F}\#))\rangle$.

This finishes the hardness proof for finitary SAT. For infinitary SAT, we do not need (2) and instead use the usual approach and define a formula which expresses that $\mathcal{I}_1$ is visited infinitely often: $\mathsf{GF}\mathcal{I}_1$. We can further simplify some of the formulae, as we do not have to check whether we have reached the halting instruction. □

Remark 4. Using Ehrenfeucht-Fraïssé-Games defined in [8], one can prove that unaryTPTL^1-formulae of the form $x.\mathsf{F}(b \wedge \mathsf{F}(c \wedge x = 0))$ cannot be expressed in MTL. We remark that it is exactly this kind of formulae that we use in the unaryTPTL^1 formulae without X modality in (6) to (9). It is an open problem whether we can express the conditions stated there in unaryMTL without using the X modality.

6 Results for Positive Fragments

Next we consider the fragment of MTL and TPTL, in which negation is restricted to propositional variables. Note that this excludes the globally operator, which seems to be crucial in the proofs for lower bounds of SAT for the mentioned logics. It also allows us to prove the following interesting property:

Theorem 5 (Finite model property). *Let $\varphi \in \mathsf{positiveTPTL}$. Then φ is satisfiable by a data word if, and only if, it is satisfiable by a finite data word.*

This implies that infinitary and finitary SAT are equivalent problems. It turns out that the restriction of negation to propositional variables does not change anything about the complexity status of SAT compared to *finitary* SAT for the corresponding full logics.

Theorem 6. *For $\mathsf{positiveMTL}$ and $\mathsf{positiveTPTL}^1$, SAT is Σ_1^0-complete. For $\mathsf{positiveFreezeLTL}^1$, SAT is not primitive recursive.*

Proof. For $\mathsf{positiveMTL}$, we reduce the halting problem for two-counter machines to SAT. Given a two-counter machine $\mathcal{M}$ with n instructions, we define a $\mathsf{positiveMTL}$ formula $\varphi_{\mathcal{M}}$ over $\mathsf{P} = \{\mathcal{I}_1, \ldots, \mathcal{I}_n, r_1, r_2\}$ that is satisfiable if, and only if, $\mathcal{M}$ has a halting computation. A computation $(J_0, c_0, d_0)(J_1, c_1, d_1)\ldots$ is encoded by a data word of the form

$$(\{J_0\}, s)(\{r_1\}, s + c_0)(\{r_2\}, s + d_0)(\{J_1\}, s)(\{r_1\}, s + c_1)(\{r_2\}, s + d_1)\ldots$$

Using this structure, we can avoid using the G modality.

For $\mathsf{positiveTPTL}^1$ and $\mathsf{positiveFreezeLTL}^1$, the proof is by reduction of SAT for the corresponding unary fragment to SAT for the corresponding positive fragment without the X modality. Let φ be, *e.g.*, a formula in $\mathsf{unaryTPTL}^1$. For this we may assume without loss of generality that φ is in *negation normal form*, where the application of negation is restricted to propositional variables. For the reduction to work, we must assume that every data word contains a special symbol halt marking the end of the *finite* data word. The idea is to exploit the fact that for finite data words, the formula $\mathsf{G}\varphi$ means that φ must hold *until* the symbol halt marks the end of the word. We define a function h mapping $\mathsf{unaryTPTL}^1$ formulae in negation normal form into $\mathsf{positiveTPTL}^1$ formulae. The definition is by induction on the construction of a formula, we only give the interesting cases: $h(\mathsf{G}\varphi) := (h(\varphi) \wedge \neg\mathsf{halt})\mathsf{U}\mathsf{halt}$, $h(\mathsf{F}\varphi) := (\mathtt{true} \wedge \neg\mathsf{halt})\mathsf{U}(h(\varphi) \wedge \neg\mathsf{halt})$. We have φ is satisfiable if, and only if, $h(\varphi)$ is satisfiable.

Note that this proof idea is not trivially applicable to $\mathsf{positiveMTL}$, because it is not clear how to express the semantics of the $\mathsf{G}_{[a,b]}$-modality for $[a, b] \neq (-\infty, +\infty)$. $\square$

Last but not least, we consider the unary fragment of $\mathsf{positiveTPTL}^1$, in which the only allowed modalities are the F and X modalities. This fragment has also been considered for MTL and TPTL over monotonic timed words [7]. In this setting, SAT for both logics is NP-complete. Here, we show that this applies also to the setting of non-monotonic data words.

Theorem 7. *For* posUnaryTPTL, *SAT is* NP*-complete.*

Proof. (Sketch) The lower bound follows by reduction from SAT for propositional logic. Now we prove that the problem is in NP. Let ψ be a posUnaryTPTL formula. We denote by $\Gamma(\psi)$ the set containing all those formulae that can be obtained from ψ by resolving the non-determinism induced by the occurrences of disjunctions. Formulae in $\Gamma(\psi)$ are is satisfiable if, and only if, there exists a satisfiable formula ψ^* in $\Gamma(\psi)$.

Now, consider a formula $\psi^* \in \Gamma(\psi)$ that is satisfied by a data word w. One can see that only $n \leq |\psi^*|$ points in w are relevant for a successful model checking of w on ψ^*. This is because new further points are only required by subformulae of the form $\mathsf{F}\phi$ or $\mathsf{X}\phi$ and negation only occurs in front of atoms. Therefore, a word w' of length n can be obtained from w such that $w' \models \psi^*$:

Lemma 8. *Let ψ be a* posUnaryTPTL *formula. If ψ is satisfiable, then there exists a data word w such that $w \models \psi$ and $|w| \leq |\psi|$.*

The difference to Th. 5 is that we are able to additionally provide a bound to the length of the possible finite witness of satisfiability.

Based on this, we can decide satisfiability in polynomial non-deterministic time: guess a formula ψ^* in $\Gamma(\psi)$; then guess a data word $w = (P_0, 0) \dots (P_{n-1}, 0)$ of length $n = |\psi^*|$; last, for each subformula ϕ of ψ^* guess a position from $\{0 \dots n-1\}$ and verify that w model checks ψ^* (without considering data value constraints) with respect to the guessed positions. Finally, solve the set of linear inequalities $\mathcal{C}$ built up from the position assigned to each subformula $x \sim c$ and its corresponding less outer subformula $x.\phi$ (in case it does not exists, then ψ^* is used). All the guesses are independent and the verifications can be done in polynomial time. Each inequality in $\mathcal{C}$ belongs to the class of *difference constraints* and a system of such a class of constraints can be solved in polynomial time [9]. Thus, the problem is in NP. $\square$

Corollary 9. *For* posUnaryMTL *and* posUnaryFreezeLTL, *SAT is* NP*-complete.*

7 Conclusion and Open Problems

The main open problem of this paper is the decidability status for the unary fragment of MTL without the X modality. While the X modality can be avoided in reductions for showing undecidability of SAT for unaryTPTL, 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 X modality could be enough to change the decidability status of the SAT problem.

We are also surprised that the decidability of SAT for unaryFreezeLTL does neither apply to unaryMTL nor to unaryTPTL. This is opposed to a recent extension of a decidability result for $\mathsf{FreezeLTL}^1$-model checking *deterministic* one-counter automata [11] to MTL and TPTL^1 [17].

Note that our undecidability results for SAT of different fragments of MTL and TPTL, also imply the undecidability of the existential model checking and (apart from the positive fragments) of the universal model checking problem for one-counter machines and the corresponding logics.

References

1. Alur, R., Henzinger, T.A.: Real-time logics: Complexity and expressiveness. Inf. Comput. 104(1), 35–77 (1993)
2. Alur, R., Henzinger, T.A.: A really temporal logic. J. ACM 41(1), 181–204 (1994)
3. Bojanczyk, M., David, C., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on data words. ACM Trans. Comput. Log. 12(4), 27 (2011)
4. Bollig, B.: An automaton over data words that captures EMSO logic. In: Katoen, J.-P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 171–186. Springer, Heidelberg (2011)
5. Bollig, B., Cyriac, A., Gastin, P., Narayan Kumar, K.: Model checking languages of data words. In: Birkedal, L. (ed.) FOSSACS 2012. LNCS, vol. 7213, pp. 391–405. Springer, Heidelberg (2012)
6. Bouyer, P.: A logical characterization of data languages. Inf. Process. Lett. 84(2), 75–85 (2002)
7. Bouyer, P., Chevalier, F., Markey, N.: On the expressiveness of TPTL and MTL. Inf. Comput. 208(2), 97–116 (2010)
8. Carapelle, C., Feng, S., Fernandez Gil, O., Quaas, K.: Ehrenfeucht-Fraïssé games for TPTL and MTL over data words, http://arxiv.org/abs/1311.6250
9. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, Second Edition, 2nd edn. The MIT Press and McGraw-Hill Book Company (2001)
10. Demri, S., Lazic, R.: LTL with the freeze quantifier and register automata. ACM Trans. Comput. Log. 10(3) (2009)
11. Demri, S., Lazić, R.S., Sangnier, A.: Model checking Freeze LTL over one-counter automata. In: Amadio, R.M. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 490–504. Springer, Heidelberg (2008)
12. Figueira, D., Segoufin, L.: Future-looking logics on data words and trees. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 331–343. Springer, Heidelberg (2009)
13. Jaffar, J., Maher, M.J., Stuckey, P.J., Yap, R.H.C.: Beyond finite domains. In: Borning, A. (ed.) PPCP 1994. LNCS, vol. 874, pp. 86–94. Springer, Heidelberg (1994)
14. Minsky, M.L.: Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. Annals of Mathematics 74(3), 437–455 (1961)
15. Ouaknine, J., Worrell, J.B.: On metric temporal logic and faulty Turing machines. In: Aceto, L., Ingólfsdóttir, A. (eds.) FOSSACS 2006. LNCS, vol. 3921, pp. 217–230. Springer, Heidelberg (2006)
16. Ouaknine, J., Worrell, J.: On the decidability and complexity of metric temporal logic over finite words. Logical Methods in Computer Science 3(1) (2007)
17. Quaas, K.: Model checking metric temporal logic over automata with one counter. In: Dediu, A.-H., Martín-Vide, C., Truthe, B. (eds.) LATA 2013. LNCS, vol. 7810, pp. 468–479. Springer, Heidelberg (2013)
18. Segoufin, L.: Automata and logics for words and trees over an infinite alphabet. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 41–57. Springer, Heidelberg (2006)