The Expressive Power, Satisfiability and Path Checking Problems of MTL and TPTL over Non-Monotonic Data Words

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

Doctor rerum naturalium

(Dr. rer. nat.)

im Fachgebiet
Informatik

Vorgelegt

von Shiguang Feng

geboren am 17.01.1984 in Shandong, China

Die Annahme der Dissertation wurde empfohlen von:

1. Prof. Dr. Markus Lohrey, Universität Siegen

2. Prof. Dr. Manfred Droste, Universität Leipzig

3. Prof. Dr. Martin Lange, Universität Kassel

Die Verleihung des akademischen Grades erfolgt mit Bestehen

der Verteidigung am 19.04.2016 mit dem Gesamtprädikat

magna cum laude

Acknowledgements

First of all, I am most grateful to my supervisor Prof. Dr. Markus Lohrey for his patient and continuous support throughout my work. His illuminating thoughts and insightful remarks have contributed to this thesis in many ways. His generosity of time and guidance has deeply informed my development. I have gained much from collaboration with him.

I would like to thank the Deutsche Forschungsgemeinschaft (DFG) Research Training Group 1763 "Quantitative logics and automata" for giving me the opportunity to pursue my interests in an inspiring international community and the financial support for me during the last three years. I also want to thank my second supervisor Prof. Dr. Manfred Droste for providing an excellent working and learning environment in Universität Leipzig.

In addition, I am grateful to my colleague Dr. Karin Quaas for introducing me to the wonderful topic of temporal logic over non-monotonic data words. Many ideas for this work are derived from the discussion with her. I have benefited a lot from her insight and patience.

Finally, I would like to thank my colleagues Claudia Carapelle and Oliver Fernández Gil for their interest and help in my work. The work in this thesis has been co-authored and discussed with them. I also want to thank Parvaneh Babari Ghorghi, Vitaly Perevoshchikov, Eric Nöth, Alexander Kartzow and my former Supervisor Prof. Xishun Zhao and colleague Dr. Yuping Shen for their kind help in my work and daily life.

Table of contents

Chapter 1

Introduction

Recently, verification and analysis of sets of data words have gained a lot of interest [12–14, 17, 28, 30, 76]. A data word is a sequence over $\mathbf{P} \times D$, where $\mathbf{P}$ is a finite set of labels, and D is a set of data values. One prominent example of data words are timed words, used in the analysis of real-time systems [3]. Linear-time temporal logic (LTL) is nowadays one of the main logical formalisms used for the specification and verification of reactive systems, and has found applications in industrial tools. In this context, satisfiability and model checking are the main computational problems for LTL. The complexity of these problems in various settings is well-understood, see e.g. [10] for background. Triggered by applications in real-time systems, various timed extensions of LTL have been invented. Two of the most prominent examples are metric temporal logic (MTL) [54] and timed propositional temporal logic (TPTL) [7]. In MTL, the temporal operator until (U) is indexed by a time interval. For instance, the formula $p \,\mathsf{U}_{[2,3)}\, q$ holds at a certain 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 formalism. It uses register variables, which can be set to the current time value and later these register variables can be compared with the current time value. For instance, the above 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: The difference of the current time value and the value stored in x is in the interval $[2,3)$.

For both MTL and TPTL, two different semantics exist: the continuous semantics, where the time domain are the real numbers, and the discrete semantics, where the time domain are the natural numbers. We will be only interested in the discrete semantics, where formulas are evaluated over finite or infinite sequences $(P_0, d_0)(P_1, d_1)\ldots$ of pairs (P_i, d_i). Here $P_i \subseteq \mathbf{P}$ is a finite set of atomic propositions (from some pre-specified finite set $\mathbf{P}$) and $d_i \in \mathbb{N}$ is a time stamp such that $d_i \leq d_{i+1}$ for all $i \geq 0$. The freeze mechanism from TPTL has also received attention in connection with non-monotonic data words. A non-monotonic data word

is a finite or infinite sequence $(P_0, d_0)(P_1, d_1)\ldots$ of the above form, where we do not require the data values d_i to be monotonic. In timed words, intuitively, the sequence of data values describes the timestamps at which the properties from the labels set **P** hold. Non-monotonic sequences of natural numbers, instead, can model the variation of an observed value during a time elapse: we can think of the heartbeat rate recorded by a cardiac monitor, atmospheric pressure, humidity or temperature measurements obtained from a meteorological station. For example, let $\mathsf{Weather} = \{\mathsf{sunny}, \mathsf{cloudy}, \mathsf{rainy}\}$ be a set of labels. A data word modeling the changing of the weather and highest temperature day after day could be:

$$(\mathsf{rainy}, 10)(\mathsf{cloudy}, 8)(\mathsf{sunny}, 12)(\mathsf{sunny}, 13)\ldots$$

Applications for MTL and TPTL over non-monotonic data values can be seen in areas, where data streams of discrete values have to be analyzed and the focus is on the dynamic variation of the values (e.g. streams of discrete sensor data or stock charts). Both logics, however, have not gained much attention in the specification of non-monotonic data words, albeit they can express many interesting properties. The goal of this thesis is to investigate MTL and TPTL when evaluated over non-monotonic data words: the expressive power, satisfiability problem and path checking problem.

To continue our example, using the TPTL-formula $x.(\mathsf{sunny}\ \mathsf{U}\ (\mathsf{cloudy} \wedge -3 \leq x \leq -1))$ over the labels set Weather, we can express the following property: It is sunny until it becomes cloudy and the highest temperature has decreased of 1 to 3 degrees. This formula is equivalent to the MTL-formula $(\mathsf{sunny}\ \mathsf{U}_{[-3,-1]}\ \mathsf{cloudy})$. The main advantage of MTL with respect to TPTL is its concise syntax. It would be practical if we could show that MTL equals TPTL over data words. It is a simple observation that every MTL-formula can be translated into an equivalent TPTL-formula with only one register variable. For the other direction, however, it turns out that the result depends on the data domain. For monotonic data words over the natural numbers, Alur and Henzinger [6] proved that MTL and TPTL are equally expressive. For timed words over the non-negative reals, instead, Bouyer et al. [18] showed that TPTL is strictly more expressive than MTL. We consider the relative expressive power of TPTL and MTL over non-monotonic data words, and show that TPTL is strictly more expressive than MTL in this setting.

Satisfiability and model checking problem for MTL and TPTL have been studied intensively in the past, see e.g. [6, 7, 20, 21, 28, 31, 56, 63, 65]. On monotonic data words over the natural numbers, the satisfiability problem for both MTL and TPTL is EXPSPACE-complete, and is undecidable for TPTL over non-monotonic data words [6, 7]. However, over timed words, the satisfiability for both logics is undecidable over infinite timed words [6, 63], there is a difference in the finite timed words case: TPTL has undecidable satisfiability problem [6],

while satisfiability for MTL is decidable (but non-primitive recursive) [65]. The type (finite or infinite) of data words has influence on the decidability of satisfiability and the complexity of model checking. In this thesis, we consider MTL and TPTL over infinite data words and finite data words, respectively. We show that over non-monotonic data words, for MTL and most fragments of MTL and TPTL, the satisfiability problem is undecidable, which is either Σ_1^1-complete or Σ_1^0-complete depending on the data words in consideration are infinite or finite.

As for TPTL, the logic freezeLTL can store the current data value in a register x. But in contrast to TPTL, the value of x can only be compared for equality with the current data value. Model checking one-counter machines with freezeLTL is in general undecidable [30], and so is the satisfiability problem [28]. A good number of recent publications deal with decidable and undecidable fragments of freezeLTL [28–32]. The authors of [31] consider one-counter machines (OCM) as a mechanism for generating infinite non-monotonic data words, where the data values are the counter values along the unique computation path. Whereas freezeLTL model checking for non-deterministic OCM is Σ_1^1-complete, the problem becomes PSPACE-complete for deterministic OCM [31]. We investigate the complexity of path checking problems for MTL and TPTL over non-monotonic data words. These data words can be either finite or infinite periodic. Non-monotonic data words can be considered as behavioral models of one-counter machines. Our results strengthens the recent decidability result for model checking of TPTL over deterministic OCM [69], and also generalizes the PSPACE-completeness result for freezeLTL over deterministic OCM from [31].

Below we give a brief description of the contents of this thesis.

In Chapter 2, we give some basic definitions and notations about data words, metric temporal logic, timed propositional temporal logic, the relative expressive power, computational complexity and two-counter machines.

In Chapter 3, we study the relative expressive power of MTL and TPTL, and the expressive power of several fragments of MTL and TPTL by restriction of the syntactic resources, e.g., the number of register variables, the until rank and the set of constraint numbers (or interval borders). As a main tool for showing the results, we introduce quantitative versions of Ehrenfeucht-Fraïssé games for MTL and TPTL over data words. In model theory, EF-game is mainly used to prove inexpressibility results for some logics [34], e.g., first-order logic, monadic second-order logic. Etessami and Wilke [36] introduced the EF-game for LTL and used it to show that the until hierarchy for LTL is strict. Quantitative EF-games provide a very general and intuitive mean to prove results concerning the expressive power of quantitative logics. Using the EF-game for MTL, we show that TPTL is strictly more expressive than MTL over both infinite data words and finite data words. In [18], Bouyer et al. used the formula $x.\mathsf{F}(b \wedge \mathsf{F}(c \wedge x \leq 2))$ to separate these two logics over timed words. We show that the simpler

TPTL-formula $x.\mathsf{XX}(x = 0)$ is not definable in MTL. Note that this formula is in the unary fragment of freezeLTL and uses only one register variable, which is very restrictive. Actually, we prove TPTL^1 is strictly more expressive than MTL. The intuitive reason for the difference in expressiveness is that, using register variables, we can store data values at any position of a word to compare them with a later position, and it is possible to check that other properties are verified in between. This cannot be done using the constrained temporal operators in MTL. This does not result in a gap in expressiveness in the monotonic data words setting, because the monotonicity of the data sequence does not allow arbitrary data values between two positions of a data word. Furthermore, we show that the MTL definability problem: whether a TPTL-formula is definable in MTL, is undecidable. We prove the undecidability over infinite and finite data words by reductions of recurrent state problem and halting problem of two-counter machines, respectively.

The register variables in TPTL play an important role in reaching its greater expressive power compared to MTL. When restricting the number of register variables, we are able to show that there is a strict increase in expressiveness when allowing two register variables instead of just one, i.e., TPTL^2 is strictly more expressive than TPTL^1. We obtain this result by proving the TPTL^2-formula $x_1.\mathsf{X}(x_1 > 0 \wedge x_2.\mathsf{F}(x_1 > 0 \wedge x_2 < 0))$ is not definable in TPTL^1. But it is still open for the general case that whether TPTL^{r+1} is strictly more expressive than TPTL^r when $r \geq 2$. We conjecture that the hierarchy about the number of register variables for TPTL is strict.

We also consider the expressive power of several fragments of MTL and TPTL by restriction of the until rank and the set of constraint numbers (or interval borders). We show that the until rank hierarchies for MTL and TPTL are strict over both infinite and finite data words. Similar to the MTL definability problem, for every $k \in \mathbb{N}$, whether an MTL-formula (respectively, TPTL-formula) is definable in MTL_k (respectively, TPTL_k) is also undecidable. When the set of constraint numbers (or interval borders) is restricted, we obtain linear constraint hierarchies and lattice constraint hierarchies for both MTL and TPTL.

There is an alternative definition for MTL that uses the non-strict semantics for the until modality. We can show that, over non-monotonic data words, MTL with strict semantics is strictly more expressive than MTL with non-strict semantics, whereas these two logics are equivalent over monotonic data words.

In Chapter 4, we study the satisfiability of MTL and TPTL and some fragments of them over non-monotonic data words. More detailed, we consider satisfiability over infinite data words (infinitary SAT) and finite data words (finitary SAT), respectively. We show that for MTL, the unary fragment of MTL and the pure fragment of MTL, infinitary SAT is Σ_1^1-complete and finitary SAT is Σ_1^0-complete. This still holds even for the unary fragment of

MTL with two propositions and for the unary fragment of TPTL[1] without the X modality. This is opposed to the decidability result for freezeLTL with one register variable evaluated over finite data words [28]. However, it is an open problem whether undecidability also holds for the unary fragment of MTL in which the X modality is not allowed. We prove the undecidability of infinitary SAT (respectively, finitary SAT) by a reduction from the recurrent state problem (respectively, halting problem) of two-counter machines.

We also consider another syntactic restriction of the logics, namely we restrict the negation operator to propositions and constraint formulas, 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. For the positive fragments of MTL and TPTL, we show that a positive formula is satisfiable if and only it is satisfied by a finite data word. Finitary SAT and infinitary SAT coincide for positive MTL and positive TPTL. Both of them are Σ_1^0-complete. Last but not least, we study the unary positive fragments of MTL and TPTL (called existential fragment in [18]). For existential TPTL and existential MTL, we show that SAT is NP-complete.

The main insight of this chapter 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 extensions of temporal logics.

In Chapter 5, we investigate the complexity of path checking problems for MTL and TPTL over non-monotonic data words. These data words can be either finite or infinite periodic; in the latter case the data word is specified by two finite data words $u = (P_1, d_1) \cdots (P_m, d_m)$ and $v = (Q_1, e_1) \cdots (Q_n, e_n)$, which are the initial part and the period, respectively, and an offset number K. The resulting infinite data word is $u \prod_{i \geq 0} (v + iK)$, where $v + M$ denotes the data word $(Q_1, e_1 + M) \cdots (Q_n, e_n + M)$. It can be easily seen that the infinite data word produced by a deterministic OCM is such a periodic data word. For periodic words without data values, the complexity of LTL path checking belongs to $\mathsf{AC}^1(\mathsf{LogDCFL})$ (a subclass of NC) [55]. This result solved a long standing open problem. For finite monotonic data words, the same complexity bound has been shown for MTL in [21].

We show that the latter result of [21] is quite sharp in the following sense: Path checking for MTL over non-monotonic (finite or infinite) data words as well as path checking for TPTL with one register variable over monotonic (finite or infinite) data words is P-complete. Moreover, path checking for TPTL (with an arbitrary number of register variables) over finite as well as infinite periodic data words becomes PSPACE-complete. We also show that PSPACE-hardness already holds for the fragment of TPTL with only two register variables and all constraint numbers are encoded in unary notation. If we only consider finite data words and

the number of register variables is bounded by r ($r \in \mathbb{N}$), then the complexity of path checking for TPTL^r becomes P-complete.

For MTL, we prove the P-hardness over non-monotonic data words. This is unavoidable by the result in [21] that path checking for MTL over monotonic data words belongs to $\mathsf{AC}^1(\mathsf{logDCFL})$. We define the logic SMTL which is a succinct version of MTL and has the same expressive power as MTL. For SMTL, we show that patch checking over monotonic data words is P-complete. We also show that path checking for MTL over infinite monotonic periodic data words of the form $(u)^{\omega}_{+k}$ (i.e., without the initial part) belongs to $\mathsf{AC}^1(\mathsf{logDCFL})$. All these results yield a rather complete picture on the complexity of path checking for MTL and TPTL.

Since the infinite data word produced by a deterministic OCM is periodic, we can transfer all complexity results for the infinite periodic case to deterministic OCM. In [69], the author proved recently that model checking for non-monotonic TPTL over deterministic OCM is decidable, but the complexity remained open. Our results show that the precise complexity is PSPACE-complete. This also generalizes the PSPACE-completeness result for freezeLTL over deterministic OCM in [31].

Chapter 2

Preliminaries

In this chapter, we give some basic definitions and notations about data words, temporal logics, two-counter machines and computational complexity.

2.1 Data words

We use $\mathbb{Z}$ and $\mathbb{N}$ to denote the set of integers and the set of natural numbers, respectively. Let $\mathbf{P}$ be a finite set of *atomic propositions*. A *word* over $\mathbf{P}$ is a finite or infinite sequence $P_0P_1\cdots$, where $P_i \subseteq \mathbf{P}\,(i \in \mathbb{N})$. A *data word* over $\mathbf{P}$ is a finite or infinite sequence $(P_0,d_0)(P_1,d_1)\cdots$, where $(P_i,d_i) \in (2^{\mathbf{P}} \times \mathbb{N})\,(i \in \mathbb{N})$. It is *monotonic* (*strictly monotonic*), if $d_i \leq d_{i+1}$ ($d_i < d_{i+1}$) for all $i \in \mathbb{N}$. It is *pure*, if $P_i = \emptyset$ for all $i \in \mathbb{N}$. A pure data word is just written as a sequence of natural numbers. We denote with $(2^{\mathbf{P}} \times \mathbb{N})^*$ and $(2^{\mathbf{P}} \times \mathbb{N})^\omega$, respectively, the set of finite and infinite, respectively, data words over $\mathbf{P}$. Let u be a data word. We use $\min(u)$ and $\max(u)$ to denote *the minimal data value* and *the maximal data value* in u, respectively. If there is no maximal data value in u, we set $\max(u) = +\infty$. We use $|u|$ to denote *the length of* u, i.e., the number of all pairs (P_i,d_i) in u. For example, $|(P_0,d_0)(P_1,d_1)\cdots(P_n,d_n)| = n+1$. If u is an infinite data word, we set $|u| = +\infty$. Let u be a finite data word. We use $||u||$ to denote *the size of* u, i.e., the number of all symbols occurring in u.

Given a data word $u = (P_0,d_0)(P_1,d_1)\cdots$, we use the notations $u[i] := (P_i,d_i)$,

$$\begin{aligned} u[i:] &:= (P_i,d_i)(P_{i+1},d_{i+1})\cdots, \\ u[:i] &:= (P_0,d_0)(P_1,d_1)\cdots(P_i,d_i), \\ u[i:j] &:= (P_i,d_i)(P_{i+1},d_{i+1})\cdots(P_j,d_j), \end{aligned}$$

and $u_{+k} := (P_0,d_0+k)(P_1,d_1+k)\cdots$, where $k \in \mathbb{N}$. We use u_1u_2 to denote the concatenation of two data words u_1 and u_2, where u_1 has to be finite. For a finite data word u and numbers

$n, k \in \mathbb{N}$, we define

$$u^n_{+k} := u\,u_{+k}\,u_{+2k}\,u_{+3k}\cdots u_{+(n-1)k},$$
$$u^\omega_{+k} := u\,u_{+k}\,u_{+2k}\,u_{+3k}\cdots.$$

In the pair (P_i, d_i), if P_i is a singleton set $\{p\}$, we write it (p, d_i) for brevity. Let w_0, w_1 be two data words. We will use $P_{i,j}$ and $d_{i,j}$ to denote the set of propositions and data value in the position j of data word w_i ($i \in \{0,1\}, j \in \mathbb{N}$), respectively.

For words w, w_1, w_2 and numbers $i, n \in \mathbb{N}$, the notations $w[i]$, $w[:i]$, $w[i:]$, $w_1(w_2)^n w$ and $w_1(w_2)^\omega$ are defined in the expected way, where w_1, w_2 have to be finite.

2.2 Linear temporal logic

The set of formulas of linear-time temporal logic (LTL) is built up from **P** by Boolean connectives, and the *until* modality U using the following grammar:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \mathsf{U} \varphi$$

where $p \in \mathbf{P}$.

Formulas of LTL are interpreted over *words*. Let $w = P_0 P_1 P_2 \cdots$ be a word, and let i be a position in w. We define the *satisfaction relation for* LTL inductively as follows:

- $(w,i) \models \top$.
- $(w,i) \models p$ if and only if $p \in P_i$.
- $(w,i) \models \neg\varphi$ if and only if $(w,i) \not\models \varphi$.
- $(w,i) \models \varphi_1 \wedge \varphi_2$ if and only if $(w,i) \models \varphi_1$ and $(w,i) \models \varphi_2$.
- $(w,i) \models \varphi_1 \mathsf{U} \varphi_2$ if and only if there exists a position $j > i$ in w such that $(w,j) \models \varphi_2$, and $(w,t) \models \varphi_1$ for all positions t with $i < t < j$.

We say that a word *satisfies* an LTL-formula φ, written $w \models \varphi$, if $(w,0) \models \varphi$. We use the following standard abbreviations:

$$\bot := \neg\top \qquad \mathsf{F}\varphi := \top\mathsf{U}\varphi$$
$$\varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi_2) \qquad \mathsf{G}\varphi := \neg\mathsf{F}\neg\varphi$$
$$\varphi_1 \rightarrow \varphi_2 := \neg\varphi_1 \vee \varphi_2 \qquad \mathsf{X}\varphi := \bot\mathsf{U}\,\varphi$$
$$\varphi_1\,\mathsf{R}\,\varphi_2 := \neg(\neg\varphi_1\mathsf{U}\neg\varphi_2) \qquad \mathsf{X}^m\varphi := \underbrace{\mathsf{X}\cdots\mathsf{X}}_{m}\varphi$$

The modalities X (*next*), F (*eventually*) and G (*globally*) are all *unary* operators, which refer to the next position, some position in the future and all positions in the future, respectively. The modality R is the *release* operator, which is useful to transform a formula into negation normal form, i.e, the negation operator ($\neg$) is only applied to $\top$ or atomic propositions.

Example 1. Let w be a word, i a position in w, and let $\mathsf{X}\varphi$, $\mathsf{F}\varphi$, $\mathsf{G}\varphi$ and $\varphi_1\mathsf{R}\varphi_2$ be LTL-formulas. Then

- $(w,i) \models \mathsf{X}\varphi$ if and only if i is not the last position of w and $(w,i+1) \models \varphi$.
- $(w,i) \models \mathsf{F}\varphi$ if and only if there exists a position $j > i$ in w such that $(w,j) \models \varphi$.
- $(w,i) \models \mathsf{G}\varphi$ if and only if for all positions $j > i$ in w, $(w,j) \models \varphi$.
- $(w,i) \models \varphi_1\mathsf{R}\varphi_2$ if and only if either for all positions $j > i$, $(w,j) \models \varphi_2$, or there is a position $j' > i$ such that $(w,j') \models \varphi_1$, and for all positions t with $i < t \leq j'$, $(w,t) \models \varphi_2$.

It is easy to check that, if w is a finite word, then for every formula φ, the formulas $\mathsf{X}\varphi$ and $\mathsf{F}\varphi$ are always false and $\mathsf{G}\varphi$ is always true at the last position of w.

We define two quantitative extensions of LTL: MTL and TPTL, which are evaluated over *data words*, in the following.

2.3 Metric temporal logic

Metric temporal logic (MTL) is an extension of LTL where the until modality U is augmented with a constraint interval over $\mathbb{Z}$. More precisely, the formulas of MTL are built by the following grammar:

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

where $I \subseteq \mathbb{Z}$ is an open, closed or half-closed interval. We use pseudo-arithmetic expressions to denote intervals. For instance, $= 2$ and ≥ 1 denote the intervals $[2,2]$ and $[1,\infty)$, respectively. If $I = \mathbb{Z}$, then we may omit the index I in U_I.

Formulas of MTL are interpreted over data words. Let $w = (P_0, d_0)(P_1, d_1)\cdots$ be a data word, and let $i < |w|$. We define the satisfaction relation for MTL inductively as follows:

- $(w,i) \models \top$.
- $(w,i) \models p$ if and only if $p \in P_i$.
- $(w,i) \models \neg\varphi$ if and only if $(w,i) \not\models \varphi$.
- $(w,i) \models \varphi_1 \wedge \varphi_2$ if and only if $(w,i) \models \varphi_1$ and $(w,i) \models \varphi_2$.
- $(w,i) \models \varphi_1 \mathsf{U}_I \varphi_2$ if and only if there exists a position j with $i < j < |w|$ such that $(w,j) \models \varphi_2$, $d_j - d_i \in I$, and for all positions t with $i < t < j$, $(w,t) \models \varphi_1$.

We say that a data word satisfies an MTL-formula φ, written $w \models \varphi$, if $(w,0) \models \varphi$. We use the same syntactic abbreviations as for LTL where every temporal operator is augmented with a constraint interval, i.e., $\mathsf{X}_I\varphi := \bot\mathsf{U}_I\varphi$, $\mathsf{F}_I\varphi := \top\mathsf{U}_I\varphi$, $\mathsf{G}_I\varphi := \neg\mathsf{F}_I\neg\varphi$, $\varphi_1\mathsf{R}_I\varphi_2 := \neg(\neg\varphi_1\mathsf{U}_I\neg\varphi_2)$, and $\mathsf{X}_I^m\varphi := \underbrace{\mathsf{X}_I\cdots\mathsf{X}_I}_{m}\varphi$.

Example 2. The following formula, over the set $\mathsf{Weather} = \{\mathsf{cloudy}, \mathsf{sunny}, \mathsf{rainy}\}$ of atomic propositions, expresses the fact that the weather is sunny until it becomes cloudy and the temperature has decreased by three degrees. Furthermore in the future it will rain and the temperature will increase by at least one degree:

$$\mathsf{sunny}\ \mathsf{U}_{=-3}\ (\mathsf{cloudy} \wedge \mathsf{F}_{\geq 1}\ \mathsf{rainy}). \tag{2.1}$$

We say that an MTL-formula φ is *pure* if there are no atomic propositions in φ. The *pure fragment of* MTL, denoted by pureMTL, is the set of all pure MTL-formulas. An MTL-formula is *unary* if it is built from $\top$ and atomic propositions, using the Boolean connectives, and the unary temporal modalities X_I and F_I. The *unary fragment of* MTL, denoted by unaMTL, is the set of all unary MTL-formulas. We use pureUnaMTL to denote the set of all pure unary MTL-formulas.

2.4 Timed propositional temporal logic

Let V be a countable set of *register variables*. The formulas of timed propositional temporal logic (TPTL) are built by the following grammar:

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

where $x \in V$, $c \in \mathbb{Z}$, and $\sim \in \{<, \leq, =, \geq, >\}$. We may also use formulas of the form $x \in I$ as abbreviation for conjunctions of constraints, e.g., 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 $\mathbb{Z}$. Given a register valuation ν, a data value $d \in \mathbb{Z}$, and a variable $x \in V$, we define the register valuations $\nu + d$ and $\nu[x \mapsto d]$ as follows: $(\nu + d)(y) = \nu(y) + d$ for every $y \in V$, $(\nu[x \mapsto d])(y) = \nu(y)$ for every $y \in V \setminus \{x\}$, 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 $i < |w|$. The satisfaction relation for TPTL is defined inductively as follows:

- $(w, i, \nu) \models \top$.
- $(w, i, \nu) \models p$ if and only if $p \in P_i$.
- $(w, i, \nu) \models \neg\varphi$ if and only if $(w, i, \nu) \not\models \varphi$.
- $(w, i, \nu) \models \varphi_1 \wedge \varphi_2$ if and only if $(w, i, \nu) \models \varphi_1$ and $(w, i, \nu) \models \varphi_2$.
- $(w, i, \nu) \models \varphi_1 \mathsf{U} \varphi_2$ if and only if there exists a position j with $i < j < |w|$ such that $(w, j, \nu) \models \varphi_2$, and for all positions t with $i < t < j$, $(w, t, \nu) \models \varphi_1$.
- $(w, i, \nu) \models x \sim c$ if and only if $d_i - \nu(x) \sim c$.
- $(w, i, \nu) \models x.\varphi$ if and only if $(w, i, \nu[x \mapsto d_i]) \models \varphi$.

Intuitively, $x.\varphi$, means that we are *resetting* x to the current data value, and $x \sim c$ means that, compared to the last time that we reset x, the data value has increased or decreased at least by c. We say that a data word w satisfies a TPTL-formula φ, written $w \models \varphi$, if $(w, 0, \overline{0}) \models \varphi$, where $\overline{0}$ denotes the valuation that maps all register variables to the initial data value d_0.

We use the same syntactic abbreviations as for LTL. The pure and unary TPTL-formulas are defined similarly to the pure and unary MTL-formulas, respectively, but in which we can use the constraints $x \sim c$. We use pureTPTL, unaTPTL and pureUnaTPTL to denote the set of all pure TPTL-formulas, unary TPTL-formulas and pure unary TPTL-formulas, respectively. Moreover, we define freezeLTL to be the set of all TPTL-formulas that are obtained by allowing only constraints of the form $x = 0\,(x \in V)$. Given $r \in \mathbb{N}$, we use TPTL^r (respectively, $\mathsf{freezeLTL}^r$) to denote the fragment of TPTL (respectively, freezeLTL) that uses at most r different register variables $x_1, \ldots, x_r$.

Example 3. The MTL-formula 2.1 in Example 2 is equivalent to the TPTL^1-formula

$$x.[\mathsf{sunny}\ \mathsf{U}\ (x = -3 \wedge \mathsf{cloudy} \wedge x.\mathsf{F}\ (x \geq 1 \wedge \mathsf{rainy}))].$$

The formulas $x.((\mathsf{cloudy} \wedge x \leq 2)\mathsf{U}\ \mathsf{sunny})$ and $x.\mathsf{F}\ (\mathsf{cloudy} \wedge \mathsf{F}\ (\mathsf{sunny} \wedge x \leq 2))$ express the following properties:

(1) The weather will eventually become sunny. Until then it is cloudy every day and the temperature is at most two degrees higher than the temperature at the present day.

(2) It will be cloudy in the future, later it will become sunny, and the temperature will have increased by at most 2 degrees.

2.5 Expressive power

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

Let $\mathscr{L}_1$ and $\mathscr{L}_2$ be two logics. We say that an $\mathscr{L}_1$-formula φ is definable in $\mathscr{L}_2$ if there is an $\mathscr{L}_2$-formula ψ such that φ and ψ are equivalent. We say that $\mathscr{L}_2$ is *more expressive than* $\mathscr{L}_1$, written $\mathscr{L}_1 \preccurlyeq \mathscr{L}_2$, if for every $\mathscr{L}_1$-formula, there exists an equivalent $\mathscr{L}_2$-formula. $\mathscr{L}_2$ is *strictly more expressive than* $\mathscr{L}_1$, written $\mathscr{L}_1 \prec \mathscr{L}_2$ if, additionally, there is an $\mathscr{L}_2$-formula that does not have any equivalent $\mathscr{L}_1$-formula. Further, $\mathscr{L}_1$ and $\mathscr{L}_2$ are *equally expressive*, written $\mathscr{L}_1 \equiv \mathscr{L}_2$, if $\mathscr{L}_1 \preccurlyeq \mathscr{L}_2$ and $\mathscr{L}_2 \preccurlyeq \mathscr{L}_1$. $\mathscr{L}_1$ and $\mathscr{L}_2$ are *incomparable*, if neither $\mathscr{L}_1 \preccurlyeq \mathscr{L}_2$ nor $\mathscr{L}_2 \preccurlyeq \mathscr{L}_1$.

Remark 1. In the definition of LTL, we use the *strict semantics* for the until modality U, i.e., a word w satisfies the formula $\varphi_1 \mathsf{U} \varphi_2$ in a position i if and only if there is a position $j > i$ such that φ_2 holds in the position j, and φ_1 holds in all positions between i and j. The strict semantics for the until modality U is essential to derive the next modality X. There is an alternative definition for LTL that uses the *non-strict semantics* for the until modality [26, 36, 55]. In this definition, a word w satisfies the formula $\varphi_1 \mathsf{U} \varphi_2$ in a position i if and only if either φ_2 holds in the position i, or there is a position $j > i$ such that φ_2 holds in the position j and φ_1 holds in the position i and all positions between i and j. Since the next modality X is not definable by the until modality interpreted by the non-strict semantics, it is given explicitly in the syntax of LTL in this definition. More precisely, the interpretations for the next modality X and the until modality U are as follows (we use a dot over the modality operator to denote that the modality operator is interpreted by the non-strict semantics):

- $(w,i) \models \dot{\mathsf{X}}\varphi$ if and only if $i+1 < |w|$ and $(w,i+1) \models \varphi$.
- $(w,i) \models \varphi_1 \dot{\mathsf{U}} \varphi_2$ if and only if there exists a position j with $i \leq j < |w|$ such that $(w,j) \models \varphi_2$, and for all positions t with $i \leq t < j$, $(w,t) \models \varphi_1$.

There are also definitions for MTL and TPTL that use the non-strict semantics for the until modality U, which is defined in a similar way as that for LTL [6, 7]. For LTL and TPTL, both definitions are equivalent to each other with respect to their expressive power. For example, we have $\dot{\mathsf{X}}\,p \equiv \mathsf{X}\,p$, $p\,\dot{\mathsf{U}}\,q \equiv q \vee (p \wedge (p\,\mathsf{U}\,q))$, and $p\,\mathsf{U}\,q \equiv \dot{\mathsf{X}}(p\,\dot{\mathsf{U}}\,q)$, where p and q are atomic propositions. But for MTL, we can show that the logic MTL interpreted by the strict semantics is strictly more expressive than its counterpart interpreted by the non-strict semantics (see Section 3.3).

2.6 Computational complexity

A Turing machine **TM** contains an infinite tape divided into cells, and a head that can read and write symbols on the cell and move on the tape. To be more precise, a Turing machine is a 7-tuple $\langle Q, \Sigma, \Gamma, \delta, q_0, q_{\mathsf{acc}}, q_{\mathsf{rej}} \rangle$ where

(1) Q is a non-empty finite set of states,

(2) Γ is a non-empty finite set of tape symbols where the blank symbol $\sqcup \in \Gamma$,

(3) $\Sigma = \Gamma \setminus \{\sqcup\}$ is the set of input symbols,

(4) $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function,

(5) $q_0 \in Q$ is the initial state,

(6) q_{acc} is the accepting state,

(7) q_{rej} is the rejecting state.

The input of **TM** is a *finite string* $s = a_0 a_1 \ldots a_n$ where $a_i \in \Sigma (0 \leq i \leq n)$ stored in the tape. Each cell contains a symbol a_i. Initially, **TM** reads a_0 with state q_0. Then **TM** runs on s according to the transition function δ. If **TM** reads a_i with state q_i, and $\delta(a_i, q_i) = (a_j, q_j, L)$ (respectively, $\delta(a_i, q_i) = (a_j, q_j, R)$), then **TM** erases a_i and writes a_j in the same cell, and changes into state q_j and moves to the left cell (respectively, the right cell). We say that **TM** *accepts* the input string if it reaches the state q_{acc}, and *rejects* the input string if it reaches the state q_{rej}.

A *problem* is a set of strings. We say that a problem $\mathbb{P}$ is *decidable* if there exists a Turing machine $\mathbf{TM}_1$ such that for every string s, if s belongs to $\mathbb{P}$ then $\mathbf{TM}_1$ accepts s, otherwise, $\mathbf{TM}_1$ rejects s.

Let $f : \mathbb{N} \to \mathbb{N}$ be a function. We say that problem $\mathbb{P}$ is decidable in *time* $f(n)$ (respectively, *space* $f(n)$) if there exists a Turing machine $\mathbf{TM}_2$ which decides $\mathbb{P}$ such that for all

strings s, $\mathbf{TM}_2$ can accept or reject s in $f(|s|)$ many steps (respectively, using at most $f(|s|)$ many cells), where $|s|$ is the length of s, i.e., the number of symbols in s. We will denote by $\mathsf{TIME}(f(n))$ (respectively, $\mathsf{SPACE}(f(n))$) the class of all problems that are decidable in time $f(n)$ (respectively, space $f(n)$). We define

$$\mathsf{P} = \bigcup_{k\in\mathbb{N}} \mathsf{TIME}(n^k),$$
$$\mathsf{PSPACE} = \bigcup_{k\in\mathbb{N}} \mathsf{SPACE}(n^k).$$

If the Turing machine contains two tapes: an input tape and a work tape, where the input tape stores the input and is read-only, and the work tape may be read and written in the usual way, then we can define a sublinear space complexity class where only the cells used on the work tape are counted as follows:

$$\mathsf{LOGSPACE} = \bigcup_{k\in\mathbb{N}} \mathsf{SPACE}(k\cdot\log(n)).$$

A *nondeterministic* Turing machine is a variant of Turing machine where at any point in a computation it has several possibilities to proceed. More precisely, its transition function has the form

$$\delta : Q\times\Gamma \to \mathscr{P}(Q\times\Gamma\times\{L,R\})$$

where $\mathscr{P}(Q\times\Gamma\times\{L,R\})$ is the set of all subsets of $Q\times\Gamma\times\{L,R\}$. Let **TM** be a nondeterministic Turing machine. In the computation of **TM**, if it reads a_i with state q_i, and $\delta(a_i,q_i) = S$ where $S\in\mathscr{P}(Q\times\Gamma\times\{L,R\})$, then **TM** nondeterministically chooses some (a_j,q_j,D) (D is L or R) from S, and acts according to (a_j,q_j,D). We say that **TM** accept the input if there is a computation of **TM** that can reach the state q_{acc}, and reject otherwise. We will denote by $\mathsf{NTIME}(f(n))$ (respectively, $\mathsf{NSPACE}(f(n))$) the class of all problems that are decidable by a non-deterministic Turing machine in time $f(n)$ (respectively, space $f(n)$). We define

$$\mathsf{NP} = \bigcup_{k\in\mathbb{N}} \mathsf{NTIME}(n^k),$$
$$\mathsf{NL} = \bigcup_{k\in\mathbb{N}} \mathsf{NSPACE}(k\cdot\log(n)).$$

An *alternating* Turing machine is an ordinary non-deterministic Turing-machine, whose states, except for q_{acc} and q_{rej}, are divided into two sets: $Q_\exists$ (existential states) and $Q_\forall$ (uni-

versal states). In the computation of an alternating Turing machine, we say that a state q is *accepting* if (i) $q = q_{\mathsf{acc}}$ or (ii) $q \in Q_\exists$ and there exists a transition from the current configuration leading to an accepting state or (iii) $q \in Q_\forall$ and every transition from the current configuration leads to an accepting state. The machine accepts an input s if and only if the initial state over s is accepting.

We will denote by $\mathsf{ATIME}(f(n))$ (respectively, $\mathsf{ASPACE}(f(n))$) the class of all problems that are decidable by an alternating Turing machine in time $f(n)$ (respectively, space $f(n)$). We define

$$\mathsf{APTIME} = \bigcup_{k\in\mathbb{N}} \mathsf{ATIME}(n^k),$$
$$\mathsf{ALOGSPACE} = \bigcup_{k\in\mathbb{N}} \mathsf{ASPACE}(k\cdot\log(n)).$$

It is known that $\mathsf{APTIME} = \mathsf{PSPACE}$ and $\mathsf{ALOGSPACE} = \mathsf{PTIME}$ [77].

We say that a problem $\mathbb{P}$ is P-*hard* (NP-*hard* and PSPACE-*hard*, respectively) if for every problem $\mathbb{P}'$ in P (NP and PSPACE, respectively) there is a logarithmic space (polynomial time) reduction from $\mathbb{P}'$ to $\mathbb{P}$.

A problem $\mathbb{P}$ is P-*complete* (NP-*complete* and PSPACE-*complete*, respectively) if it is in P (NP and PSPACE, respectively) and is P-hard (NP-hard and PSPACE-hard, respectively).

2.7 Two-counter machines

A two-counter machine $\mathbf{M}$ contains two counters denoted by C_1 and C_2, and a finite set $\{\mathbf{I}_0,\ldots,\mathbf{I}_n\}$ of instructions, which operate on C_1 and C_2. Each instruction $\mathbf{I}_j\,(0 \le j \le n)$ is one of the following instructions, where i is 1 or 2:

(1) **increment**: $C_i := C_i + 1$; nondeterministically go to some $\mathbf{I}_k \in S_j$, where S_j is a nonempty subset of $\{\mathbf{I}_0,\ldots,\mathbf{I}_n\}$.

(2) **decrement**: if $C_i = 0$ then nondeterministically go to some $\mathbf{I}_k \in S_j^1$ else $C_i := C_i - 1$; nondeterministically go to some $\mathbf{I}_m \in S_j^2$, where S_j^1, S_j^2 are nonempty subsets of $\{\mathbf{I}_0,\ldots,\mathbf{I}_n\}$.

(3) **halt**: $\mathbf{M}$ halts.

If for each instruction $\mathbf{I}_j\,(0 \le j \le n)$ the set S_j (or S_j^1 and S_j^2) contains exactly one element, then we say that $\mathbf{M}$ is *deterministic*. A *configuration* of a two-counter machine $\mathbf{M}$ is a triple $\gamma = (\mathbf{I}_j, c_1, c_2)$, where $\mathbf{I}_j \in \{\mathbf{I}_0,\ldots,\mathbf{I}_n\}$ and $c_1, c_2 \in \mathbb{N}$. A *computation* of $\mathbf{M}$ is a finite or infinite

sequence $(\gamma_i)_{i\geq 0}$ of configurations, starting from the initial configuration $\gamma_0 = (\mathbf{I}_0, 0, 0)$, such that γ_{i+1} is the result of executing the instruction $\mathbf{I}_i$ on γ_i for all $i \geq 0$.

The *halting problem* for two-counter machines is defined as follows:

Input: a two-counter machine **M**

Output: `yes` if there exists a finite computation of **M** that reaches a halting instruction, `no` otherwise.

The *recurrent state problem* for two-counter machines is defined as follows:

Input: a two-counter machine **M**

Output: `yes` if there exists an infinite computation of **M** that visits the initial instruction $\mathbf{I}_0$ infinitely often, `no` otherwise.

Both of these two problems are not decidable. In addition, the halting problem is Σ_1^0-complete [60], and the recurrent state problem is Σ_1^1-complete [7].

Chapter 3

The expressive power of MTL and TPTL

In this chapter we introduce the Ehrenfeucht–Fraïssé games for MTL and TPTL, respectively. We show that TPTL is strictly more expressive than MTL over both infinite data words and finite data words. We also consider the expressive power of several fragments of TPTL and MTL by restriction of syntactic resources, e.g., the until rank, the set of constraint numbers (or interval borders) and the number of register variables.

3.1 The Ehrenfeucht–Fraïssé game for MTL

In this section we define the Ehrenfeucht–Fraïssé game for MTL, and prove two theorems about the relationship between EF-game and expressive power.

First we give the definition of the until rank of an MTL-formula.

Definition 1. The until rank of an MTL-formula φ, denoted by $\mathsf{Rank}(\varphi)$, is defined inductively on the structure of φ:

- If φ is $\top$ or $p \in \mathbf{P}$, then $\mathsf{Rank}(\varphi) = 0$.
- If φ is $\neg\varphi_1$, then $\mathsf{Rank}(\varphi) = \mathsf{Rank}(\varphi_1)$.
- If φ is $\varphi_1 \wedge \varphi_2$, then $\mathsf{Rank}(\varphi) = \max\{\mathsf{Rank}(\varphi_1), \mathsf{Rank}(\varphi_2)\}$.
- If φ is $\varphi_1 \mathsf{U}_I \varphi_2$, then $\mathsf{Rank}(\varphi) = \max\{\mathsf{Rank}(\varphi_1), \mathsf{Rank}(\varphi_2)\} + 1$.

In the following we define several fragments of MTL. Let $S \subseteq \mathbb{Z}$, and let $k \in \mathbb{N}$. Define

$$\begin{aligned}
&\mathsf{MTL}^S = \{\varphi \in \mathsf{MTL} \mid \text{all interval borders in } \varphi \text{ are in } S\},\\
&\mathsf{MTL}_k = \{\varphi \in \mathsf{MTL} \mid \mathsf{Rank}(\varphi) \leq k\},\\
&\mathsf{MTL}_k^S = \mathsf{MTL}^S \cap \mathsf{MTL}_k.
\end{aligned}$$

It is easily seen that $\mathsf{MTL} = \bigcup_{k\in\mathbb{N}} \bigcup_{S\subseteq\mathbb{Z}} \mathsf{MTL}_k^S$.

Example 4. In $\mathsf{MTL}^{\emptyset}$ we can only use the until modality $\mathsf{U}_{\mathbb{Z}}$ (U for short) that is augmented with the interval of all integers, i.e., if $\varphi_1 \mathsf{U}_I \varphi_2 \in \mathsf{MTL}^{\emptyset}$, then $I = \mathbb{Z}$.

Let $S = \{1,3\}$, and let φ be a formula in MTL^S. Then the until modality U in φ can be augmented with the intervals $\mathbb{Z}$, $[1,3],(1,3],[1,3),(1,3)$ or $(-\infty,i),(-\infty,i],[i,i],(i,+\infty),[i,+\infty)$, where i is 1 or 3.

If $S \subseteq \mathbb{Z}$ is a finite set, we can show that MTL_k^S is also finite up to equivalence for every $k \in \mathbb{N}$. This result is important for the proofs of Theorems 1 and 2.

Lemma 1. *For every finite $S \subseteq \mathbb{Z}$ and every $k \in \mathbb{N}$, there are only finitely many formulas in MTL_k^S up to equivalence.*

Proof. Fix a finite set $S \subseteq \mathbb{Z}$. We prove this lemma by induction on k.

For $k = 0$, no until modality will be used, the number of propositional formulas is finite up to equivalence since $\mathbf{P}$ is a finite set. Suppose that it holds for k, we will prove it for $k+1$. Because S is a finite set, the number of different until modalities U_I is finite, where the borders of I are in S. So there are finitely many formulas up to equivalence in the set

$$R = \{\varphi_1 \mathsf{U}_I \varphi_2 \mid \varphi_1, \varphi_2 \in \mathsf{MTL}_k^S\} \cup \mathsf{MTL}_k^S.$$

The formulas obtained by using Boolean connectives on the formulas in R are also finitely many up to equivalence. Hence MTL_{k+1}^S is a finite set up to equivalence. □

Definition 2. 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 $\varphi \in \mathsf{MTL}_k^S$, $(w_0,i_0) \models \varphi$ if and only if $(w_1,i_1) \models \varphi$.

We will write $w_0 \equiv_k^S w_1$ if $(w_0,0) \equiv_k^S (w_1,0)$ in the following.

Definition 3. Let $S \subseteq \mathbb{Z}$ be a finite set, and let $c_1,\ldots,c_n$ be a list of all numbers in S such that $c_i < c_{i+1}\,(1 \leq i < n)$. For any $a,b \in \mathbb{Z}$, we say that $a \overset{S}{\equiv} b$ if $a = b$, or both a and b belong to one of the intervals $(-\infty,c_1)$, $(c_n,+\infty)$, $(c_i,c_{i+1})\,(1 \leq i < n)$ (or belong to the interval $(-\infty,+\infty)$, if $S = \emptyset$).

It is easily seen that for every finite $S \subseteq \mathbb{Z}$ the binary relation $\overset{S}{\equiv}$ is an equivalence relation on $\mathbb{Z}$. The equivalence classes form a partition of $\mathbb{Z}$.

Example 5. Let $S = \{1,4,8\}$. The set of equivalence classes is

$$\{(-\infty,1),[1,1],(1,4),[4,4],(4,8),[8,8],(8,+\infty)\}.$$

We have $2 \overset{S}{\equiv} 3$, $5 \overset{S}{\equiv} 6$, $a_1 \overset{S}{\equiv} b_1\,(a_1, b_1 \leq 0)$ and $a_2 \overset{S}{\equiv} b_2\,(a_2, b_2 \geq 9)$.

Let w_0, w_1 be two data words. Recall that we use $P_{i,j}$ and $d_{i,j}\,(i \in \{0,1\}, j \in \mathbb{N})$ to denote the set of propositions and data value in the position j of data word w_i, respectively.

We define the Ehrenfeucht–Fraïssé game for MTL in the following. The EF-game is played by two players, called Spoiler and Duplicator, on two data words w_0 and w_1 with a finite set $S \subseteq \mathbb{Z}$. A game configuration is a pair of positions $(i_0, i_1) \in \mathbb{N} \times \mathbb{N}$, where i_0 is a position in w_0 and i_1 is a position in w_1. In each round of the game, where the current game configuration is (i_0, i_1):

(1) Spoiler chooses an index $l \in \{0,1\}$ and a position $j_l > i_l$ in data word w_l.

(2) Duplicator responds with a position $j_{1-l} > i_{1-l}$ in data word w_{1-l} such that

- if $j_l = i_l + 1$, then $j_{1-l} = i_{1-l} + 1$, and
- $(d_{l,j_l} - d_{l,i_l}) \overset{S}{\equiv} (d_{1-l,j_{1-l}} - d_{1-l,i_{1-l}})$.

Then Spoiler chooses between one of the following two options:

(a) The new configuration becomes (j_0, j_1).

(b) Spoiler chooses a position $i_{1-l} < j'_{1-l} < j_{1-l}$ in w_{1-l}, Duplicator responds with a position $i_l < j'_l < j_l$ in w_l, and the new configuration becomes (j'_0, j'_1).

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

The *winning condition for Duplicator* is defined inductively. We say that Duplicator wins the 0-round EF-game $\mathrm{MG}_0^S(w_0, i_0, w_1, i_1)$ if $P_{0,i_0} = P_{1,i_1}$. Duplicator wins the $(k+1)$-round EF-game $\mathrm{MG}_{k+1}^S(w_0, i_0, w_1, i_1)$ if she wins the 0-round EF-game $\mathrm{MG}_0^S(w_0, i_0, w_1, i_1)$, and either i_0 and i_1 are the last positions of w_0 and w_1, respectively (Spoiler has no position to choose), or for every choice of moves of Spoiler in the first round, Duplicator can respond correctly and wins the k-round EF-game $\mathrm{MG}_k^S(w_0, n_0, w_1, n_1)$, where (n_0, n_1) is the new configuration after the first round. If Duplicator cannot win the game, we say that Spoiler wins the game. We write $(w_0, i_0) \sim_k^S (w_1, i_1)$ if Duplicator wins the k-round EF-game $\mathrm{MG}_k^S(w_0, i_0, w_1, i_1)$. It follows easily that if $(w_0, i_0) \sim_k^S (w_1, i_1)$, then for all $m < k$, $(w_0, i_0) \sim_m^S (w_1, i_1)$. We will write $w_0 \sim_k^S w_1$ if $(w_0, 0) \sim_k^S (w_1, 0)$ in the following.

Theorem 1. *Let w_0, w_1 be two data words, and let i_0, i_1 be two positions in w_0, w_1, respectively. For every finite $S \subseteq \mathbb{Z}$ and every $k \in \mathbb{N}$, $(w_0, i_0) \equiv_k^S (w_1, i_1)$ if and only if $(w_0, i_0) \sim_k^S (w_1, i_1)$.*

Proof. Fix a finite set $S \subseteq \mathbb{Z}$. We prove this theorem by induction on k. It is clear for $k = 0$. Suppose that this theorem holds for k, we show that it also holds for $k+1$.

For the direction "$\Rightarrow$", we give a proof by contradiction. Suppose $(w_0, i_0) \equiv^S_{k+1} (w_1, i_1)$ holds and $(w_0, i_0) \sim^S_{k+1} (w_1, i_1)$ does not hold. Then Spoiler wins $\mathrm{MG}^S_{k+1}(w_0, i_0, w_1, i_1)$. Without loss of generality suppose that Spoiler chooses w_0 and a position $i'_0 > i_0$ in w_0. For each $i_0 < j \leq i'_0$, define

$$\varphi_j = \bigwedge \{\varphi \in \mathsf{MTL}^S_k \mid (w_0, j) \models \varphi\}.$$

Note that φ_j is well-defined since there are only finitely many MTL^S_k-formulas up to equivalence by Lemma 1. Define the MTL^S_{k+1}-formula

$$\varphi = (\bigvee_{i_0 < j < i'_0} \varphi_j) \mathsf{U}_I \varphi_{i'_0}$$ [1]

where $d = d_{0,i'_0} - d_{0,i_0}$ and

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

Clearly, $\mathsf{Rank}(\varphi) \leq k+1$ and $(w_0, i_0) \models \varphi$. We have $(w_1, i_1) \models \varphi$ since $(w_0, i_0) \equiv^S_{k+1} (w_1, i_1)$. Hence there exists $i'_1 > i_1$ such that $(w_1, i'_1) \models \varphi_{i'_0}$ and $d_{1,i'_1} - d_{1,i_1} \in I$, and for all $i_1 < i''_1 < i'_1$, $(w_1, i''_1) \models \bigvee_{i_0 < j < i'_0} \varphi_j$. So Duplicator can respond with the position i'_1 in w_1. Now if they start a new round from the configuration (i'_0, i'_1), we know by $(w_1, i'_1) \models \varphi_{i'_0}$ and the definition of $\varphi_{i'_0}$ that $(w_0, i'_0) \equiv^S_k (w_1, i'_1)$. By induction hypothesis, $(w_0, i'_0) \sim^S_k (w_1, i'_1)$, which means that Duplicator wins the remaining k rounds. On the other hand, if Spoiler chooses a position $i_1 < i''_1 < i'_1$ in w_1, we know by $(w_1, i''_1) \models \bigvee_{i_0 < j < i'_0} \varphi_j$ that there is a position $i_0 < i''_0 < i'_0$ such that $(w_1, i''_1) \models \varphi_{i''_0}$. Hence Duplicator can respond with the position i''_0 in w_0. If they start a new round from the configuration (i''_0, i''_1), we know by the definition of $\varphi_{i''_0}$ that $(w_0, i''_0) \equiv^S_k (w_1, i''_1)$, and thus, by induction hypothesis, that $(w_0, i''_0) \sim^S_k (w_1, i''_1)$, which means that Duplicator wins the remaining k rounds again. Because $(w_0, i_0) \equiv^S_{k+1} (w_1, i_1)$, we have $P_{0,i_0} = P_{1,i_1}$. Finally, we know that Duplicator wins the game $\mathrm{MG}^S_{k+1}(w_0, i_0, w_1, i_1)$, which contradicts the assumption.

For the "$\Leftarrow$" direction, suppose that $(w_0, i_0) \sim^S_{k+1} (w_1, i_1)$, i.e., Duplicator wins the game. We show that $(w_0, i_0) \equiv^S_{k+1} (w_1, i_1)$. Let $\varphi \in \mathsf{MTL}^S_{k+1}$. If $\mathsf{Rank}(\varphi) \leq k$, then $(w_0, i_0) \models \varphi$

[1] $\varphi = \bot \mathsf{U}_I \varphi_{i'_0}$ if $i'_0 = i_0 + 1$, i.e., there are no positions between position i_0 and position i'_0.

if and only if $(w_1, i_1) \models \varphi$, since $(w_0, i_0) \sim^S_{k+1} (w_1, i_1)$ implies $(w_0, i_0) \sim^S_k (w_1, i_1)$, and by induction hypothesis, $(w_0, i_0) \sim^S_k (w_1, i_1)$ if and only if $(w_0, i_0) \equiv^S_k (w_1, i_1)$. If $\mathsf{Rank}(\varphi) = k+1$, we prove for the case $\varphi = \varphi_1 \mathsf{U}_I \varphi_2$ where $\varphi_1, \varphi_2 \in \mathsf{MTL}^S_k$, that $(w_0, i_0) \models \varphi_1 \mathsf{U}_I \varphi_2$ if and only if $(w_1, i_1) \models \varphi_1 \mathsf{U}_I \varphi_2$. The proof for the other cases is easy. Without loss of generality, suppose $(w_0, i_0) \models \varphi_1 \mathsf{U}_I \varphi_2$, we show that $(w_1, i_1) \models \varphi_1 \mathsf{U}_I \varphi_2$. The other direction can be proved analogously.

We have $(w_0, i_0) \models \varphi_1 \mathsf{U}_I \varphi_2$ if and only if there exists $i'_0 > i_0$ such that $(w_0, i'_0) \models \varphi_2$ and $d_{0,i'_0} - d_{0,i_0} \in I$, and for all $i_0 < j < i'_0$, $(w_0, j) \models \varphi_1$. Assume Spoiler chooses the position i'_0 in w_0. Since Duplicator wins the game, she can respond with a position $i'_1 > i_1$ in w_1 such that $(d_{0,i'_0} - d_{0,i_0}) \overset{S}{\equiv} (d_{1,i'_1} - d_{1,i_1})$ and $(w_0, i'_0) \sim^S_k (w_1, i'_1)$. By induction hypothesis, $(w_0, i'_0) \equiv^S_k (w_1, i'_1)$. Thus, $(w_1, i'_1) \models \varphi_2$. On the other hand, if Spoiler chooses a position $i_1 < i''_1 < i'_1$ in w_1, we know by assumption that Duplicator can respond with a position $i_0 < i''_0 < i'_0$ such that $(w_0, i''_0) \sim^S_k (w_1, i''_1)$. 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 choose an arbitrary position between i_1 and i'_1, we can know that $(w_1, j) \models \varphi_1$ for every $i_1 < j < i'_1$. Adding $(d_{0,i'_0} - d_{0,i_0}) \overset{S}{\equiv} (d_{1,i'_1} - d_{1,i_1})$ and $(w_1, i'_1) \models \varphi_2$ we have $(w_1, i_1) \models \varphi_1 \mathsf{U}_I \varphi_2$. Finally, we have $(w_0, i_0) \equiv^S_{k+1} (w_1, i_1)$. □

Theorem 2. *Let* $\mathbf{C}$ *be a class of data words. The following are equivalent:*

(1) $\mathbf{C}$ *is not definable in* MTL.

(2) For every finite $S \subseteq \mathbb{Z}$ *and every* $k \in \mathbb{N}$*, there exist two data words* $w_0 \in \mathbf{C}$ *and* $w_1 \notin \mathbf{C}$ *such that* $w_0 \sim^S_k w_1$.

Proof. From (1) to (2), we give a proof by contradiction. Assume (1) holds and (2) does not hold. Then there exist a finite set $S \subseteq \mathbb{Z}$ and a number $k \in \mathbb{N}$ such that for every pair of data words w_0, w_1, if $w_0 \in \mathbf{C}$ and $w_1 \notin \mathbf{C}$, then $w_0 \not\sim^S_k w_1$. By Theorem 1, this implies $w_0 \not\equiv^S_k w_1$. So for every pair of data words w_0, w_1, if $w_0 \equiv^S_k w_1$, then $w_0 \in \mathbf{C}$ if and only if $w_1 \in \mathbf{C}$. For a data word w, we define

$$\varphi_w = \bigwedge \{\varphi \in \mathsf{MTL}^S_k \mid w \models \varphi\}.$$

We define the MTL^S_k-formula

$$\Phi = \bigvee_{w \in \mathbf{C}} \varphi_w.$$

Note that there are only finitely many formulas in MTL^S_k up to equivalence, so both φ_w and Φ are well-defined. We show that $\mathbf{C}$ is definable by the formula Φ, which contradicts the assumption. For an arbitrary data word w, if $w \in \mathbf{C}$, then $w \models \Phi$ by the definition of Φ. If $w \models \Phi$, there must exists some data word $w' \in \mathbf{C}$ such that $w \models \varphi_{w'}$. This implies that w and w' satisfy the same formulas in MTL^S_k, i.e., $w \equiv^S_k w'$. Then we have $w \in \mathbf{C}$ since $w' \in \mathbf{C}$.

From (2) to (1), suppose by contradiction that $\mathbf{C}$ is definable by the MTL_k^S-formula ψ for some finite set $S \subseteq \mathbb{Z}$ and $k \in \mathbb{N}$. We can know that for every pair of data words w_0, w_1, if $w_0 \in \mathbf{C}$ and $w_1 \notin \mathbf{C}$, then $w_0 \not\equiv_k^S w_1$, since $w_0 \models \psi$ and $w_1 \not\models \psi$. By Theorem 1, this implies $w_0 \not\sim_k^S w_1$, contrary to (2). □

3.2 Application of the EF-game for MTL

In this section, we present one of our main results: TPTL is strictly more expressive than MTL over both infinite data words and finite data words. Furthermore, we show that the problem which asks whether a TPTL-formula is definable in MTL is undecidable. We also prove one hierarchy theorem for the fragment of MTL that restrict the until rank and two hierarchy theorems for the fragment of MTL that restrict the set of interval borders.

3.2.1 Relative expressiveness of TPTL and MTL

We first prove two useful lemmas.

Lemma 2. *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 $i \geq 0$, $P_{0,i} = P_{1,i}$ and for every $j' > j \geq 0$, $(d_{0,j'} - d_{0,j}) \overset{S}{\equiv} (d_{1,j'} - d_{1,j})$, then for every $k \in \mathbb{N}$, $w_0 \sim_k^S w_1$.*

Proof. The proof is straightforward. In each round, Duplicator can always respond with the same position that Spoiler chooses in the other data word. It is easy to check that Duplicator wins the game $\mathrm{MG}_k^S(w_0, 0, w_1, 0)$ for every $k \in \mathbb{N}$. □

Corollary 1. *For every $n \in \mathbb{N}$, w and w_{+n} satisfy the same* MTL*-formulas.*

Lemma 3. *Let $S \subseteq \mathbb{Z}$ be a nonempty finite set, and let m be the maximal number in S. Let u_1, u_2 be two finite data words and $c \in \mathbb{N}$ such that*

$$\min(u_2) - \max(u_1) > m, \tag{3.1}$$

and

$$\min(u_2) + c - \max(u_2) > m. \tag{3.2}$$

Then for every $k \in \mathbb{N}$ and every data word w, if either w is empty or

$$\min(w) - \max(u_2) - kc > m, \tag{3.3}$$

then

$$u_1(u_2)^k_{+c}w \sim^S_k u_1(u_2)^{k+1}_{+c}w.$$

Proof. Without loss of generality we can assume $k \geq 1$ and w is not empty. Let $w_0 = u_1(u_2)^k_{+c}w$ and $w_1 = u_1(u_2)^{k+1}_{+c}w$. We show that Duplicator wins the game $\mathrm{MG}^S_k(w_0, 0, w_1, 0)$. We only need to consider the choices of Spoiler in the first round. There are four cases:

(1) Spoiler chooses a position in w from w_0 or w_1, Duplicator can respond with the same position in w from the other data word. Let w' be the suffix of w from that position. Suppose that they continue to play the remaining $(k-1)$ rounds on the new data word w'. Obviously, Duplicator wins the game $\mathrm{MG}^S_{k-1}(w', 0, w', 0)$.

(2) Spoiler chooses a position in the prefix u_1u_2 of w_0 or w_1, Duplicator can respond with the same position in u_1u_2 from the other data word. Let u'_1 be the suffix of u_1u_2 from that position. Suppose that they continue to play the remaining $(k-1)$ rounds on the new data words $w'_0 = u'_1(u'_2)^{k-1}_{+c}w$ and $w'_1 = u'_1(u'_2)^k_{+c}w$, where $u'_2 = (u_2)_{+c}$. It is easily seen that u'_1, u'_2 also satisfy the premise of the lemma. By induction on k, we can show that Duplicator wins the game $\mathrm{MG}^S_{k-1}(w'_0, 0, w'_1, 0)$.

(3) Spoiler chooses a position in the $i^{\text{th}}\,(2 \leq i \leq k)$ repetition of u_2 in w_0, Duplicator can respond with the same position in the $(i+1)^{\text{th}}$ repetition of u_2 in w_1. Let w'_0 be the suffix of w_0 from the position that Spoiler chooses in w_0, and let w'_1 be the suffix of w_1 from the position with which Duplicator responds in w_1. Suppose that they continue to play the remaining $(k-1)$ rounds on the new data words w'_0 and w'_1. By Lemma 2, Duplicator wins the game $\mathrm{MG}^S_{k-1}(w'_0, 0, w'_1, 0)$.

(4) Spoiler chooses a position in the $i^{\text{th}}\,(2 \leq i \leq k+1)$ repetition of u_2 in w_1, Duplicator can respond with the same position in the $(i-1)^{\text{th}}$ repetition of u_2 in w_0. Let w'_1 be the suffix of w_1 from the position that Spoiler chooses in w_1, and let w'_0 be the suffix of w_0 from the position with which Duplicator responds in w_0. Suppose they continue to play the remaining $(k-1)$-round game on the new data words w'_0 and w'_1. By Lemma 2, Duplicator wins the game $\mathrm{MG}^S_{k-1}(w'_0, 0, w'_1, 0)$.

In the first round, whatever Spoiler chooses, Duplicator can respond according to the above four cases. It is easily seen that the response of Duplicator satisfies the winning condition about the atomic propositions and the difference of data values by (3.1), (3.2) and (3.3). □

In [6], the authors showed that MTL and TPTL have the same expressive power over infinite monotonic data words. Let w be a data word, and let i be a position in w. It is easily seen that $(w, i) \models \mathsf{X}\top$ if and only if the position i is not the last position of w. So a data word

w is infinite if and only if $w \models \mathsf{X}\top \wedge \mathsf{GX}\top$. We consider the expressive power over the class of infinite data words and the class of finite data words separately. In the following we show that TPTL is strictly more expressive than MTL over both infinite and finite data words.

Theorem 3. TPTL^1 *is strictly more expressive than* MTL *over both infinite and finite data words.*

Proof. To show that $\mathsf{MTL} \preccurlyeq \mathsf{TPTL}^1$, for every MTL-formula ψ, we can inductively define an equivalent TPTL^1-formula ψ' by the following rules:

- If ψ is $\top$ or $p \in \mathbf{P}$, then $\psi' = \psi$.
- If ψ is $\neg\psi_1$, then $\psi' = \neg\psi_1'$.
- If ψ is $\psi_1 \wedge \psi_2$, then $\psi' = \psi_1' \wedge \psi_2'$.
- If ψ is $\psi_1 \mathsf{U}_I \psi_2$, then $\psi' = x.(\psi_1' \mathsf{U} (\psi_2' \wedge x \in I))$.

In the following we show that the TPTL^1-formula $x.\mathsf{XX}(x = 0)$ is not definable in MTL over both infinite and finite data words.

First we consider the non-definability over infinite data words. For every finite $S \subseteq \mathbb{Z}$, we define two infinite data words w_0 and w_1 such that $w_0 \not\models x.\mathsf{XX}(x = 0)$, $w_1 \models x.\mathsf{XX}(x = 0)$, and $w_0 \sim_k^S w_1$ for every $k \in \mathbb{N}$. Then, by Theorem 2, we can know that $x.\mathsf{XX}(x = 0)$ is not definable in MTL over infinite data words.

Let $S \subseteq \mathbb{Z}$ be a finite set, 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 $\pm r$ in data value cannot be detected by a formula in MTL^S, as all numbers in S are contained in $(-r, +r)$. Define two infinite pure data words $w_0 = (s)(s-2r)^\omega_{+r}$ and $w_1 = (s)(s-r)^\omega_{+r}$ (see Fig. 3.1). It

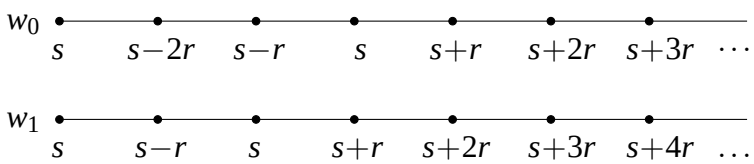

Fig. 3.1 The infinite data words w_0 and w_1

is easily seen that $w_0 \not\models x.\mathsf{XX}(x = 0)$ and $w_1 \models x.\mathsf{XX}(x = 0)$. We show that Duplicator wins the game $\mathrm{MG}_k^S(w_0, 0, w_1, 0)$ for every $k \in \mathbb{N}$. The proof for the case $k = 0$ is easy. Without loss

	Case 1	Case 2	Case 3	Case 4
Move 1	$(\underline{1},1)$ or $(1,\underline{1})$	$(\underline{2},1)$	$(\underline{i},i-1)$, $i \geq 3$	$(i+1,\underline{i})$, $i \geq 2$
Move 2	-	-	$(j+1,\underline{j})$, $1 \leq j \leq i-2$	$(\underline{1},1)$ or $(\underline{j},j-1)$ $2 \leq j \leq i$

Table 3.1 The winning strategy for Duplicator in the first round

of generality, we assume $k \geq 1$. We give the winning strategy for Duplicator in the first round. There are four cases (see Table 3.1 [2]):

(1) Spoiler chooses the position 1 in w_0 or w_1, Duplicator can respond with the position 1 in the other data word. We have $(d_{0,1} - d_{0,0}) \overset{S}{\equiv} (d_{1,1} - d_{1,0})$ since $(d_{0,1} - d_{0,0}) = -2r$ and $(d_{1,1} - d_{1,0}) = -r$.

(2) Spoiler chooses the position 2 in w_0, Duplicator can respond with the position 1 in w_1. We have $(d_{0,2} - d_{0,0}) \overset{S}{\equiv} (d_{1,1} - d_{1,0})$ since $(d_{0,2} - d_{0,0}) = (d_{1,1} - d_{1,0}) = -r$.

(3) Spoiler chooses the position $i\,(i \geq 3)$ in w_0, Duplicator can respond with the position $i-1$ in w_1. We have $(d_{0,i} - d_{0,0}) \overset{S}{\equiv} (d_{1,i-1} - d_{1,0})$ since $(d_{0,i} - d_{0,0}) = (d_{1,i-1} - d_{1,0}) = (i-3)r$. In the next move, if Spoiler chooses a position $1 \leq j \leq i-2$ in w_1, Duplicator can respond with the position $j+1$ in w_0.

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

After the first round, the new game configuration is either $(1,1)$ or $(i+1,i)\,(i \geq 1)$. If the new configuration is $(1,1)$, we have $(w_0[1:])_{+r} = w_1[1:]$. By Lemma 2, Duplicator wins the game $\mathrm{MG}^S_{k-1}(w_0,1,w_1,1)$. If the new configuration is $(i+1,i)\,(i \geq 1)$, we have $w_0[(i+1):] = w_1[i:]$. Obviously, Duplicator can win the game $\mathrm{MG}^S_{k-1}(w_0,i+1,w_1,i)$.

In the following we show that $x.\mathsf{XX}(x=0)$ is not definable in MTL over finite data words. It is similar to the case for infinite data words. Let $S \subseteq \mathbb{Z}$ be a finite set, let $r,s > 0$ be defined as

[2] Let (i_0,i_1) be a pair of positions, where i_0,i_1 are positions in w_0,w_1, respectively. We underline $i_j\,(j \in \{0,1\})$ to denote that Spoiler chooses the position i_j in w_j and Duplicator responds with the position i_{1-j} in w_{1-j}.

above. For every $k \in \mathbb{N}$, define two finite pure data words $w_0 = (s)(s-2r)^{k+3}_{+r}$ and $w_1 = (s)(s-r)^{k+2}_{+r}$. We show that Duplicator wins the game $\mathrm{MG}^S_k(w_0, 0, w_1, 0)$. Without loss of generality we assume $k \geq 1$. We can adapt the winning strategy for Duplicator in Table 3.1. After the first round, if the new configuration is $(1,1)$, we have $w_0[1:] = (s-2r)^{k+3}_{+r}$ and $w_1[1:] = (s-r)^{k+2}_{+r}$. By Lemma 3, $(s-2r)^{k+3}_{+r} \sim^S_{k-1} (s-2r)^{k+2}_{+r}$, and by Lemma 2, $(s-2r)^{k+2}_{+r} \sim^S_{k-1} (s-r)^{k+2}_{+r}$. This implies $(s-2r)^{k+3}_{+r} \sim^S_{k-1} (s-r)^{k+2}_{+r}$. So Duplicator wins the game $\mathrm{MG}^S_{k-1}(w_0, 1, w_1, 1)$. If the new configuration is $(i+1, i)$ $(i \geq 1)$, we have $w_0[(i+1):] = w_1[i:]$. Obviously, Duplicator can win the game $\mathrm{MG}^S_{k-1}(w_0, i+1, w_1, i)$. □

Remark 2. In [18], the authors showed that the TPTL-formula $x.\mathsf{F}(b \wedge \mathsf{F}(c \wedge x \leq 2))$ is not definable in MTL over timed words. A timed word is an infinite sequence $(P_0,t_0)(P_1,t_1)\ldots$, where $(P_i,t_i) \in (2^{\mathbf{P}} \times \mathbb{R}^+)$ $(i \in \mathbb{N})$, such that

- $t_0 = 0$,
- $\forall i \in \mathbb{N}, t_{i+1} \geq t_i$,
- $\forall s \in \mathbb{R}^+, \exists i \in \mathbb{N}$ such that $t_i > s$.

We can show that the formula $x.\mathsf{F}(b \wedge \mathsf{F}(c \wedge x \leq 2))$ is also not definable in MTL over infinite data words.

For every finite set $S \subseteq \mathbb{Z}$, let $r > 0$ be such that all numbers in S are contained in $(-r, +r)$ and $s > 3r$. Define two infinite data words (see Fig. 3.2)

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

where b, c are atomic propositions. It is easily seen that $w_0 \models x.\mathsf{F}(b \wedge \mathsf{F}(c \wedge x \leq 2))$ and $w_1 \not\models x.\mathsf{F}(b \wedge \mathsf{F}(c \wedge x \leq 2))$. We leave it to the reader to verify that Duplicator can win the game $\mathrm{MG}^S_k(w_0, 0, w_1, 0)$ for every $k \in \mathbb{N}$.

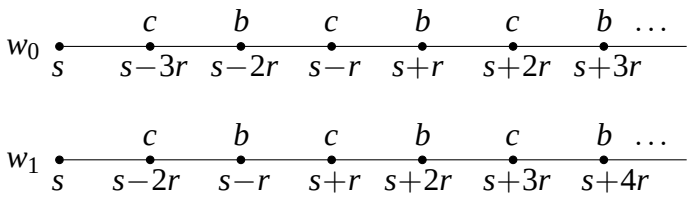

Fig. 3.2 The data words w_0 and w_1

Corollary 2. TPTL *is strictly more expressive than* MTL *over both infinite and finite data words.*

3.2.2 The MTL definability decision problem

In the last subsection, we show that TPTL is strictly more expressive than MTL. It is natural to ask: Given a TPTL-formula φ, is φ definable in MTL? In the following we show that this problem is undecidable over both infinite and finite data words using the EF-game method.

Theorem 4. *The problem that whether a* TPTL*-formula is definable in* MTL *is undecidable over both infinite and finite data words.*

Proof. First we show that whether a TPTL-formula is definable in MTL is undecidable over infinite data words. We reduce the recurrent state problem of two-counter machines which is undecidable to the MTL definability decision problem in the following way: For every two-counter machine $\mathbf{M}$, we construct a TPTL-formula $\psi_{\mathbf{M}}$ such that $\psi_{\mathbf{M}}$ is definable in MTL if and only if $\mathbf{M}$ is a negative instance of the recurrent state problem.

For every two-counter machine $\mathbf{M}$ there is a TPTL-formula φ_{infin} such that $\mathbf{M}$ is a positive instance of the recurrent state problem if and only if there is an infinite data word w such that $w \models \varphi_{\mathsf{infin}}$[3]. Define

$$\psi_{\mathbf{M}} = x.\mathsf{XX}(x = 0) \wedge \mathsf{F}\,\varphi_{\mathsf{infin}}\,.$$

If $\mathbf{M}$ is a negative instance, then φ_{infin} cannot be satisfied by any infinite data word, hence $\psi_{\mathbf{M}}$ is equivalent to the MTL-formula $\bot$ over infinite data words. Otherwise, if $\mathbf{M}$ is a positive instance, we show that for every finite $S \subseteq \mathbb{Z}$ and $k \in \mathbb{N}$, there are two infinite data words w_0 and w_1 such that $w_0 \not\models \psi_{\mathbf{M}}$, $w_1 \models \psi_{\mathbf{M}}$ and $w_0 \sim_k^S w_1$. Then by Theorem 2, we can know that $\psi_{\mathbf{M}}$ is not definable in MTL.

Suppose that $\mathbf{M}$ is a positive instance and φ_{infin} is satisfied by the infinite data word w. Let $S \subseteq \mathbb{Z}$ be a finite set, and let $r, s > 0$ be such that all numbers in S are contained in $(-r, +r)$ and $s > 2r$. For every $k \in \mathbb{N}$, we define two infinite data words $w_0 = (s)(s-2r)_{+r}^{k+3} w_{+m}$ and $w_1 = (s)(s-r)_{+r}^{k+2} w_{+m}$, where $m = s + (k+1)r$. By Lemma 2, we can know that $w_{+m} \models \varphi_{\mathsf{infin}}$. It is easily seen that $w_0 \not\models \psi_{\mathbf{M}}$ and $w_1 \models \psi_{\mathbf{M}}$. We shall show that $w_0 \sim_k^S w_1$. This can easily be shown by using the winning strategy for Duplicator in the proof of Theorem 3.

To prove that whether a TPTL-formula is definable in MTL is undecidable over finite data words, we use a reduction from the halting problem of two-counter machines. The proof is similar to the infinite case. For every two-counter machine $\mathbf{M}$ there is a TPTL-formula φ_{fin}

[3] In [7], the authors constructed a TPTL-formula that can capture the computation of a two-counter machine. We construct an equivalent MTL-formula. For more details about the formula we refer the reader to Section 4.1 in Chapter 4.

such that $\mathbf{M}$ is a positive instance of halting problem if and only if there is a finite data word w' such that $w' \models \varphi_{\mathsf{fin}}$ (see Chapter 4). For every two-counter machine $\mathbf{M}$, we can define the TPTL-formula $\psi'_{\mathbf{M}} = x.\mathsf{XX}(x=0) \wedge \mathsf{F}\,\varphi_{\mathsf{fin}}$ such that $\psi'_{\mathbf{M}}$ is definable in MTL if and only if $\mathbf{M}$ is a negative instance of the halting problem.

Suppose that $\mathbf{M}$ is a positive instance of the halting problem and φ_{fin} is satisfied by the finite data word w'. Let $S \subseteq \mathbb{Z}$ be a finite set, and let $r, s > 0$ be such that all numbers in S are contained in $(-r,+r)$ and $s > 2r$. For every $k \in \mathbb{N}$, we define two finite data words $w_0 = (s)(s-2r)^{k+3}_{+r} w'_{+m}$ and $w_1 = (s)(s-r)^{k+2}_{+r} w'_{+m}$, where $m = s+(k+1)r$. It is a simple matter to check that $w_0 \not\models \psi'_{\mathbf{M}}$, $w_1 \models \psi'_{\mathbf{M}}$ and $w_0 \sim^S_k w_1$. □

3.2.3 Effects on the expressiveness by restriction of syntactic resources

In this subsection, we consider the expressive power of several fragments of MTL by restriction of the until rank or the set of interval borders. In [36], the authors showed that the until rank hierarchy is strict for LTL. In a similar way, we can show that the until rank hierarchy for MTL is also strict over both infinite and finite data words.

Proposition 1. *For every $k \in \mathbb{N}$, MTL_{k+1} is strictly more expressive than MTL_k over both infinite and finite data words.*

Proof. Let $\varphi_1 = (p \wedge \mathsf{X}\,p)$, where p is an atomic proposition. For every $k \geq 1$, define $\varphi_{k+1} = (p \wedge \mathsf{X}\,\varphi_k)$. Note that for every $k \geq 1$, $\varphi_k \in \mathsf{MTL}_k$. We shall show that φ_k is not definable in MTL_{k-1} over both infinite and finite data words.

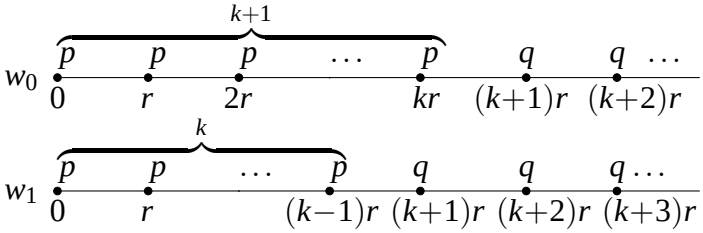

Fig. 3.3 The data words w_0 and w_1

Since $\mathsf{MTL}_{k-1} = \bigcup_{\text{finite } S \subseteq \mathbb{Z}} \mathsf{MTL}^S_{k-1}$, it will thus be sufficient to prove that φ_k is not definable in MTL^S_{k-1} for every finite $S \subseteq \mathbb{Z}$. Let $S \subseteq \mathbb{Z}$ be a finite set, and let $r > 0$ be such that all numbers in S are less than r. First we prove φ_k is not definable in MTL^S_{k-1} over infinite data words. Define two infinite data words $w_0 = (p,0)^{k+1}_{+r}(q,(k+1)r)^{\omega}_{+r}$ and $w_1 = (p,0)^{k}_{+r}(q,(k+1)r)^{\omega}_{+r}$, where p, q are propositions (see Fig. 3.3). We see that $w_0 \models \varphi_k$

and $w_1 \not\models \varphi_k$. By Lemma 3, we have $w_0 \sim^S_{k-1} w_1$, which implies that φ_k is not definable in MTL^S_{k-1}.

To prove φ_k is not definable in MTL^S_{k-1} over finite data words, we define two finite data words $w_0 = (p,0)^{k+1}_{+r}$ and $w_1 = (p,0)^k_{+r}$. Again, by Lemma 3, we have $w_0 \sim^S_{k-1} w_1$. □

Similar to the MTL definability decision problem, we can show that the problem that whether an MTL_{k+1}-formula is definable in MTL_k is also undecidable over both infinite and finite data words.

Proposition 2. *For every $k \geq 5$ (respectively, $k \geq 4$), the problem that whether an MTL_{k+1}-formula is definable in MTL_k is undecidable over infinite data words (respectively, finite data words).*

Proof. This proof is adapted from the proof of Theorem 4. For every two-counter machine $\mathbf{M}$ there is an MTL-formula φ_{infin} (respectively, φ_{fin}) such that $\mathbf{M}$ is a positive instance of the recurrent state problem (respectively, the halting problem) if and only if there is an infinite data word w such that $w \models \varphi_{\mathsf{infin}}$ (respectively, a finite data word w' such that $w' \models \varphi_{\mathsf{fin}}$).[4] We see that $\mathsf{Rank}(\varphi_{\mathsf{infin}}) = 5$ and $\mathsf{Rank}(\varphi_{\mathsf{fin}}) = 4$. Hence, $\mathsf{Rank}(\mathsf{F}\,\varphi_{\mathsf{infin}}) = 6$ and $\mathsf{Rank}(\mathsf{F}\,\varphi_{\mathsf{fin}}) = 5$.

Let $k \geq 5$. First we show that whether an MTL_{k+1}-formula is definable in MTL_k is undecidable over infinite data words. Define

$$\psi_{\mathbf{M}} = \varphi_{k+1} \wedge \mathsf{F}\,\varphi_{\mathsf{infin}},$$

where φ_{k+1} is the formula defined in the proof of Proposition 1, and we assume the proposition p does not occur in φ_{infin}. We have $\mathsf{Rank}(\psi_{\mathbf{M}}) = k+1$. If $\mathbf{M}$ is a negative instance, then $\psi_{\mathbf{M}}$ cannot be satisfied by any infinite data word, hence $\psi_{\mathbf{M}}$ is equivalent to the formula $\bot$ over infinite data words. Otherwise, if $\mathbf{M}$ is a positive instance, we show that for every finite $S \subseteq \mathbb{Z}$, there are two infinite data words w_0 and w_1 such that $w_0 \models \psi_{\mathbf{M}}$, $w_1 \not\models \psi_{\mathbf{M}}$ and $w_0 \sim^S_k w_1$. Then we can know that $\psi_{\mathbf{M}}$ is not definable in MTL_k. Let $S \subseteq \mathbb{Z}$ be a finite set, and let $r > 0$ be such that all numbers in S are less than r. Define $w_0 = (p,0)^{k+1}_{+r} w_{+(k+1)r}$ and $w_1 = (p,0)^k_{+r} w_{+(k+1)r}$. Similar to the proof of Proposition 1, we can show that $w_0 \sim^S_k w_1$.

To prove that whether an MTL_{k+1}-formula is definable in MTL_k is undecidable over finite data words, where $k \geq 4$, we can define

$$\psi'_{\mathbf{M}} = \varphi_{k+1} \wedge \mathsf{F}\,\varphi_{\mathsf{fin}}.$$

[4] For the details of φ_{infin} and φ_{fin} we refer the reader to the proof of Theorem 10 in Chapter 4.

For every finite $S \subseteq \mathbb{Z}$, let $r > 0$ be such that all numbers in S are less than r. Set $w_0 = (p,0)^{k+1}_{+r} w'_{+(k+1)r}$ and $w_1 = (p,0)^{k}_{+r} w'_{+(k+1)r}$. It is a simple matter to check that $w_0 \models \psi_{\mathbf{M}}$, $w_1 \not\models \psi_{\mathbf{M}}$ and $w_0 \sim^S_k w_1$. □

Corollary 3. *For every $k \in \mathbb{N}$, the problem that whether an* MTL*-formula is definable in* MTL_k *is undecidable over both infinite and finite data words.*

Let $S_1 \subseteq \mathbb{Z}$ and let S_2 be a proper subset of S_1. Intuitively, MTL^{S_1} is more expressive than MTL^{S_2}, e.g., $\mathsf{MTL}^{\{0\}} \preccurlyeq \mathsf{MTL}^{\{0,2\}}$. But this does not hold in general. For example, let $S_1 = \{0,1,2\}$ and $S_2 = \{0,2\}$. MTL^{S_1} and MTL^{S_2} have the same expressive power, since we can replace the interval that uses number 1 as the border by an equivalent interval that uses number 0 or 2 as the border, e.g., the MTL^{S_1}-formula $p \mathsf{U}_{[1,1]} q$ is equivalent to the MTL^{S_2}-formula $p \mathsf{U}_{(0,2)} q$, where p, q are atomic propositions. In the following we study the effects on the expressive power of MTL by restriction of the set of interval borders.

Lemma 4. *Let $S \subseteq \mathbb{Z}$, and let $n \in \mathbb{Z}$. If $\{n-1, n\} \cap S = \emptyset$ or $\{n, n+1\} \cap S = \emptyset$, then the formula $\mathsf{X}_{=n} \top$ is not definable in* MTL^S *over both infinite and finite data words.*

Proof. Since $\mathsf{MTL}^S = \bigcup_{\text{finite } S' \subseteq S} \bigcup_{k \in \mathbb{N}} \mathsf{MTL}^{S'}_k$, it is enough to show that $\mathsf{X}_{=n} \top$ is not definable in $\mathsf{MTL}^{S'}_k$ for every finite $S' \subseteq S$ and every $k \in \mathbb{N}$.

We first show that $\mathsf{X}_{=n} \top$ is not definable in MTL^S over infinite data words. Let $S' \subseteq S$ be a finite set. If $\{n, n+1\} \cap S = \emptyset$, then let $s, r > 0$ be such that $s + n > 0$ and all numbers in S' are less than $n + r$. We define two pure infinite data words $w_0 = (s)(s+n)^{\omega}_{+r}$ and $w_1 = (s)(s+n+1)^{\omega}_{+r}$ (see Fig. 3.4). It is easily seen that $w_0 \models \mathsf{X}_{=n} \top$ and $w_1 \not\models \mathsf{X}_{=n} \top$.

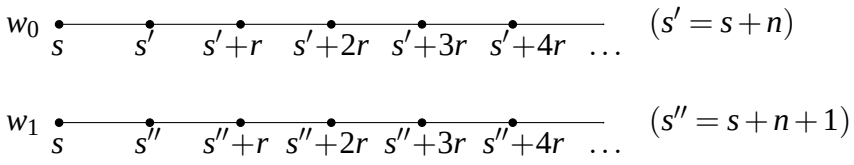

Fig. 3.4 The data words w_0 and w_1

For every $i' > i \geq 0$, we have

$$d_{0,i'} - d_{0,i} = \begin{cases} n & \text{if } i' = 1 \text{ and } i = 0, \\ n + (i'-1)r & \text{if } i' > 1 \text{ and } i = 0, \\ (i'-i)r & \text{if } i' > i \geq 1, \end{cases}$$

and

$$d_{1,i'} - d_{1,i} = \begin{cases} n+1 & \text{if } i' = 1 \text{ and } i = 0, \\ n+(i'-1)r+1 & \text{if } i' > 1 \text{ and } i = 0, \\ (i'-i)r & \text{if } i' > i \geq 1. \end{cases}$$

Since $\{n, n+1\} \cap S = \emptyset$ and all numbers in S' are less than $n+r$, we have for every $i' > i \geq 0$, $(d_{0,i'} - d_{0,i}) \overset{S'}{\equiv} (d_{1,i'} - d_{1,i})$. Then, by Lemma 2, we have $w_0 \sim_k^{S'} w_1$ for every $k \in \mathbb{N}$. This implies that $\mathsf{X}_{=n} \top$ is not definable in $\mathsf{MTL}^{S'}$.

Similarly, if $\{n-1, n\} \cap S = \emptyset$, then let $s, r > 0$ be such that $s+n-1 > 0$ and all numbers in S' are less than $n+r-1$. We define two pure infinite data words $w_0 = (s)(s+n)_{+r}^{\omega}$ and $w_1 = (s)(s+n-1)_{+r}^{\omega}$. We have $w_0 \models \mathsf{X}_{=n} \top$ and $w_1 \not\models \mathsf{X}_{=n} \top$, and for every $i' > i \geq 0$, $(d_{0,i'} - d_{0,i}) \overset{S'}{\equiv} (d_{1,i'} - d_{1,i})$. By Lemma 2 again, we have $w_0 \sim_k^{S'} w_1$ for every $k \in \mathbb{N}$.

To prove $\mathsf{X}_{=n} \top$ is not definable in MTL^S over finite data words, let $S' \subseteq S$ be a finite set and let $k \in \mathbb{N}$. We can set $w_0 = (s)(s+n)_{+r}^{k+1}$ and $w_1 = (s)(s+n+1)_{+r}^{k+1}$, if $\{n, n+1\} \cap S = \emptyset$, or $w_0 = (s)(s+n)_{+r}^{k+1}$ and $w_1 = (s)(s+n-1)_{+r}^{k+1}$, if $\{n-1, n\} \cap S = \emptyset$, where s, r are defined as in the infinite case. By Lemma 2, we have $w_0 \sim_k^{S'} w_1$. □

The following three propositions hold over both infinite data words and finite data words.

Proposition 3. *Let $S_1 \subseteq \mathbb{Z}$ and $S_2 \subseteq \mathbb{Z}$. $\mathsf{MTL}^{S_1} \preccurlyeq \mathsf{MTL}^{S_2}$ if and only if for every $n \in S_1$, either $n \in S_2$ or $\{n-1, n+1\} \subseteq S_2$.*

Proof. For the direction "⇒", suppose that there exists a number $m \in S_1$ such that $\{m-1, m\} \cap S_2 = \emptyset$ or $\{m, m+1\} \cap S_2 = \emptyset$. Then by Lemma 4, the MTL^{S_1}-formula $\mathsf{X}_{=m} \top$ is not definable in MTL^{S_2}, contrary to $\mathsf{MTL}^{S_1} \preccurlyeq \mathsf{MTL}^{S_2}$.

For the direction "⇐", let φ be an MTL^{S_1}-formula. We can construct an equivalent MTL^{S_2}-formula by replacing every constraint interval in φ that uses a number m as the border, where $m \in S_1$ and $m \notin S_2$, with an equivalent interval that uses $m-1$ or $m+1$ as the border. □

For every $n \in \mathbb{Z}$, let $\mathsf{MTL}^{\leq n} = \mathsf{MTL}^{\{m \in \mathbb{Z} \mid m \leq n\}}$. The expressive power relation $\preccurlyeq$ defines a linear order on the set $\{\mathsf{MTL}^{\leq n} \mid n \in \mathbb{Z}\}$ such that if $n_1 \leq n_2$, then $\mathsf{MTL}^{\leq n_1} \preccurlyeq \mathsf{MTL}^{\leq n_2}$. This gives the following constraint hierarchy for MTL.

Proposition 4. (Linear Constraint Hierarchy of MTL)
For any $n_1, n_2 \in \mathbb{Z}$, if $n_1 < n_2$, then $\mathsf{MTL}^{\leq n_1} \prec \mathsf{MTL}^{\leq n_2}$.

Proof. Obviously, $\mathsf{MTL}^{\leq n_1} \preccurlyeq \mathsf{MTL}^{\leq n_2}$. By Lemma 4, the $\mathsf{MTL}^{\leq n_2}$-formula $\mathsf{X}_{=n_2} \top$ is not definable in $\mathsf{MTL}^{\leq n_1}$. Hence, we have $\mathsf{MTL}^{\leq n_1} \prec \mathsf{MTL}^{\leq n_2}$. □

Let $R \subseteq \mathbb{Z}$ be a set such that for every $n \in \mathbb{Z}$, $n \in R$ if and only if $n+1 \notin R$, e.g., the set of all even numbers. By Proposition 3, we have $\mathsf{MTL}^R \equiv \mathsf{MTL}$. For any $S_1 \subseteq R$ and $S_2 \subseteq R$, if $S_1 \subsetneq S_2$, by Lemma 4, we have $\mathsf{MTL}^{S_1} \prec \mathsf{MTL}^{S_2}$. The expressive power relation $\preccurlyeq$ defines a partial order on the set $\{\mathsf{MTL}^S \mid S \subseteq R\}$. In the following we give another constraint hierarchy for MTL.

Proposition 5. (Lattice Constraint Hierarchy of MTL)
Let $R \subseteq \mathbb{Z}$ be a set such that for every $n \in \mathbb{Z}$, $n \in R$ if and only if $n+1 \notin R$. Then $\langle \{\mathsf{MTL}^S \mid S \subseteq R\}, \preccurlyeq \rangle$ constitutes a complete lattice in which

(i) the greatest element is MTL^R,

(ii) the least element is $\mathsf{MTL}^{\emptyset}$,

and for every $Q \subseteq \mathscr{P}(R)$, where $\mathscr{P}(R)$ is the power set of R,

(iii) $\bigwedge_{S \in Q} \mathsf{MTL}^S = \mathsf{MTL}^{\bigcap_{S \in Q} S}$,

(iv) $\bigvee_{S \in Q} \mathsf{MTL}^S = \mathsf{MTL}^{\bigcup_{S \in Q} S}$.

Proof. The proof for (i) and (ii) is easy.

For (iii), it is easily seen that $\mathsf{MTL}^{\bigcap_{S \in Q} S}$ is a lower bound of $\{\mathsf{MTL}^S \mid S \in Q\}$. For every $S' \subseteq R$, if $\mathsf{MTL}^{S'}$ is a lower bound of $\{\mathsf{MTL}^S \mid S \in Q\}$, then we must have $S' \subseteq \bigcap_{S \in Q} S$, i.e., $\mathsf{MTL}^{S'} \preccurlyeq \mathsf{MTL}^{\bigcap_{S \in Q} S}$. Otherwise, there exists $m \in S'$ such that $m \notin \bigcap_{S \in Q} S$. We can know that $m \notin S_1$ for some $S_1 \in Q$. By Lemma 4, the $\mathsf{MTL}^{S'}$-formula $\mathsf{X}_{=m} \top$ is not definable in MTL^{S_1}, contrary to that $\mathsf{MTL}^{S'}$ is a lower bound.

For (iv), it is easily seen that $\mathsf{MTL}^{\bigcup_{S \in Q} S}$ is an upper bound of $\{\mathsf{MTL}^S \mid S \in Q\}$. For every $S' \subseteq R$, if $\mathsf{MTL}^{S'}$ is an upper bound of $\{\mathsf{MTL}^S \mid S \in Q\}$, then we must have $\bigcup_{S \in Q} S \subseteq S'$, i.e., $\mathsf{MTL}^{\bigcup_{S \in Q} S} \preccurlyeq \mathsf{MTL}^{S'}$. Otherwise, there exists $m \in \bigcup_{S \in Q} S$ such that $m \notin S'$. We can know that $m \in S_1$ for some $S_1 \in Q$. By Lemma 4, the MTL^{S_1}-formula $\mathsf{X}_{=m} \top$ is not definable in $\mathsf{MTL}^{S'}$, contrary to that $\mathsf{MTL}^{S'}$ is an upper bound. □

3.3 MTL with non-strict semantics

In the definition of MTL, we use the strict semantics for the until modality. There is another definition for MTL that uses the non-strict semantics (see [6], and we denote this logic by weakMTL in this section). In weakMTL, the next modality X is given explicitly in the syntax, since it is not definable by the until modality interpreted by the non-strict semantics.

The interpretations for the next modality and the until modality in weakMTL are as follows (we use a dot over the modality operator to denote that the modality operator is interpreted by the non-strict semantics):

- $(w,i) \models \dot{\mathsf{X}}_I \varphi$ if and only if $i+1 < |w|$, $(w,i+1) \models \varphi$ and $d_{i+1} - d_i \in I$.
- $(w,i) \models \varphi_1 \dot{\mathsf{U}}_I \varphi_2$ if and only if there exists a position j with $i \leq j < |w|$ such that $(w,j) \models \varphi_2$, $d_j - d_i \in I$, and for all positions t with $i \leq t < j$, $(w,t) \models \varphi_1$.

It is easy to check that every weakMTL-formula is equivalent to an MTL-formula. Let φ be a weakMTL-formula. The equivalent MTL-formula φ' can be defined inductively by the following rules:

- If φ is $\top$ or $p \in \mathbf{P}$, then $\varphi' = \varphi$.
- If φ is $\neg\varphi_1$, then $\varphi' = \neg\varphi_1'$.
- If φ is $\varphi_1 \wedge \varphi_2$, then $\varphi' = \varphi_1' \wedge \varphi_2'$.
- If φ is $\dot{\mathsf{X}}_I \varphi_1$, then $\varphi' = \mathsf{X}_I \varphi_1'$.
- If φ is $\varphi_1 \,\dot{\mathsf{U}}_I\, \varphi_2$, then

$$\varphi' = \begin{cases} \varphi_1' \wedge (\varphi_1' \,\mathsf{U}_I\, \varphi_2') & \text{if } 0 \notin I, \\ \varphi_2' \vee (\varphi_1' \wedge (\varphi_1' \,\mathsf{U}_I\, \varphi_2')) & \text{otherwise.} \end{cases}$$

In [6], the authors showed that weakMTL and TPTL (with the non-strict semantics) have the same expressive power over infinite monotonic data words. Since every MTL-formula is equivalent to a TPTL-formula (with the strict semantics and over all data words), and both of the strict and the non-strict semantics are equivalent for TPTL. We can conclude that weakMTL and MTL are equivalent over infinite monotonic data words. But over non-monotonic data words, we can show that MTL is strictly more expressive than weakMTL. To prove this, we first introduce the Ehrenfeucht–Fraïssé game for weakMTL.

The EF-game for weakMTL is similar to the EF-game for MTL. Let $S \subseteq \mathbb{Z}$ be a finite set, and let w_0, w_1 be two data words. In each round of the game, suppose the current game configuration is (i_0, i_1), where i_0 and i_1 are positions in w_0 and w_1, respectively:

(1) Spoiler chooses an index $l \in \{0,1\}$ and a position $j_l \geq i_l$ in data word w_l.

(2) Duplicator responds with a position $j_{1-l} \geq i_{1-l}$ in data word w_{1-l} such that

- if $j_l = i_l + 1$, then $j_{1-l} = i_{1-l} + 1$, and if $j_l = i_l$, then $j_{1-l} = i_{1-l}$,
- $(d_{l,j_l} - d_{l,i_l}) \overset{S}{\equiv} (d_{1-l,j_{1-l}} - d_{1-l,i_{1-l}})$.

Then Spoiler chooses between one of the following two options:

(a) The new configuration becomes (j_0, j_1).

(b) Spoiler chooses a position $i_{1-l} \leq j'_{1-l} < j_{1-l}$ in w_{1-l}, Duplicator responds with a position $i_l \leq j'_l < j_l$ in w_l, and the new configuration becomes (j'_0, j'_1).

The difference between the EF-game for weakMTL and the EF-game for MTL is: In the EF-game for weakMTL, Spoiler can choose the current position in w_0 or w_1, but in the EF-game for MTL, Spoiler can only choose a position after the current position in w_0 or w_1.

We use $\mathrm{wMG}_k^S(w_0, i_0, w_1, i_1)$ to denote the k-round EF-game for weakMTL starting from the position i_0 in w_0 and the position i_1 in w_1 with a finite set $S \subseteq \mathbb{Z}$. We say that Duplicator wins the 0-round EF-game $\mathrm{wMG}_0^S(w_0, i_0, w_1, i_1)$ if $P_{0,i_0} = P_{1,i_1}$. Duplicator wins the $(k+1)$-round EF-game $\mathrm{wMG}_{k+1}^S(w_0, i_0, w_1, i_1)$ if she wins the 0-round EF-game $\mathrm{wMG}_0^S(w_0, i_0, w_1, i_1)$, and for every choice of moves of Spoiler in the first round, Duplicator can respond correctly and wins the k-round EF-game $\mathrm{wMG}_k^S(w_0, n_0, w_1, n_1)$, where (n_0, n_1) is the new configuration after the first round.

Let $S \subseteq \mathbb{Z}$ and $k \in \mathbb{N}$. The fragments $\mathsf{weakMTL}^S$, $\mathsf{weakMTL}_k$ and $\mathsf{weakMTL}_k^S$ are defined in a similar way to that of MTL. We abuse the notations "$(w_0, i_0) \equiv_k^S (w_1, i_1)$" and "$(w_0, i_0) \sim_k^S (w_1, i_1)$" for weakMTL, which are defined in the expected way. It is a simpler matter to check that Theorems 1 and 2, Lemmas 1, 2 and 3 also hold for weakMTL.

In the following we show that the MTL-formula $\mathsf{F}_{=0}\top$ is not definable in weakMTL over both infinite and finite data words. Note that a data word w satisfies the formula $\mathsf{F}_{=0}\top$ if and only if there is a position $j > 0$ such that $d_j = d_0$, where d_j and d_0 are the data values in the positions j and 0, respectively.

Theorem 5. MTL *is strictly more expressive than* weakMTL *over both infinite and finite data words.*

Proof. We prove the theorem by showing that the MTL-formula $\mathsf{F}_{=0}\top$ is not definable in weakMTL over both infinite and finite data words. The proof is similar to the proof of Theorem 3. For every finite $S \subseteq \mathbb{Z}$ and every $k \in \mathbb{N}$, we construct two data words w_0, w_1 such that $w_0 \models \mathsf{F}_{=0}\top$, $w_1 \not\models \mathsf{F}_{=0}\top$ and $w_0 \sim_k^S w_1$. Then we can know that $\mathsf{F}_{=0}\top$ is not definable in $\mathsf{weakMTL}_k^S$

First we consider the non-definability over infinite data words. 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 let $s > r$. Define two infinite pure data words $w_0 = (s)(s+r, s)^\omega_{+2r}$ and $w_1 = (s)(s+2r, s+r)^\omega_{+2r}$ (see Fig. 3.5).

In data word w_0, the data value s in position 2 equals to the data value in position 0, and in data word w_1, all data values after position 0 are greater than s. It is easily seen that $w_0 \models \mathsf{F}_{=0}\top$ and $w_1 \not\models \mathsf{F}_{=0}\top$. We show that $w_0 \sim_k^S w_1$ for every $k \in \mathbb{N}$ in the following. We

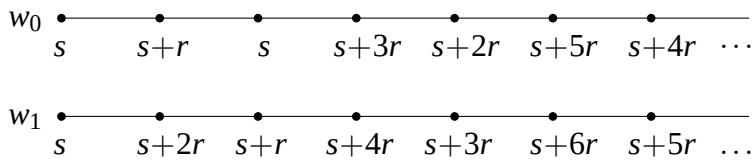

Fig. 3.5 The infinite data words w_0 and w_1

prove by induction on k. It is easy for the case $k = 0$. Suppose $k \geq 1$, we give the winning strategy for Duplicator in the first round. There are five cases (see Table 3.2 [5]):

(1) Spoiler chooses the position 0 in $w_j\,(j \in \{0,1\})$, Duplicator can respond with the position 0 in w_{1-j}.

(2) Spoiler chooses the position 1 in $w_j\,(j \in \{0,1\})$, Duplicator can respond with the position 1 in w_{1-j}. We have $(d_{0,1} - d_{0,0}) \overset{S}{\equiv} (d_{1,1} - d_{1,0})$ since $(d_{0,1} - d_{0,0}) = r$ and $(d_{1,1} - d_{1,0}) = 2r$. In the next move, if Spoiler chooses the position 0 in w_{1-j}, Duplicator can respond with the position 0 in w_j.

(3) Spoiler chooses the position 2 in w_0, Duplicator can respond with the position 0 in w_1. We have $(d_{0,2} - d_{0,0}) \overset{S}{\equiv} (d_{1,0} - d_{1,0})$ since $(d_{0,2} - d_{0,0}) = (d_{1,0} - d_{1,0}) = 0$.

(4) Spoiler chooses the position $i\,(i \geq 3)$ in w_0, Duplicator can respond with the position $i-2$ in w_1. We have $(d_{0,i} - d_{0,0}) \overset{S}{\equiv} (d_{1,i-2} - d_{1,0})$ since $(d_{0,i} - d_{0,0}) \geq r$ and $(d_{1,i-2} - d_{1,0}) \geq r$. In the next move, if Spoiler chooses a position $0 \leq j \leq i-3$ in w_1, Duplicator can respond with the position $j+2$ in w_0.

(5) Spoiler chooses the position $i\,(i \geq 2)$ in w_1, Duplicator can respond with the position $i+2$ in w_0. We have $(d_{0,i+2} - d_{0,0}) \overset{S}{\equiv} (d_{1,i} - d_{1,0})$ since $(d_{0,i+2} - d_{0,0}) \geq r$ and $(d_{1,i} - d_{1,0}) \geq r$. In the next move, if Spoiler chooses the position 0 (respectively, 1) in w_0, Duplicator can respond with the position 0 (respectively, 1) in w_1. If Spoiler chooses a position $2 \leq j \leq i+1$ in w_0, Duplicator can respond with the position $j-2$ in w_1.

After the first round, the new game configuration is either $(0,0)$, or $(1,1)$, or $(i+2,i)\,(i \geq 0)$. We shall show that Duplicator can win the remaining $(k-1)$ rounds from these configurations. If the new configuration is $(0,0)$, by induction hypothesis, we have $w_0 \sim^S_{k-1} w_1$,

[5] Given a pair (i_0,i_1) of positions, we underline $i_l\,(l \in \{0,1\})$ to denote that Spoiler chooses the position i_l in w_l and Duplicator responds with the position i_{1-l} in w_{1-l}.

	Case 1	Case 2	Case 3	Case 4	Case 5
Move 1	$(\underline{0},0)$ or $(0,\underline{0})$	$(\underline{1},1)$ or $(1,\underline{1})$	$(\underline{2},0)$	$(\underline{i},i-2)$, $i \geq 3$	$(i+2,\underline{i})$, $i \geq 2$
Move 2	-	$(0,\underline{0})$ if $(\underline{1},1)$, $(\underline{0},0)$ if $(1,\underline{1})$	-	$(j+2,\underline{j})$, $0 \leq j \leq i-3$	$(\underline{0},0),(\underline{1},1)$ or $(\underline{j},j-2)$, $2 \leq j \leq i+1$

Table 3.2 The winning strategy for Duplicator in the first round

which means that Duplicator can win the remaining $(k-1)$ rounds. If the new configuration is $(1,1)$, we have $(w_0[1:])_{+r} = w_1[1:]$, by Lemma 2, we have $(w_0,1) \sim^S_{k-1} (w_1,1)$. If the new configuration is $(i+2,i)\,(i \geq 0)$, by Lemma 2 again, we have $(w_0,i+2) \sim^S_{k-1} (w_1,i)$.

In the following we show that $\mathsf{F}_{=0}\top$ is not definable in weakMTL over finite data words. It is similar to the case for infinite data words. Let $S \subseteq \mathbb{Z}$ be a finite set, and let $r,s > 0$ be defined as above. For every $k \in \mathbb{N}$, define $w_0 = (s)(s+r,s)^{k+1}_{+2r}$ and $w_1 = (s)(s+2r,s+r)^{k}_{+2r}$. It is easily seen that $w_0 \models \mathsf{F}_{=0}\top$ and $w_1 \not\models \mathsf{F}_{=0}\top$. We shall show that $w_0 \sim^S_k w_1$.

We prove, by induction on k, that for every $l \geq k$, $w'_0 \sim^S_k w'_1$, where $w'_0 = (s)(s+r,s)^{l+1}_{+2r}$ and $w'_1 = (s)(s+2r,s+r)^{l}_{+2r}$. It is clear for the case $k = 0$. Let $k \geq 1$ and $l \geq k$. We show that Duplicator wins the game $\mathrm{wMG}^S_k(w'_0,0,w'_1,0)$. In the first round, we can adapt the winning strategy for Duplicator in Table 3.2. After the first round, if the new configuration is $(0,0)$, by induction hypothesis, we have $w'_0 \sim^S_{k-1} w'_1$. If the new configuration is $(1,1)$, by Lemma 3, we have $(w'_0,1) \sim^S_{k-1} (w'_1,1)$. If the new configuration is $(i+2,i)\,(i \geq 0)$, by Lemma 2, we have $(w'_0,i+2) \sim^S_{k-1} (w'_1,i)$. □

3.4 The Ehrenfeucht–Fraïssé game for TPTL

In Section 3.1, we define the Ehrenfeucht–Fraïssé game for MTL. Using it we prove several results about the expressive power of MTL. To study the expressive power of TPTL, we define the Ehrenfeucht–Fraïssé game for TPTL in this section.

The *until rank* of a TPTL-formula φ, denoted by $\mathsf{Rank}(\varphi)$, is defined analogously to that of MTL-formulas, where we additionally define $\mathsf{Rank}(x \sim c) = 0$ and $\mathsf{Rank}(x.\varphi) = \mathsf{Rank}(\varphi)$. Let $S \subseteq \mathbb{Z}$, and let $k \in \mathbb{N}$, we define

$$\mathsf{TPTL}^S = \{\varphi \in \mathsf{TPTL} \mid \text{for every constraints } x \sim c \text{ in } \varphi, c \in S\},$$
$$\mathsf{TPTL}_k = \{\varphi \in \mathsf{TPTL} \mid \mathsf{Rank}(\varphi) \leq k\}.$$

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

$$\mathsf{TPTL}^{r,S} = \mathsf{TPTL}^r \cap \mathsf{TPTL}^S,$$
$$\mathsf{TPTL}^{r,S}_k = \mathsf{TPTL}^r \cap \mathsf{TPTL}^S \cap \mathsf{TPTL}_k,$$

where $S \subseteq \mathbb{Z}$ and $r, k \in \mathbb{N}$.

Lemma 5. *For every finite $S \subseteq \mathbb{Z}$, every $r \in \mathbb{N}$ and every $k \in \mathbb{N}$, there are only finitely many formulas in $\mathsf{TPTL}^{r,S}_k$ up to equivalence.*

Proof. This lemma is proved in the much same way as Lemma 1. Fix a finite set $S \subseteq \mathbb{Z}$ and a number $r \in \mathbb{N}$. So there are only finitely many different constraint formulas $x \sim c$.

We prove this lemma by induction on k. For $k = 0$, no until modality will be used, the number of formulas built from $\top$, propositions and constraint formulas using Boolean connectives and freeze quantifiers "$x_i.$" $(1 \leq i \leq r)$ is finite up to equivalence. Suppose that it holds for k, we will prove it for $k+1$. Define

$$R = \{\varphi_1 \mathsf{U} \varphi_2 \mid \varphi_1, \varphi_2 \in \mathsf{TPTL}^{r,S}_k\} \cup \mathsf{TPTL}^{r,S}_k.$$

It is easily seen that R is a finite set up to equivalence. The formulas obtained by using Boolean connectives and freeze quantifiers on the formulas in R are also finitely many up to equivalence. Hence, $\mathsf{TPTL}^{r,S}_{k+1}$ is a finite set up to equivalence. □

Definition 4. 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, let ν_0, ν_1 be two register valuations, and let $S \subseteq \mathbb{Z}$, $r \in \mathbb{N}$, $k \in \mathbb{N}$. We say that $(w_0[i_0:], \nu_0)$ and $(w_1[i_1:], \nu_1)$ are $\mathsf{TPTL}^{r,S}_k$-equivalent, written $(w_0, i_0, \nu_0) \equiv^{r,S}_k (w_1, i_1, \nu_1)$, if for every $\varphi \in \mathsf{TPTL}^{r,S}_k$, $(w_0, i_0, \nu_0) \models \varphi$ if and only if $(w_1, i_1, \nu_1) \models \varphi$.

We will write $w_0 \equiv^{r,S}_k w_1$ if $(w_0, 0, \overline{0}) \equiv^{r,S}_k (w_1, 0, \overline{0})$ in the following.

The Ehrenfeucht–Fraïssé game for TPTL is played by Spoiler and Duplicator on two data words w_0 and w_1 with a finite set of register variables $\{x_1, \ldots, x_r\}$ and a finite set of constraint numbers $S \subseteq \mathbb{Z}$. 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. Let ν be a register valuation, let Y be a set of register variables and let $d \in \mathbb{N}$. Define the register valuation

$$\nu[Y \mapsto d](x) = \begin{cases} d & \text{if } x \in Y, \\ \nu(x) & \text{otherwise.} \end{cases}$$

In each round of the game (suppose the current game configuration is (i_0, ν_0, i_1, ν_1)):

(1) Spoiler chooses a subset $Y \subseteq \{x_1, \ldots, x_r\}$ and sets $\nu'_t = \nu_t[Y \mapsto d_{t,i_t}]$ $(t \in \{0,1\})$.

(2) Spoiler chooses an index $l \in \{0,1\}$ and a position $j_l > i_l$ in data word w_l.

(3) Duplicator responds with a position $j_{1-l} > i_{1-l}$ in data word w_{1-l} such that if $j_l = i_l + 1$, then $j_{1-l} = i_{1-l} + 1$. Then Spoiler chooses between one of the following two options:

- The new configuration becomes $(j_0, \nu'_0, j_1, \nu'_1)$.
- Spoiler chooses a position $i_{1-l} < j'_{1-l} < j_{1-l}$ in w_{1-l}, Duplicator responds with a position $i_l < j'_l < j_l$ in w_l, and the new configuration becomes $(j'_0, \nu'_0, j'_1, \nu'_1)$.

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, with the register variables set $\{x_1, \ldots, x_r\}$ and constraint numbers set S.

The *winning condition for Duplicator* is defined inductively. We say that Duplicator wins the 0-round EF-game $\mathrm{TG}_0^{r,S}(w_0, i_0, \nu_0, w_1, i_1, \nu_1)$ if $P_{0,i_0} = P_{1,i_1}$, and for all constraints $x \sim c$ where $x \in \{x_1, \ldots, x_r\}$ and $c \in S$, $(w_0, i_0, \nu_0) \models x \sim c$ if and only if $(w_1, i_1, \nu_1) \models x \sim c$. Duplicator wins the $(k+1)$-round EF-game $\mathrm{TG}_{k+1}^{r,S}(w_0, i_0, \nu_0, w_1, i_1, \nu_1)$ if she wins the 0-round EF-game $\mathrm{TG}_0^{r,S}(w_0, i_0, \nu_0, w_1, i_1, \nu_1)$, and either i_0 and i_1 are the last position of w_0 and w_1, respectively (Spoiler has no position to choose), or for every choice of moves of Spoiler in the first round, Duplicator can respond correctly and wins the k-round EF-game $\mathrm{TG}_k^{r,S}(w_0, n_0, \nu'_0, w_1, n_1, \nu'_1)$, where $(n_0, \nu'_0, n_1, \nu'_1)$ 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 the k-round EF-game $\mathrm{TG}_k^{r,S}(w_0, i_0, \nu_0, w_1, i_1, \nu_1)$. It follows easily that if $(w_0, i_0, \nu_0) \sim_k^{r,S} (w_1, i_1, \nu_1)$, then $(w_0, i_0, \nu_0) \sim_m^{r,S} (w_1, i_1, \nu_1)$ for all $m < k$. We will write $w_0 \sim_k^{r,S} w_1$ if $(w_0, 0, \overline{0}) \sim_k^{r,S} (w_1, 0, \overline{0})$ in the following.

Theorem 6. *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. For every finite $S \subseteq \mathbb{Z}$, every $r \in \mathbb{N}$ and every $k \in \mathbb{N}$, $(w_0, i_0, \nu_0) \equiv_k^{r,S} (w_1, i_1, \nu_1)$ if and only if $(w_0, i_0, \nu_0) \sim_k^{r,S} (w_1, i_1, \nu_1)$.*

Proof. Fix a finite set of register variables $\{x_1, \ldots, x_r\}$ and a finite set of constraint numbers $S \subseteq \mathbb{Z}$. We prove this theorem by induction on k. It is clear for $k = 0$. Suppose that this theorem holds for k. We prove that it also holds for $k+1$.

For the direction "$\Rightarrow$", we give a proof by contradiction. Suppose that $(w_0, i_0, \nu_0) \equiv_{k+1}^{r,S} (w_1, i_1, \nu_1)$ holds and $(w_0, i_0, \nu_0) \sim_{k+1}^{r,S} (w_1, i_1, \nu_1)$ does not hold. Then Spoiler wins the game $\mathrm{TG}_{k+1}^{r,S}(w_0, i_0, \nu_0, w_1, i_1, \nu_1)$. By induction hypothesis, it is easily seen that $(w_0, i_0, \nu_0) \sim_0^{r,S} (w_1, i_1, \nu_1)$, i.e., Duplicator wins the 0-round EF-game from the configuration (i_0, ν_0, i_1, ν_1). In

the following we show that Duplicator can always win the remaining k-round EF-game, a contradiction. Without loss of generality suppose that Spoiler chooses a subset $Y \subseteq \{x_1, \ldots, x_r\}$ and sets $v_0' = v_0'[Y \mapsto d_{0,i_0}]$ and $v_1' = v_1'[Y \mapsto d_{1,i_1}]$, and then chooses a position $i_0' > i_0$ in w_0. For each $i_0 < j \leq i_0'$, define

$$\varphi_j = \bigwedge \{\varphi \in \mathsf{TPTL}_k^{r,S} \mid (w_0, j, v_0') \models \varphi\}.$$

Note that φ_j is well-defined since $\mathsf{TPTL}_k^{r,S}$ is a finite set up to equivalence by Lemma 5. Define the $\mathsf{TPTL}_{k+1}^{r,S}$-formula

$$\varphi = y_1 \ldots y_h. \left(\Big(\bigvee_{i_0 < j < i_0'} \varphi_j \Big) \mathsf{U}\, \varphi_{i_0'} \right)$$ [6]

where $y_1, \ldots, y_h$ are all register variables in Y. Clearly, $(w_0, i_0, v_0) \models \varphi$. We have $(w_1, i_1, v_1) \models \varphi$, since $\mathsf{Rank}(\varphi) \leq k+1$ and $(w_0, i_0, v_0) \equiv_{k+1}^{r,S} (w_1, i_1, v_1)$. Hence there exists $i_1' > i_1$ such that $(w_1, i_1', v_1') \models \varphi_{i_0'}$ and for all $i_1 < i_1'' < i_1'$, $(w_1, i_1'', v_1') \models \bigvee_{i_0 < j < i_0'} \varphi_j$. Hence Duplicator can respond with the position i_1' in w_1. If they start a new round from the configuration (i_0', v_0', i_1', v_1'), we know by $(w_1, i_1', v_1') \models \varphi_{i_0'}$ and the definition of $\varphi_{i_0'}$ that $(w_0, i_0', v_0') \equiv_k^{r,S} (w_1, i_1', v_1')$. By induction hypothesis, $(w_0, i_0', v_0') \sim_k^{r,S} (w_1, i_1', v_1')$. On the other hand, if Spoiler chooses a position $i_1 < i_1'' < i_1'$ in w_1, we know by $(w_1, i_1'', v_1') \models \bigvee_{i_0 < j < i_0'} \varphi_j$ that there is some $i_0 < i_0'' < i_0'$ such that $(w_1, i_1'', v_1') \models \varphi_{i_0''}$. Hence Duplicator can respond with the position i_0'' in w_0. If they start a new round from the configuration $(i_0'', v_0', i_1'', v_1')$, we know by the definition of $\varphi_{i_0''}$ that $(w_0, i_0'', v_0') \equiv_k^{r,S} (w_1, i_1'', v_1')$, and thus, by induction hypothesis, that $(w_0, i_0'', v_0') \sim_k^{r,S} (w_1, i_1'', v_1')$.

For the direction "$\Leftarrow$", suppose $(w_0, i_0, v_0) \sim_{k+1}^{r,S} (w_1, i_1, v_1)$, i.e., Duplicator wins the game. We show that $(w_0, i_0, v_0) \equiv_{k+1}^{r,S} (w_1, i_1, v_1)$. Let $\varphi \in \mathsf{TPTL}_{k+1}^{r,S}$. If $\mathsf{Rank}(\varphi) \leq k$, then, by induction hypothesis, $(w_0, i_0, v_0) \models \varphi$ if and only if $(w_1, i_1, v_1) \models \varphi$. If $\mathsf{Rank}(\varphi) = k+1$, we assume that $\varphi = y_1 \ldots y_h.(\varphi_1 \mathsf{U}\, \varphi_2)$, where $\varphi_1, \varphi_2 \in \mathsf{TPTL}_k^{r,S}$, which is the most interesting case. The proof for the other cases is easy. We must show that $(w_0, i_0, v_0) \models \varphi$ if and only if $(w_1, i_1, v_1) \models \varphi$. Suppose $(w_0, i_0, v_0) \models \varphi$, we prove $(w_1, i_1, v_1) \models \varphi$. The other direction can be proved analogously.

Let $Y = \{y_1, \ldots, y_h\}$. Then $(w_0, i_0, v_0) \models y_1 \ldots y_h.(\varphi_1 \mathsf{U}\, \varphi_2)$ if and only if $(w_0, i_0, v_0[Y \mapsto d_{0,i_0}]) \models \varphi_1 \mathsf{U}\, \varphi_2$ if and only if there is a position $i_0' > i_0$ such that $(w_0, i_0', v_0[Y \mapsto d_{0,i_0}]) \models \varphi_2$, and for all $i_0 < j < i_0'$, $(w_0, j, v_0[Y \mapsto d_{0,i_0}]) \models \varphi_1$.

Assume Spoiler chooses the set Y and sets the register valuations $v_0' = v_0[Y \mapsto d_{0,d_0}]$, $v_1' = v_1[Y \mapsto d_{1,d_1}]$, and then chooses $i_0' > i_0$ in w_0. Since Duplicator wins the game, she can respond

[6] $\varphi = y_1 \ldots y_h.(\bot \mathsf{U}\, \varphi_{i_0'})$ if $i_0' = i_0 + 1$, i.e., there are no positions between position i_0 and position i_0'.

with a position $i_1' > i_1$ in w_1 such that $(w_0, i_0', v_0') \sim_k^{r,S} (w_1, i_1', v_1')$. By induction hypothesis, we have $(w_1, i_1', v_1') \models \varphi_2$. On the other hand, if Spoiler chooses a position $i_1 < i_1'' < i_1'$ in w_1, Duplicator can respond with a position $i_0 < i_0'' < i_0'$ such that $(w_0, i_0'', v_0') \sim_k^{r,S} (w_1, i_1'', v_1')$. Hence, by induction hypothesis, $(w_1, i_1'', v_1') \models \varphi_1$. Since Spoiler can choose an arbitrary position between i_1 and i_1', we know that $(w_1, j, v_1') \models \varphi_1$ for all $i_1 < j < i_1'$. Adding $(w_1, i_1', v_1') \models \varphi_2$ we have $(w_1, i_1, v_1') \models \varphi_1 \mathsf{U} \varphi_2$, i.e., $(w_1, i_1, v_1) \models y_1 \ldots y_h.(\varphi_1 \mathsf{U} \varphi_2)$. □

Theorem 7. *Let* $\mathbf{C}$ *be a class of data words. For every finite* $S \subseteq \mathbb{Z}$*, every* $r \in \mathbb{N}$ *and every* $k \in \mathbb{N}$*, the following are equivalent:*

(1) $\mathbf{C}$ *is not definable in* $\mathsf{TPTL}_k^{r,S}$.

(2) There exist two data words $w_0 \in \mathbf{C}$ *and* $w_1 \notin \mathbf{C}$ *such that* $w_0 \sim_k^{r,S} w_1$.

Proof. From (1) to (2), we give a proof by contradiction. Assume (1) holds and (2) does not hold. Then for every pair of data words w_0 and w_1, if $w_0 \in \mathbf{C}$ and $w_1 \notin \mathbf{C}$, then $w_0 \not\sim_k^{r,S} w_1$. By Theorem 6, this implies $w_0 \not\equiv_k^{r,S} w_1$. So if $w_0 \equiv_k^{r,S} w_1$, then $w_0 \in \mathbf{C}$ if and only if $w_1 \in \mathbf{C}$. Let w be a data word, we define

$$\varphi_w = \bigwedge \{\varphi \in \mathsf{TPTL}_k^{r,S} \mid w \models \varphi\},$$

and

$$\Phi = \bigvee_{w \in \mathbf{C}} \varphi_w.$$

Note that there are only finitely many formulas in $\mathsf{TPTL}_k^{r,S}$ up to equivalence, so both φ_w and Φ are well-defined. We next show that $\mathbf{C}$ is definable by the formula Φ, which contradicts the assumption. For every data word w, if $w \in \mathbf{C}$, then $w \models \Phi$ by the definition of Φ. If $w \models \Phi$, there must exists some data word $w' \in \mathbf{C}$ such that $w \models \varphi_{w'}$. This implies that w and w' satisfy the same $\mathsf{TPTL}_k^{r,S}$-formulas, i.e., $w \equiv_k^{r,S} w'$. From $w' \in \mathbf{C}$ we can know that $w \in \mathbf{C}$.

From (2) to (1), suppose that there are two data words $w_0 \in \mathbf{C}$ and $w_1 \notin \mathbf{C}$ such that $w_0 \sim_k^{r,S} w_1$. By Theorem 6, we have $w_0 \equiv_k^{r,S} w_1$. This means that for every $\mathsf{TPTL}_k^{r,S}$-formula φ, $w_0 \models \varphi$ if and only if $w_1 \models \varphi$. Hence, $\mathbf{C}$ is not definable in $\mathsf{TPTL}_k^{r,S}$. □

3.5 Application of the EF-game for TPTL

In this section, we consider the expressive power of several fragments of TPTL by restriction of the until rank, the set of constraint numbers and the number of register variables. Similar to MTL, we show that the until rank hierarchy and the constraint hierarchy are still strict for TPTL. First we give a lemma that is useful for the proof.

Lemma 6. *Let $S \subseteq \mathbb{Z}$ be a finite set, and let $r \in \mathbb{N}$. Let w_0, w_1 be two data words such that $|w_0| = |w_1|$ and for every $i \geq 0$, $P_{0,i} = P_{1,i}$ and for every $i' > i \geq 0$,*

$$(d_{0,i'} - d_{0,i}) \overset{S}{\equiv} (d_{1,i'} - d_{1,i}). \tag{3.4}$$

For any two register valuations ν_0 and ν_1, if for every $i \geq 0$ and every constraint $x \sim c$, where $x \in \{x_1, \ldots, x_r\}$ and $c \in S$,

$$(w_0, i, \nu_0) \models x \sim c \text{ if and only if } (w_1, i, \nu_1) \models x \sim c, \tag{3.5}$$

then for every $k \in \mathbb{N}$, $(w_0, 0, \nu_0) \equiv_k^{r,S} (w_1, 0, \nu_1)$.

Proof. Let $k \in \mathbb{N}$, we show that Duplicator wins the game $\mathrm{TG}_k^{r,S}(w_0, 0, \nu_0, w_1, 0, \nu_1)$.

In each round, Duplicator can always respond with the same position that Spoiler chooses in the other data word. After one round, suppose the new game configuration is (j, ν_0', j, ν_1'), where $j > 0$. It is easily seen that $P_{0,j} = P_{1,j}$, since $P_{0,i} = P_{1,i}$ for every $i \geq 0$. We need to show that $(w_0, j, \nu_0') \models x \sim c$ if and only if $(w_1, j, \nu_1') \models x \sim c$ for every constraint $x \sim c$, where $x \in \{x_1, \ldots, x_r\}$ and $c \in S$. Let $x \sim c$ be a constraint. There are two cases:

(1) The register variable x is not freezed by Spoiler until now, i.e., $\nu_0'(x) = \nu_0(x)$ and $\nu_1'(x) = \nu_1(x)$, by (3.5), we have $(w_0, j, \nu_0') \models x \sim c$ if and only if $(w_1, j, \nu_1') \models x \sim c$.

(2) If $\nu_0'(x)$ and $\nu_1'(x)$ are obtained by freezing x to the data values in position j' $(0 \leq j' < j)$ in w_0 and w_1, respectively, i.e., $\nu_0'(x) = d_{0,j'}$ and $\nu_1'(x) = d_{1,j'}$, by (3.4), we have $(d_{0,j} - d_{0,j'}) \overset{S}{\equiv} (d_{1,j} - d_{1,j'})$. Hence, $(w_0, j, \nu_0') \models x \sim c$ if and only if $(d_{0,j} - d_{0,j'}) \sim c$ if and only if $(d_{1,j} - d_{1,j'}) \sim c$ if and only if $(w_1, j, \nu_1') \models x \sim c$.

□

Corollary 4. *For every $n \in \mathbb{N}$, w and w_{+n} satisfy the same* TPTL*-formulas.*

Proposition 6. *Let $S \subseteq \mathbb{Z}$ be a finite set, and let $C = \max\{|c| \mid c \in S\}$. Then for every finite data word w, there exists a finite data word u such that $|w| = |u|$ and*

- *all data values in u are bound by $|u| \cdot (C+1)$,*
- *w and u satisfy the same formulas in* TPTL^S.

Proof. Suppose that $|w| = n$. Let $\pi := a_1, \ldots, a_n$ be an enumeration of all data values in w such that $a_i \leq a_{i+1}$ for all $1 \leq i < n$. For each $1 < i \leq n$, define $\delta_i = a_i - a_{i-1}$. We define a

new sequence $\pi' := b_1, \ldots, b_n$ inductively as follows: $b_1 = 0$ and for all $1 < i \leq n$,

$$b_i = \begin{cases} b_{i-1} + \delta_i & \text{if } \delta_i \leq C, \\ b_{i-1} + C + 1 & \text{if } \delta_i > C. \end{cases}$$

Intuitively, the data values b_i are obtained by shrinking the a_i so that the largest difference between two different data values is bounded by $C+1$. We obtain a new data word u by replacing in w every data value a_i by $b_i\,(1 \leq i \leq n)$. Note that $b_n \leq (n-1)(C+1)$. Hence, all data values in u are bound by $|u| \cdot (C+1)$.

Let d_{j_1} and d_{j_2} (respectively, d'_{j_1} and d'_{j_2}) be the data values in the $j_1{}^{\text{th}}$ position and $j_2{}^{\text{th}}$ position of w (respectively, u), respectively, where $0 \leq j_1 < j_2 < n$. Without loss of generality we can assume $d_{j_1} \leq d_{j_2}$. Let $a_i, \ldots, a_{i+k}$ (respectively, $b_i, \ldots, b_{i+k}$) be the sub-sequence in π (respectively, π') such that $a_i = d_{j_1}$ and $a_{i+k} = d_{j_2}$ (respectively, $b_i = d'_{j_1}$ and $b_{i+k} = d'_{j_2}$). If $a_{t+1} - a_t \leq C$ for every $i \leq t < i+k$, then, by the definition of π', we have $b_{t+1} - b_t = a_{t+1} - a_t$ for every $i \leq t < i+k$, which implies $d_{j_2} - d_{j_1} = d'_{j_2} - d'_{j_1}$. If $a_{t+1} - a_t > C$ for some $i \leq t < i+k$ (hence, $d_{j_2} - d_{j_1} > C$), then, by the definition of π', we have $b_{t+1} - b_t = C+1$, which implies $d'_{j_2} - d'_{j_1} > C$. Finally, we can conclude that $d_{j_2} - d_{j_1} \overset{S}{\equiv} d'_{j_2} - d'_{j_1}$ for any j_1, j_2 such that $0 \leq j_1 < j_2 < n$. By Lemma 6, this implies that $w \equiv_k^{r,S} u$ for every $r \in \mathbb{N}$ and every $k \in \mathbb{N}$, i.e., w and u satisfy the same formulas in TPTL^S. □

3.5.1 Effects on the expressiveness by restriction of the number of register variables

In Section 3.2, we show that TPTL is strictly more expressive than MTL. The register variables play a crucial role in reaching this greater expressiveness. We want to explore more deeply whether the number of register variables allowed in a TPTL-formula has an impact on the expressive power of the logic. We are able to show that there is a strict increase in expressiveness when allowing two register variables instead of just one. For the general case, we conjecture that for every $r \in \mathbb{N}$, TPTL^{r+1} is strictly more expressive than TPTL^r.

Theorem 8. TPTL^2 *is strictly more expressive than* TPTL^1 *over both infinite and finite data words.*

Proof. In the following we show that the TPTL^2-formula

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

is not definable in TPTL^1 over both infinite and finite data words. To prove this, it is enough to show that φ is not definable in $\mathsf{TPTL}_k^{1,S}$ for every finite $S \subseteq \mathbb{Z}$ and $k \in \mathbb{N}$.

We first prove φ is not definable in TPTL^1 over infinite data words. Let $S \subseteq \mathbb{Z}$ be a finite set 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 as follows (see Fig. 3.6):

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

In data word w_0, the data value $s+2r$ in position 1 is larger than the data value s in position 0, and the data value $s+r$ in position $k+3$ is between s and $s+2r$. So we have $w_0 \models \varphi$. In data word w_1, there are no data values between s and $s+2r$ after position 1. So we have $w_1 \not\models \varphi$. We show that $w_0 \sim_k^{1,S} w_1$ in the following. By Theorem 7, we can know that φ is not definable in $\mathsf{TPTL}_k^{1,S}$. It is clear for the case $k = 0$. We assume $k \geq 1$.

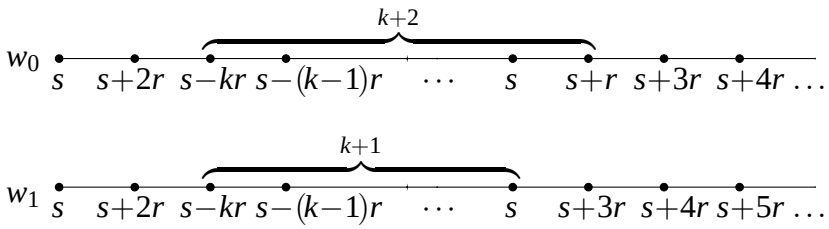

Fig. 3.6 The two data words w_0 and w_1

Without loss of generality we use the tuple (i_0, n_0, i_1, n_1) to denote the game configuration (i_0, v_0, i_1, v_1), where i_0, i_1 are positions in w_0, w_1, respectively, and $v_0(x_1) = n_0$ and $v_1(x_1) = n_1$ (note that only one register variable x_1 can be used in the game).

The initial game configuration is $(0, s, 0, s)$. In the first round, if Spoiler chooses a position $i\,(i \geq 1)$ in $w_l\,(l \in \{0,1\})$, Duplicator can respond with the same position i in w_{1-l}. In the next move, if Spoiler chooses 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 $(i, s, i, s)\,(i \geq 2)$, or $(1, s, 1, s)$. If the new configuration is $(i, s, i, s)\,(i \geq 2)$, by Lemma 6, we can know that $(w_0, i, v_0') \sim_{k-1}^{1,S} (w_1, i, v_1')$, where $v_0'(x_1) = s$ and $v_1'(x_1) = s$, i.e., Duplicator wins the remaining $(k-1)$ rounds. If the new configuration is $(1, s, 1, s)$, there are two cases in the second round:

(1) Spoiler does not freeze x_1 to the data value $s+2r$ in the current position. This case is easy. Duplicator can always respond with the same position that Spoiler chooses in the other data word. By Lemma 6, Duplicator can win the remaining rounds.

(2) Spoiler freezes x_1 to the data value $s+2r$, i.e., the configuration becomes $(1, s+2r, 1, s+2r)$. Let $v_0''(x_1) = s+2r$ and $v_1''(x_1) = s+2r$. We show that $(w_0, 1, v_0'') \sim_{k-1}^{1,S} (w_1, 1, v_1'')$. There are three cases in the next round:

(a) Spoiler chooses the position 2 in w_l $(l \in \{0,1\})$, Duplicator can respond with the same position in w_{1-l}. We shall show that Duplicator wins the remaining $(k-2)$ rounds from the new configuration $(2, s+2r, 2, s+2r)$. This is equivalent to show that Duplicator wins the game $\mathrm{TG}_h^{1,S}(w_0', 0, v_0'', w_1', 0, v_1'')$, where $h = k-2$ and

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

By induction on h, we can show that $(w_0', 0, v_0'') \sim_h^{1,S} (w_1', 0, v_1'')$.

(b) Spoiler chooses position i $(i \geq 3)$ in w_0, Duplicator can respond with the position $i-1$ in w_1. In the next move, if Spoiler chooses a position $1 < j < i-1$ in w_1, Duplicator can respond with the position $j+1$ in w_0. After this round, the new configuration is $(i, s+2r, i-1, s+2r)$ $(i \geq 3)$. By Lemma 6, we can know that $(w_0, i, v_0'') \sim_{k-2}^{1,S} (w_1, i-1, v_1'')$, i.e., Duplicator wins the remaining $(k-2)$ rounds.

(c) Spoiler chooses position i $(i \geq 3)$ in w_1, Duplicator can respond with the position $i+1$ in w_0. In the next move, if Spoiler chooses the position 2 in w_0, Duplicator can also respond with the position 2 in w_1. If Spoiler chooses a position $2 < j \leq i$ in w_0, Duplicator can respond with the 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 $(w_0, 2, v_0'') \sim_{k-2}^{1,S} (w_1, 2, v_1'')$, or $(i+1, s+2r, i, s+2r)$ $(i \geq 2)$, then by Lemma 6 we have $(w_0, i+1, v_0'') \sim_{k-2}^{1,S} (w_1, i, v_1'')$.

To prove 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 above. Define

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

Similar to the proof for the infinite case, it is easy to check that $w_0 \sim_k^{1,S} w_1$. □

3.5.2 Effects on the expressiveness by restriction of the until rank and the set of constraint numbers

In the following proposition we show that the until rank hierarchy for TPTL is strict over both infinite and finite data words.

Proposition 7. *For every $k \in \mathbb{N}$, TPTL_{k+1} is strictly more expressive than TPTL_k over both infinite and finite data words.*

Proof. Let $\varphi_1 = (p \wedge \mathsf{X}p)$, and for every $k \geq 1$, let $\varphi_{k+1} = (p \wedge \mathsf{X}\varphi_k)$, where p is an atomic proposition. We have $\mathsf{Rank}(\varphi_k) = k$. We shall show that φ_k is not definable in TPTL_{k-1} over both infinite and finite data words. It is enough to show that φ_k is not definable in $\mathsf{TPTL}_{k-1}^{r,S}$ for every finite $S \subseteq \mathbb{Z}$ and every $r \in \mathbb{N}$.

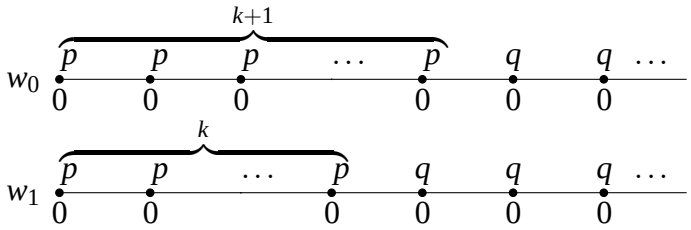

Fig. 3.7 The data words w_0 and w_1

First we prove φ_k is not definable in $\mathsf{TPTL}_{k-1}^{r,S}$ over infinite data words. Define two infinite data words $w_0 = (p,0)_{+0}^{k+1}(q,0)_{+0}^{\omega}$ and $w_1 = (p,0)_{+0}^{k}(q,0)_{+0}^{\omega}$, where p,q are propositions (see Fig. 3.7). We see that $w_0 \models \varphi_k$ and $w_1 \not\models \varphi_k$.

In the following we prove by induction on k that $w_0 \sim_{k-1}^{r,S} w_1$, then by Theorem 7, we can know that φ_k is not definable in $\mathsf{TPTL}_{k-1}^{r,S}$. It is easy for the case $k = 1$. Suppose $k \geq 2$. We give the winning strategy for Duplicator in the first round. There are three cases:

(1) Spoiler chooses the position 1 in w_0 or w_1, Duplicator can respond with the position 1 in the other data word.

(2) Spoiler chooses the position $i\,(i \geq 2)$ in w_0, Duplicator can respond with the position $i-1$ in w_1. In the next move, if Spoiler chooses a position $0 < j < i-1$ in w_1, Duplicator can respond with the position $j+1$ in w_0.

(3) Spoiler chooses the position $i\,(i \geq 2)$ in w_1, Duplicator can respond with the position $i+1$ in w_0. In the next move, if Spoiler chooses the position 1 in w_0, Duplicator can respond with the position 1 in w_1. If Spoiler chooses a position $2 \leq j \leq i$ in w_0, Duplicator can respond with the position $j-1$ in w_1.

Let υ be the register valuation that maps all register variables to 0. Note that the register valuation is always υ during the game (all register variables can only be freezed to the value 0). After the first round, the new game configuration is either $(1,\upsilon,1,\upsilon)$, then by induction hypothesis, we have $(w_0,1,\upsilon) \sim^{r,S}_{k-2} (w_1,1,\upsilon)$, or $(i+1,\upsilon,i,\upsilon)\,(i \geq 1)$, then by Lemma 6, we have $(w_0,i+1,\upsilon) \sim^{r,S}_{k-2} (w_1,i,\upsilon)$.

To prove φ_k is not definable in $\mathsf{TPTL}^{r,S}_{k-1}$ over finite data words, we define $w_0 = (p,0)^{k+1}_{+0}$ and $w_1 = (p,0)^{k}_{+0}$. Similar to the infinite case, we can show that $w_0 \sim^{r,S}_{k-1} w_1$. □

In Proposition 2 we show that the problem that whether an MTL_{k+1}-formula is definable in MTL_k is undecidable. Since every MTL-formula is equivalent to a TPTL-formula with the same until rank, we can adapt the proof for TPTL. The following proposition can be proved in much the same way as Proposition 2.

Proposition 8. *For every $k \geq 5$ (respectively, $k \geq 4$), the problem that whether a TPTL_{k+1}-formula is definable in TPTL_k is undecidable over infinite data words (respectively, finite data words).*

In the following we consider the effects on the expressive power of TPTL by restriction of the set of constraint numbers.

Lemma 7. *Let $S \subseteq \mathbb{Z}$, and let $n \in \mathbb{Z}$. If $\{n-1,n\} \cap S = \emptyset$ or $\{n,n+1\} \cap S = \emptyset$, then the formula $x.\mathsf{X}(x=n)$ is not definable in TPTL^{S} over both infinite and finite data words.*

Proof. The proof is similar to the proof of Lemma 4. We show that the formula $x.\mathsf{X}(x=n)$ is not definable in $\mathsf{TPTL}^{r,S'}_{k}$ for every finite $S' \subseteq S$, every $r \in \mathbb{N}$ and every $k \in \mathbb{N}$.

We first show that the formula is not definable in $\mathsf{TPTL}^{r,S'}_{k}$ over infinite data words. If $\{n,n+1\} \cap S = \emptyset$, then let $s,r > 0$ be such that $s+n > 0$ and all numbers in S' are less than $n+r$. We define two data words $w_0 = (s)(s+n)^{\omega}_{+r}$ and $w_1 = (s)(s+n+1)^{\omega}_{+r}$. If $\{n-1,n\} \cap S = \emptyset$, then let $s,r > 0$ be such that $s+n-1 > 0$ and all numbers in S' are less than $n+r-1$. We define $w_0 = (s)(s+n)^{\omega}_{+r}$ and $w_1 = (s)(s+n-1)^{\omega}_{+r}$. It is easily seen that $w_0 \models x.\mathsf{X}(x=n)$ and $w_1 \not\models x.\mathsf{X}(x=n)$. By Lemma 6, we have $w_0 \sim^{r,S'}_{k} w_1$ for every $k \in \mathbb{N}$. Then by Theorem 7, we can know that $x.\mathsf{X}(x=n)$ is not definable in $\mathsf{TPTL}^{r,S'}_{k}$ for every $k \in \mathbb{N}$.

To prove the formula is not definable in $\mathsf{TPTL}^{r,S'}_{k}$ over finite data words, we can adapt the proof for the infinite case. Let $k \in \mathbb{N}$, we can set $w_0 = (s)(s+n)^{k+1}_{+r}$ and $w_1 = (s)(s+n+1)^{k+1}_{+r}$, if $\{n,n+1\} \cap S = \emptyset$, or $w_0 = (s)(s+n)^{k+1}_{+r}$ and $w_1 = (s)(s+n-1)^{k+1}_{+r}$, if $\{n-1,n\} \cap S = \emptyset$. By Lemma 6, we have $w_0 \sim^{r,S'}_{k} w_1$. □

The following three propositions hold over both infinite data words and finite data words.

Proposition 9. *Let $S_1 \subseteq \mathbb{Z}$ and $S_2 \subseteq \mathbb{Z}$. $\mathsf{TPTL}^{S_1} \preccurlyeq \mathsf{TPTL}^{S_2}$ if and only if for every $n \in S_1$, either $n \in S_2$ or $\{n-1, n+1\} \subseteq S_2$.*

Proof. For the direction "$\Rightarrow$", suppose that there exists a number $m \in S_1$ such that $\{m-1, m\} \cap S_2 = \emptyset$ or $\{m, m+1\} \cap S_2 = \emptyset$. Then by Lemma 7, the TPTL^{S_1}-formula $x.\mathsf{X}(x = m)$ is not definable in TPTL^{S_2}, contrary to $\mathsf{TPTL}^{S_1} \preccurlyeq \mathsf{TPTL}^{S_2}$.

For the direction "$\Leftarrow$", let φ be a TPTL^{S_1}-formula. We can construct an equivalent TPTL^{S_2}-formula by replacing every constraint formula $x \sim m$ in φ, where $m \in S_1$ and $m \notin S_2$, with an equivalent constraint formula α, where

$$\alpha = \begin{cases} x \leq m-1 \vee x \geq m+1 & \text{if } x \sim m \text{ is } x \neq m, \\ x \geq m+1 & \text{if } x \sim m \text{ is } x > m, \\ x > m-1 & \text{if } x \sim m \text{ is } x \geq m, \\ x \leq m-1 & \text{if } x \sim m \text{ is } x < m, \\ x < m+1 & \text{if } x \sim m \text{ is } x \leq m. \end{cases}$$

□

For every $n \in \mathbb{Z}$, let $\mathsf{TPTL}^{\leq n} = \mathsf{TPTL}^{\{m \in \mathbb{Z} \mid m \leq n\}}$. By Lemma 7 and Proposition 9, we have the following two hierarchies for TPTL.

Proposition 10. (Linear Constraint Hierarchy of TPTL)
For any $n_1, n_2 \in \mathbb{Z}$, if $n_1 < n_2$, then $\mathsf{TPTL}^{\leq n_1} \prec \mathsf{TPTL}^{\leq n_2}$.

Proposition 11. (Lattice Constraint Hierarchy of TPTL)
Let $R \subseteq \mathbb{Z}$ be a set such that for every $n \in \mathbb{Z}$, $n \in R$ if and only if $n+1 \notin R$. Then $\langle \{\mathsf{TPTL}^S \mid S \subseteq R\}, \preccurlyeq \rangle$ constitutes a complete lattice in which

(i) the greatest element is TPTL^R,

(ii) the least element is $\mathsf{TPTL}^{\emptyset}$,

and for every $Q \subseteq \mathscr{P}(R)$, where $\mathscr{P}(R)$ is the power set of R,

(iii) $\bigwedge_{S \in Q} \mathsf{TPTL}^S = \mathsf{TPTL}^{\bigcap_{S \in Q} S}$,

(iv) $\bigvee_{S \in Q} \mathsf{TPTL}^S = \mathsf{TPTL}^{\bigcup_{S \in Q} S}$.

3.6 Summary of the relative expressive power

We conclude the relative expressive power of weakMTL, MTL and TPTL in Fig. 3.8. For LTL and TPTL, the semantics (strict or non-strict) has no effect on their expressive power, whereas MTL with strict semantics is strictly more expressive than MTL with non-strict semantics (weakMTL) over all data words. MTL is strictly less expressive than TPTL on data words. Actually, MTL is strictly less expressive than TPTL^1, and TPTL^1 is strictly less expressive than TPTL^2. It is still open that whether TPTL^{r+1} is strictly more expressive than TPTL^r for $r \geq 2$. The until rank for both MTL and TPTL is strict.

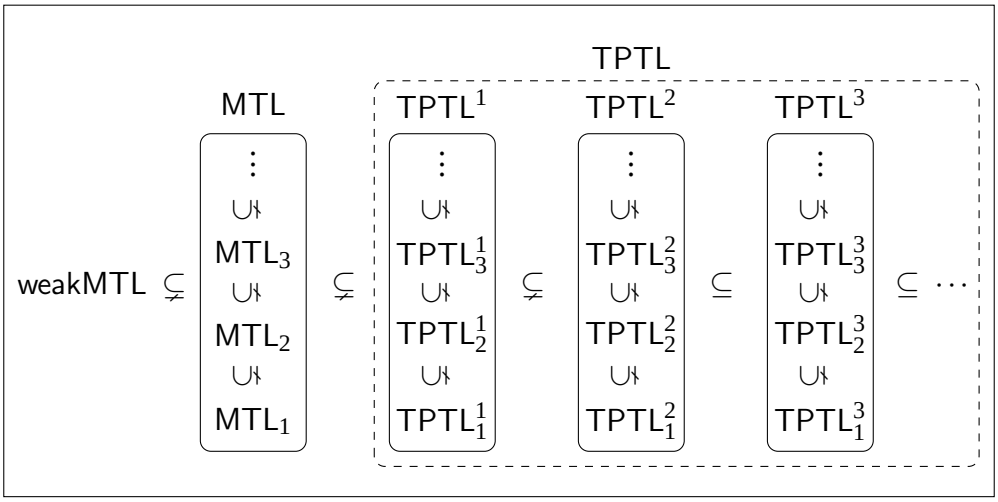

Fig. 3.8 The relative expressive power of MTL and TPTL

Chapter 4

The satisfiability problems for MTL and TPTL

In this chapter, we consider the satisfiability problems (SAT) for MTL and TPTL. A formula is satisfiable if it is satisfied by a data word. The satisfiability problem asks, given a formula φ, whether φ is satisfiable or not. More precisely, let $\mathscr{L}$ be a logic and let $\mathbf{C}$ be a class of data words, the *satisfiability problem for* $\mathscr{L}$ *over* $\mathbf{C}$ is:

Input: A formula $\varphi \in \mathscr{L}$.

Output: `yes` if there exists $w \in \mathbf{C}$ such that $w \models \varphi$, `no` otherwise.

We are interested in infinitary and finitary versions of the satisfiability problem, where $\mathbf{C}$ is the class of infinite data words and the class of finite data words, respectively.

The arithmetical hierarchy classifies problems based on the complexity of first-order arithmetic formulas that define them. The class Σ_1^0 in the arithmetical hierarchy consists of all problems that can be defined by a formula which begins with a sequence of existential quantifiers and followed by a formula with only bounded quantifiers in it. Σ_1^0 contains exactly all recursively enumerable sets. Analytical hierarchy is an extension of the arithmetical hierarchy, where second-order arithmetic formulas can be used to classify problems. The class Σ_1^1 in the analytical hierarchy consists of all problems that can be defined by a second-order arithmetic formula which begins with a sequence of second-order existential quantifiers and followed by a formula with no second-order quantifiers. Σ_1^1 contains highly undecidable problems, including nonarithmetical problems. For more details about the arithmetical hierarchy and the analytical hierarchy we refer the reader to [73].

In [7], Alur and Henzinger proved that infinitary SAT for TPTL is Σ_1^1-complete, even if one does not allow for propositions. The proof in the cited paper is by reduction of the recurrent

state problem for two-counter machines. However, one can easily adapt the proof for finitary SAT using a reduction of the halting problem for two-counter machines.

Theorem 9 ([7]). *For* TPTL*, infinitary SAT is* Σ_1^1*-complete and finitary SAT is* Σ_1^0*-complete.*

Proof. We use a reduction of the halting problem for two-counter machines to prove finitary SAT for TPTL is undecidable. This also implies the Σ_1^0-hardness of finitary SAT. To show that finitary SAT is in Σ_1^0, we only need to show that all satisfiable TPTL-formulas over finite data word are recursively enumerable. Let φ ba a TPTL-formula. We use $\mathbf{P}_\varphi$ to denote the set of all propositions occurring in φ. Observe that for each $n \in \mathbb{N}$, there are only finitely many data words w over $\mathbf{P}_\varphi$ such that $|w| \leq n$ and all data values in w are bounded by n. Define

$$S = \{(\varphi, n) \mid \varphi \text{ is a TPTL-formula and } n \in \mathbb{N}\}.$$

It is easily seen that S is recursively enumerable. Let π be an enumeration of S. For each (φ, n) in π, we check for every data word w over $\mathbf{P}_\varphi$ whether $w \models \varphi$, where $|w| \leq n$ and all data values in w are bounded by n. If there is a data word that satisfies φ, then we output φ. Otherwise, we check the next pair in π. In this way, we can enumerate all satisfiable TPTL-formulas. This procedure is effective, since for every pair (φ, n) there are only finitely many data words need to check and the path checking problem for TPTL over finite data words is decidable by Theorem 18 in Chapter 5. □

Remark 3. One can change the TPTL-formulas in the proof of Theorem 9 in [7] such that they only use one register variable. This means that Theorem 9 also holds for TPTL^1. However, by the result for MTL in Section 4.1, we can also conclude the same result for TPTL^1.

Every formula in MTL can effectively be translated into a TPTL-formula. Hence the the upper bound of infinitary SAT (respectively, finitary SAT) for TPTL also apply to infinitary SAT (respectively, finitary SAT) for MTL and other fragments of MTL and TPTL (we will not prove this in addition in the following proofs). We show that, for most of the logics in this chapter, infinitary SAT is Σ_1^1-complete and finitary SAT is Σ_1^0-complete. We also prove that finitary SAT and infinitary SAT coincide for positive TPTL and positive MTL, and SAT for existential TPTL and existential MTL are NP-complete. As a consequence, the Σ_1^1-hardness of infinitary SAT excludes the possibility to axiomatize validity for MTL and TPTL in a standard proof calculus system.

Generally, we prove the undecidability of infinitary SAT (respectively, finitary SAT) for a logic $\mathscr{L}$ by a reduction from the recurrent state problem (respectively, halting problem) of two-counter machines in the following way: For every two-counter machine $\mathbf{M}$, we construct a formula φ_{infin} (respectively, φ_{fin}) of $\mathscr{L}$ such that φ_{infin} (respectively, φ_{fin}) is satisfied by an

infinite data word (respectively, a finite data word) if and only if **M** is a positive instance of the recurrent state problem (respectively, the halting problem).

Let **M** be a two-counter machine with instructions set $\{\mathbf{I}_0,\ldots,\mathbf{I}_n\}$. In this chapter, for technical reasons, we always assume that $\mathbf{I}_0 \neq \mathbf{I}_n$ and $\mathbf{I}_n$ is the only halting instruction **halt**. We will write $\mathbf{I}_n$ simply **halt** for readability when no confusion can arise.

4.1 The satisfiability problem for MTL

In this section, we consider infinitary SAT and finitary SAT for MTL. We show that both of them are not decidable. Since every MTL-formula is equivalent to a TPTL^1-formula, we can know that this result also holds for TPTL^1.

Theorem 10. *For* MTL, *infinitary SAT is* Σ_1^1*-complete and finitary SAT is* Σ_1^0*-complete.*

Proof. Let **M** be a two-counter machine with instructions $\mathbf{I}_0,\ldots,\mathbf{I}_n$. Define a set of propositions $\mathbf{P} = \{\mathbf{I}_0,\ldots,\mathbf{I}_n,\mathbf{C}_1,\mathbf{C}_2\}$. First we show how to encode a computation of **M** into a data word over **P**. Let (J,c,d) be a configuration of **M**, where $J \in \{\mathbf{I}_0,\ldots,\mathbf{I}_n\}$ and $c,d \in \mathbb{N}$. We encode it by the data word $(J,0)(\mathbf{C}_1,c)(\mathbf{C}_2,d)$. In the encoding of the configuration we store the number 0 in the pair with an instruction proposition such that we can use it to test whether the data value stored in $\mathbf{C}_1$ or $\mathbf{C}_2$ is 0 or not. Let $\pi = (J_0,c_0,d_0)(J_1,c_1,d_1)\ldots$ be a computation of **M**, where $(J_0,c_0,d_0) = (\mathbf{I}_0,0,0)$, $J_i \in \{\mathbf{I}_0,\ldots,\mathbf{I}_n\}$ and $c_i,d_i \in \mathbb{N}\,(i \geq 1)$. We can encode π into the following data word:

$$(J_0,0)(\mathbf{C}_1,c_0)(\mathbf{C}_2,d_0)(J_1,0)(\mathbf{C}_1,c_1)(\mathbf{C}_2,d_1)\ldots.$$

In the following we define several MTL-formulas which express that a data word encodes a computation of **M** properly, where X^m is an abbreviation for m copies of the modality X.

(1) There is exactly one proposition from **P** that holds in each position:

$$\varphi_{\mathsf{prop}} := (\bigvee_{p\in\mathbf{P}} p) \wedge \bigwedge_{p\in\mathbf{P}} (p \rightarrow \bigwedge_{q\in\mathbf{P}\setminus\{p\}} \neg q).$$

(2) The sequence of all propositions in the data word is of the form $J_0,\mathbf{C}_1,\mathbf{C}_2,J_1,\mathbf{C}_1,\mathbf{C}_2,\ldots$:

$$\varphi_{\mathsf{seq}} := ((\bigvee_{0\leq i\leq n} \mathbf{I}_i) \rightarrow (\mathsf{X}\mathbf{C}_1 \wedge \mathsf{X}^2\mathbf{C}_2)) \wedge (\mathbf{C}_2 \rightarrow \mathsf{X}\bigvee_{0\leq i\leq n} \mathbf{I}_i).$$

(3) The initial configuration is $(\mathbf{I}_0, 0, 0)$:

$$\varphi_{\mathsf{init}} := \mathbf{I}_0 \wedge \mathsf{X}_{=0}(\mathbf{C}_1 \wedge \mathsf{X}_{=0}\,\mathbf{C}_2).$$

(4) For the halting instruction $\mathbf{I}_n$ (i.e., **halt**), define

$$\varphi_{\mathsf{halt}} := \mathbf{halt} \wedge \varphi_{\mathsf{prop}} \wedge \mathsf{X}(\mathbf{C}_1 \wedge \varphi_{\mathsf{prop}}) \wedge \mathsf{X}^2(\mathbf{C}_2 \wedge \varphi_{\mathsf{prop}}).$$

(5) For an increment instruction $\mathbf{I}_j$: $\mathbf{C}_1 := \mathbf{C}_1 + 1$; go to some $\mathbf{I}_k \in S_j$, where S_j is a nonempty subset of $\{\mathbf{I}_0, \ldots, \mathbf{I}_n\}$, define

$$\varphi_{I_j} := \mathbf{I}_j \rightarrow \bigvee_{\mathbf{I}_k \in S_j} [((\mathbf{C}_1 \vee \mathbf{C}_2)\,\mathsf{U}_{=0}\,\mathbf{I}_k) \wedge \mathsf{X}((\mathbf{C}_2 \vee \mathbf{I}_k)\,\mathsf{U}_{=1}\,\mathbf{C}_1) \wedge \mathsf{X}^2((\mathbf{I}_k \vee \mathbf{C}_1)\,\mathsf{U}_{=0}\,\mathbf{C}_2)].$$

If $\mathbf{I}_j$ operates on $\mathbf{C}_2$, then define

$$\varphi_{I_j} := \mathbf{I}_j \rightarrow \bigvee_{\mathbf{I}_k \in S_j} [((\mathbf{C}_1 \vee \mathbf{C}_2)\,\mathsf{U}_{=0}\,\mathbf{I}_k) \wedge \mathsf{X}((\mathbf{C}_2 \vee \mathbf{I}_k)\,\mathsf{U}_{=0}\,\mathbf{C}_1) \wedge \mathsf{X}^2((\mathbf{I}_k \vee \mathbf{C}_1)\,\mathsf{U}_{=1}\,\mathbf{C}_2)].$$

(6) For a decrement instruction $\mathbf{I}_j$: if $\mathbf{C}_1 = 0$ then go to some $\mathbf{I}_k \in S_j^1$ else $\mathbf{C}_1 := \mathbf{C}_1 - 1$; go to some $\mathbf{I}_m \in S_j^2$, where S_j^1 and S_j^2 are nonempty subsets of $\{\mathbf{I}_0, \ldots, \mathbf{I}_n\}$, define

$$\varphi_{I_j} := (\mathbf{I}_j \wedge \mathsf{X}_{=0}\,\mathbf{C}_1 \rightarrow \psi_{\mathsf{zero}}) \wedge (\mathbf{I}_j \wedge \mathsf{X}_{>0}\,\mathbf{C}_1 \rightarrow \psi_{\mathsf{notzero}}),$$

where

$$\psi_{\mathsf{zero}} := \bigvee_{\mathbf{I}_k \in S_j^1} [((\mathbf{C}_1 \vee \mathbf{C}_2)\mathsf{U}_{=0}\,\mathbf{I}_k) \wedge \mathsf{X}((\mathbf{C}_2 \vee \mathbf{I}_k)\mathsf{U}_{=0}\,\mathbf{C}_1) \wedge \mathsf{X}^2((\mathbf{I}_k \vee \mathbf{C}_1)\mathsf{U}_{=0}\,\mathbf{C}_2)],$$

$$\psi_{\mathsf{notzero}} := \bigvee_{\mathbf{I}_m \in S_j^2} [((\mathbf{C}_1 \vee \mathbf{C}_2)\mathsf{U}_{=0}\,\mathbf{I}_m) \wedge \mathsf{X}((\mathbf{C}_2 \vee \mathbf{I}_m)\mathsf{U}_{=-1}\,\mathbf{C}_1) \wedge \mathsf{X}^2((\mathbf{I}_m \vee \mathbf{C}_1)\mathsf{U}_{=0}\,\mathbf{C}_2)].$$

If $\mathbf{I}_j$ operates on $\mathbf{C}_2$, then define

$$\varphi_{I_j} := (\mathbf{I}_j \wedge (\mathbf{C}_1\,\mathsf{U}_{=0}\,\mathbf{C}_2) \rightarrow \psi_{\mathsf{zero}}) \wedge (\mathbf{I}_j \wedge (\mathbf{C}_1\,\mathsf{U}_{>0}\,\mathbf{C}_2) \rightarrow \psi'_{\mathsf{notzero}}),$$

where

$$\psi'_{\mathsf{notzero}} := \bigvee_{\mathbf{I}_m \in S_j^2} [((\mathbf{C}_1 \vee \mathbf{C}_2)\mathsf{U}_{=0}\,\mathbf{I}_m) \wedge \mathsf{X}((\mathbf{C}_2 \vee \mathbf{I}_m)\mathsf{U}_{=0}\,\mathbf{C}_1) \wedge \mathsf{X}^2((\mathbf{I}_m \vee \mathbf{C}_1)\mathsf{U}_{=-1}\,\mathbf{C}_2)].$$

We define two formulas φ_{infin} and φ_{fin} in the following such that φ_{infin} (respectively, φ_{fin}) is satisfiable if and only if $\mathbf{M}$ has a recurring computation (respectively, a halting computation).

$$\begin{aligned}\varphi_{\mathsf{infin}} :=& \varphi_{\mathsf{init}} \wedge \varphi_{\mathsf{prop}} \wedge \varphi_{I_0} \wedge \mathsf{GF}\,\mathbf{I}_0 \wedge \\ & \mathsf{G}\,(\varphi_{\mathsf{prop}} \wedge \varphi_{\mathsf{seq}} \wedge \neg\mathbf{halt} \wedge \bigwedge_{0\leq j<n} \varphi_{I_j}),\end{aligned}$$

and

$$\begin{aligned}\varphi_{\mathsf{fin}} :=& \varphi_{\mathsf{init}} \wedge \varphi_{\mathsf{prop}} \wedge \varphi_{I_0} \wedge \mathsf{F}\,\mathbf{halt} \wedge \\ & (\varphi_{\mathsf{prop}} \wedge \varphi_{\mathsf{seq}} \wedge \neg\mathbf{halt} \wedge \bigwedge_{0\leq j<n} \varphi_{I_j})\,\mathsf{U}\,\varphi_{\mathsf{halt}}.\end{aligned}$$

It is easily seen that if $\mathbf{M}$ is a positive instance of the recurrent state problem (respectively, the halting problem), then there is an infinite data word w (respectively, a finite data word w') which is the encoding of a recurring computation of $\mathbf{M}$ (respectively, a halting computation of $\mathbf{M}$) such that $w \models \varphi_{\mathsf{infin}}$ (respectively, $w' \models \varphi_{\mathsf{fin}}$). Conversely, if φ_{infin} is satisfiable over infinite data words, then there is an infinite data word that encodes an infinite computation of $\mathbf{M}$, which visits the instruction $\mathbf{I}_0$ infinitely often by the formula $\mathsf{GF}\,\mathbf{I}_0$. Similarly, if φ_{fin} is satisfiable over finite data words, then there is a finite data word that encodes a finite computation of $\mathbf{M}$ which reaches the instruction **halt**. □

4.2 SAT for the positive fragments of MTL and TPTL

In this section, we consider the satisfiability problem for the positive fragments of MTL and TPTL, in which the negation operator $\neg$ is only applied to propositions or atomic constraints. We show that a positive formula is satisfiable if and only if it is satisfied by a finite data word. This means that finitary SAT and infinitary SAT coincide for positive formulas. First we give the definitions for positive MTL and positive TPTL in the following.

Definition 5. The set of positive MTL-formulas (posMTL) is built by the following grammar:

$$\varphi ::= \top \mid \bot \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \mathsf{U}_I \varphi$$

The set of positive TPTL-formulas (posTPTL) is built by the following grammar:

$$\varphi ::= \top \mid \bot \mid p \mid \neg p \mid x \sim c \mid \neg x \sim c \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \mathsf{U} \varphi \mid x.\varphi$$

In the following, we show that a posTPTL-formula is satisfiable if and only if it is satisfied by a finite data word. First we prove two lemmas.

Lemma 8. *Let u be a finite data word, let i be a position in u and let ν be a register valuation. Then for every* posTPTL*-formula φ and every data word w, if $(u,i,\nu) \models \varphi$, then $(uw,i,\nu) \models \varphi$.*

Proof. We prove the lemma by induction on φ. The proof for the cases that φ is $\top, \bot, p, \neg p, x \sim c$ or $\neg x \sim c$ is easy.

- If φ is $\varphi_1 \wedge \varphi_2$, then $(u,i,\nu) \models \varphi_1 \wedge \varphi_2$ if and only if $(u,i,\nu) \models \varphi_1$ and $(u,i,\nu) \models \varphi_2$. By induction hypothesis, we have $(uw,i,\nu) \models \varphi_1$ and $(uw,i,\nu) \models \varphi_2$. This implies $(uw,i,\nu) \models \varphi_1 \wedge \varphi_2$.
- If φ is $\varphi_1 \vee \varphi_2$, then $(u,i,\nu) \models \varphi_1 \vee \varphi_2$ if and only if $(u,i,\nu) \models \varphi_1$ or $(u,i,\nu) \models \varphi_2$. By induction hypothesis, we have $(uw,i,\nu) \models \varphi_1$ or $(uw,i,\nu) \models \varphi_2$. This implies $(uw,i,\nu) \models \varphi_1 \vee \varphi_2$.
- If φ is $x.\varphi_1$, then $(u,i,\nu) \models x.\varphi_1$ if and only if $(u,i,\nu[x \mapsto d_i]) \models \varphi_1$. By induction hypothesis, we have $(uw,i,\nu[x \mapsto d_i]) \models \varphi_1$. This implies $(uw,i,\nu) \models x.\varphi_1$.
- If φ is $\varphi_1 \mathsf{U} \varphi_2$, then $(u,i,\nu) \models \varphi_1 \mathsf{U} \varphi_2$ if and only if there is a position $i < j < |u|$ such that $(u,j,\nu) \models \varphi_2$ and for all $i < t < j$, $(u,t,\nu) \models \varphi_1$. By induction hypothesis, we have $(uw,j,\nu) \models \varphi_2$ and for all $i < t < j$, $(uw,t,\nu) \models \varphi_1$. This implies $(uw,i,\nu) \models \varphi_1 \mathsf{U} \varphi_2$.

□

Lemma 9. *Let w be a data word, let i be a position in w and let ν be a register valuation. Then for every* posTPTL*-formula φ, if $(w,i,\nu) \models \varphi$, then there exists a position $j \geq i$ in w such that $(w[0:j],i,\nu) \models \varphi$.*

Proof. We prove the lemma by induction on φ.

- If φ is $\top, \bot, p, \neg p, x \sim c$ or $\neg x \sim c$, then $(w,i,\nu) \models \varphi$ if and only if $(w[0:i],i,\nu) \models \varphi$.
- If φ is $\varphi_1 \wedge \varphi_2$, then $(w,i,\nu) \models \varphi_1 \wedge \varphi_2$ if and only if $(w,i,\nu) \models \varphi_1$ and $(w,i,\nu) \models \varphi_2$. By induction hypothesis, there exist two positions $j_1 \geq i$ and $j_2 \geq i$ in w such that $(w[0:j_1],i,\nu) \models \varphi_1$ and $(w[0:j_2],i,\nu) \models \varphi_2$. Let $j = \max\{j_1, j_2\}$. By Lemma 8, we have $(w[0:j],i,\nu) \models \varphi_1$ and $(w[0:j],i,\nu) \models \varphi_2$. This implies $(w[0:j],i,\nu) \models \varphi_1 \wedge \varphi_2$.
- If φ is $\varphi_1 \vee \varphi_2$, then $(w,i,\nu) \models \varphi_1 \vee \varphi_2$ if and only if $(w,i,\nu) \models \varphi_1$ or $(w,i,\nu) \models \varphi_2$. By induction hypothesis, there exist two positions $j_1 \geq i$ and $j_2 \geq i$ in w such that $(w[0:j_1],i,\nu) \models \varphi_1$ or $(w[0:j_2],i,\nu) \models \varphi_2$. Let $j = \max\{j_1, j_2\}$. By Lemma 8, we have $(w[0:j],i,\nu) \models \varphi_1$ or $(w[0:j],i,\nu) \models \varphi_2$. This implies $(w[0:j],i,\nu) \models \varphi_1 \vee \varphi_2$.

- If φ is $x.\varphi_1$, then $(w,i,v) \models x.\varphi_1$ if and only if $(w,i,v[x \mapsto d_i]) \models \varphi_1$. By induction hypothesis, there exists a position $j \geq i$ in w such that $(w[0:j],i,v[x \mapsto d_i]) \models \varphi_1$. This implies $(w[0:j],i,v) \models x.\varphi_1$.

- If φ is $\varphi_1 \mathsf{U} \varphi_2$, then $(w,i,v) \models \varphi_1 \mathsf{U} \varphi_2$ if and only if there is a position $i < i' < |w|$ such that $(w,i',v) \models \varphi_2$ and for all $i < t < i'$, $(w,t,v) \models \varphi_1$. Suppose $i' = i+n$, where $n \geq 1$. By inductive hypothesis, there exist positions $j_s \geq (i+s)$ in w, where $1 \leq s \leq n$, such that $(w[0:j_n],i+n,v) \models \varphi_2$ and for all $1 \leq s < n$, $(w[0:j_s],i+s,v) \models \varphi_1$. Let $j = \max\{j_1,\ldots,j_n\}$. By Lemma 8, we have $(w[0:j],i+n,v) \models \varphi_2$ and for all $1 \leq s < n$, $(w[0:j],i+s,v) \models \varphi_1$. This implies $(w[0:j],i,v) \models \varphi_1 \mathsf{U} \varphi_2$.

□

Theorem 11. (Finite Model Property for positive TPTL)
For every posTPTL*-formula φ, if φ is satisfiable, then it is satisfied by a finite data word.*

Proof. Let φ be a posTPTL-formula. Suppose that there exists a data word w such that $w \models \varphi$. By Lemma 9, we can know that there exists a position j in w such that $w[0:j] \models \varphi$. Obviously, $w[0:j]$ is a finite data word. □

Since every posMTL-formula is equivalent to a posTPTL-formula, we can get the following corollary.

Corollary 5. *For every* posMTL*-formula φ, if φ is satisfiable, then it is satisfied by a finite data word.*

Theorem 12. *For* posMTL *and* posTPTL*, finitary SAT and infinitary SAT coincide, and both of them are Σ_1^0-complete.*

Proof. It is easily seen that finitary SAT and infinitary SAT coincide for posMTL and posTPTL by Lemma 8 and Theorem 11.

We show that finitary SAT is Σ_1^0-complete for posMTL by a reduction from the halting problem for two-counter machines. Note that the formula φ_{fin} constructed in the proof of Theorem 10 is in positive form except the formulas φ_{I_j} for decrement instructions $\mathbf{I}_j$ in it. We can construct an equivalent posMTL-formula φ'_{fin} by replacing φ_{I_j} with equivalent formulas. Let $\mathbf{I}_j$ be a decrement instruction. If $\mathbf{I}_j$ operates on $\mathbf{C}_1$, then we can replace φ_{I_j} constructed in (6) in the proof of Theorem 10 with the following posMTL-formula

$$\varphi'_{I_j} := (\neg \mathbf{I}_j \vee \mathsf{X}_{>0}\, \mathbf{C}_1 \vee \psi_{\mathsf{zero}}) \wedge (\neg \mathbf{I}_j \vee \mathsf{X}_{=0}\, \mathbf{C}_1 \vee \psi_{\mathsf{notzero}}).$$

If $\mathbf{I}_j$ operates on $\mathbf{C}_2$, then we can replace φ_{I_j} with

$$\varphi'_{I_j} := (\neg \mathbf{I}_j \vee (\mathbf{C}_1 \,\mathsf{U}_{>0}\, \mathbf{C}_2) \vee \psi_{\mathsf{zero}}) \wedge (\neg \mathbf{I}_j \vee (\mathbf{C}_1 \,\mathsf{U}_{=0}\, \mathbf{C}_2) \vee \psi'_{\mathsf{notzero}}).$$

Clearly, φ'_{fin} is equivalent to a posTPTL-formula, so finitary SAT for posTPTL is also Σ^0_1-complete. □

4.3 SAT for the unary fragments of MTL and TPTL

In this section, we consider the satisfiability problem for the unary fragments of MTL and TPTL, in which only the modalities F and X are allowed to use. First we give the definitions for unaMTL and unaTPTL in the following.

Definition 6. The set of unary MTL-formulas (unaMTL) is built by the following grammar:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}_I\, \varphi \mid \mathsf{F}_I\, \varphi$$

The set of unary TPTL-formulas (unaTPTL) is built by the following grammar:

$$\varphi ::= \top \mid p \mid x \sim c \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \mathsf{F}\varphi \mid x.\varphi$$

In [28], non-primitive recursive complexity for finitary SAT for unary freezeLTL^1 is proved. This result was strengthened to SAT for unary freezeLTL^1 without the X modality [40]. Unfortunately, if we extend freezeLTL^1 to TPTL^1, we can obtain undecidability for the satisfiability problem, and this still holds even for TPTL^1 without the X modality. We also prove undecidability of SAT for unaMTL, however, it is an open problem whether undecidability also holds for the unaMTL fragment in which the X modality is not allowed.

Theorem 13. *For* unaMTL, *infinitary SAT is* Σ^1_1*-complete and finitary SAT is* Σ^0_1*-complete. For* unaTPTL^1*, this is even the case if we do not allow for the* X *modality.*

Proof. Let $\mathbf{M}$ be a two-counter machine with instructions $\mathbf{I}_0, \ldots, \mathbf{I}_n$. Define a set of propositions $\mathbf{P} = \{\mathbf{I}_0, \ldots, \mathbf{I}_n, \mathbf{C}_1, \mathbf{C}_2\}$. First we show how to encode the computation of $\mathbf{M}$ into a data word over $\mathbf{P}$. Let $\pi = (J_0, c_0, d_0)(J_1, c_1, d_1)\ldots$ be a computation of $\mathbf{M}$, where $(J_0, c_0, d_0) = (\mathbf{I}_0, 0, 0)$, $J_i \in \{\mathbf{I}_0, \ldots, \mathbf{I}_n\}$ and $c_i, d_i \in \mathbb{N}\,(i \geq 1)$. We can encode π as follows:

$$(J_0, 0)(\mathbf{C}_1, c_0)(\mathbf{C}_2, d_0)(J_1, 1)(\mathbf{C}_1, c_1 + 1)(\mathbf{C}_2, d_1 + 1)\ldots,$$

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

$$(J_i, i)(\mathbf{C}_1, c_i + i)(\mathbf{C}_2, d_i + i).$$

The data values in the positions where instruction propositions J_i hold are strictly monotonic and increase progressively by exactly 1. We can use these numbers for the zero test operation. By this encoding we can know that for any two consecutive configurations in the computation, the subdata word

$$(J, i)(\mathbf{C}_1, n_{1,1})(\mathbf{C}_2, n_{2,1})(J', i+1)(\mathbf{C}_1, n_{1,2})(\mathbf{C}_2, n_{2,2}),$$

which is the encoding of them in the data word that encodes the whole computation, satisfies the following conditions, where j is 1 or 2:

- If J sets $\mathbf{C}_j := \mathbf{C}_j + 1$, then $n_{j,2} = n_{j,1} + 2$.
- If J sets $\mathbf{C}_j := \mathbf{C}_j - 1$, then $n_{j,2} = n_{j,1}$.
- If $\mathbf{C}_j$ does not change, then $n_{j,2} = n_{j,1} + 1$.

Hence, the data values in the positions where $\mathbf{C}_1$ (respectively, $\mathbf{C}_2$) holds are also monotonic. We can exploit this monotonicity property to get rid of the U modality, and also the X modality for $\mathsf{unaTPTL}^1$.

In the following we define several unaMTL-formulas (respectively, $\mathsf{unaTPTL}^1$-formulas without the X modality) which express that a data word encodes a computation of $\mathbf{M}$ properly. It is easily seen that every unaMTL-formula without the X modality can effectively be translated into an equivalent $\mathsf{unaTPTL}^1$-formula without the X modality. So we do not define the $\mathsf{unaTPTL}^1$-formulas explicitly where we have these unaMTL-formulas below. But for the unaMTL-formulas that use the X modality, we will give the equivalent $\mathsf{unaTPTL}^1$-formulas without the X modality.

(1) There is exactly one proposition from $\mathbf{P}$ that holds in every position:

$$\varphi_{\mathsf{prop}} := (\bigvee_{p \in \mathbf{P}} p) \wedge \bigwedge_{p \in \mathbf{P}} (p \rightarrow \bigwedge_{q \in \mathbf{P} \setminus \{p\}} \neg q).$$

(2) The data values in the positions where $\mathbf{I}_i \, (0 \leq i \leq n)$ holds are strictly monotonic and increase progressively by exactly 1. Define φ_{dat} to be a conjunction of the following two formulas.

- The data values are strictly monotonic:

$$(\bigvee_{0\le i\le n} \mathbf{I}_i) \to \neg \mathsf{F}_{\le 0} \bigvee_{0\le i\le n} \mathbf{I}_i.$$

- The data values increase by 1:

$$(\bigvee_{0\le i< n} \mathbf{I}_i) \to \mathsf{F}_{=1} \bigvee_{0\le i\le n} \mathbf{I}_i.$$

(3) The sequence of all propositions in the data word is of the form $J_0, \mathbf{C}_1, \mathbf{C}_2, J_1, \mathbf{C}_1, \mathbf{C}_2, \ldots$.

- For unaMTL, define

$$\varphi_{\mathsf{seq}} := ((\bigvee_{0\le i\le n} \mathbf{I}_i) \to (\mathsf{X}\mathbf{C}_1 \wedge \mathsf{X}^2\mathbf{C}_2)) \wedge (\mathbf{C}_2 \to \mathsf{X} \bigvee_{0\le i\le n} \mathbf{I}_i).$$

- For $\mathsf{unaTPTL}^1$ without the X modality, define φ'_{seq} to be a conjunction of the following two formulas.
  - There are two propositions $\mathbf{C}_1$ and $\mathbf{C}_2$ between two consecutive instruction propositions:

$$(\bigvee_{0\le i\le n} \mathbf{I}_i) \to x.\mathsf{F}[\mathbf{C}_1 \wedge \mathsf{F}(\mathbf{C}_2 \wedge \mathsf{F}(x = 1 \wedge \bigvee_{0\le i\le n} \mathbf{I}_i))].$$

  - There is only one proposition $\mathbf{C}_1$ (respectively, $\mathbf{C}_2$) between two consecutive instruction propositions:

$$\begin{aligned}&(\bigvee_{0\le i\le n} \mathbf{I}_i) \to \neg x.\mathsf{F}[\mathbf{C}_1 \wedge \mathsf{F}(\mathbf{C}_1 \wedge \mathsf{F}(x = 1 \wedge \bigvee_{0\le i\le n} \mathbf{I}_i))]\\ \wedge&(\bigvee_{0\le i\le n} \mathbf{I}_i) \to \neg x.\mathsf{F}[\mathbf{C}_2 \wedge \mathsf{F}(\mathbf{C}_2 \wedge \mathsf{F}(x = 1 \wedge \bigvee_{0\le i\le n} \mathbf{I}_i))].\end{aligned}$$

(4) The data values in the positions where $\mathbf{C}_1$ (respectively, $\mathbf{C}_2$) holds are monotonic, and are no less than the previous data values in the positions where instruction propositions hold:

$$\begin{aligned}\varphi_{\mathsf{cntdat}} :=&(\mathbf{C}_1 \to \neg \mathsf{F}_{<0}\mathbf{C}_1) \wedge (\mathbf{C}_2 \to \neg \mathsf{F}_{<0}\mathbf{C}_2)\\ &\wedge (\bigvee_{0\le i\le n} \mathbf{I}_i) \to \neg(\mathsf{F}_{<0}\mathbf{C}_1 \vee \mathsf{F}_{<0}\mathbf{C}_2).\end{aligned}$$

(5) The initial configuration is $(\mathbf{I}_0, 0, 0)$.

- For unaMTL, define

$$\varphi_{\mathsf{init}} := \mathbf{I}_0 \wedge \mathsf{X}_{=0}(\mathbf{C}_1 \wedge \mathsf{X}_{=0}\,\mathbf{C}_2).$$

- For unaTPTL^1 without the X modality, define

$$\varphi'_{\mathsf{init}} := \mathbf{I}_0 \wedge x.\mathsf{F}[\mathbf{C}_1 \wedge x = 0 \wedge \mathsf{F}(\mathbf{C}_2 \wedge x = 0 \wedge \mathsf{F}(x = 1 \wedge \bigvee_{0 \le i \le n} \mathbf{I}_i))].$$

(6) For the halting instruction $\mathbf{I}_n$ (i.e., **halt**), define

$$\begin{aligned}\varphi_{\mathsf{halt}} :=& \mathbf{halt} \wedge \varphi_{\mathsf{prop}} \wedge \mathsf{G}(\varphi_{\mathsf{prop}} \wedge \neg \bigvee_{0 \le i \le n} \mathbf{I}_i) \wedge \\ & \mathsf{F}(\mathbf{C}_1 \wedge \mathsf{F}\,\mathbf{C}_2) \wedge \mathsf{G}((\mathbf{C}_1 \rightarrow \neg \mathsf{F}\,\mathbf{C}_1) \wedge (\mathbf{C}_2 \rightarrow \neg \mathsf{F}\,\top)).\end{aligned}$$

This formula guarantees that the data word ends with $(\mathbf{halt}, n_1)(\mathbf{C}_1, n_2)(\mathbf{C}_2, n_3)$, where $n_1, n_2, n_3 \in \mathbb{N}$.

(7) $\mathbf{I}_j$ is an increment instruction: $\mathbf{C}_1 := \mathbf{C}_1 + 1$; go to some $\mathbf{I}_k \in S_j$.

- For unaMTL, define

$$\varphi_{I_j} := \mathbf{I}_j \rightarrow [(\mathsf{F}_{=1} \bigvee_{\mathbf{I}_k \in S_j} \mathbf{I}_k) \wedge \mathsf{X}(\neg \mathsf{F}_{<2}\,\mathbf{C}_1 \wedge \mathsf{F}_{=2}\,\mathbf{C}_1) \wedge \mathsf{X}^2(\neg \mathsf{F}_{<1}\,\mathbf{C}_2 \wedge \mathsf{F}_{=1}\,\mathbf{C}_2)].$$

Note that incrementing the counter $\mathbf{C}_1$ by 1 corresponds to incrementing the data value by exactly 2 in the encoding. The value of counter $\mathbf{C}_2$ does not change, and this corresponds to incrementing the data value by exactly 1.

If $\mathbf{I}_j$ operates on $\mathbf{C}_2$, then define

$$\varphi_{I_j} := \mathbf{I}_j \rightarrow [(\mathsf{F}_{=1} \bigvee_{\mathbf{I}_k \in S_j} \mathbf{I}_k) \wedge \mathsf{X}(\neg \mathsf{F}_{<1}\,\mathbf{C}_1 \wedge \mathsf{F}_{=1}\,\mathbf{C}_1) \wedge \mathsf{X}^2(\neg \mathsf{F}_{<2}\,\mathbf{C}_2 \wedge \mathsf{F}_{=2}\,\mathbf{C}_2)].$$

- For unaTPTL^1 without the X modality, define

$$\varphi'_{I_j} := \mathbf{I}_j \rightarrow \bigvee_{\mathbf{I}_k \in S_j} (\varphi_{\mathbf{C}_1} \wedge \varphi_{\mathbf{C}_2}),$$

where

$$\varphi_{\mathbf{C}_1} := x.\mathsf{F}[\mathbf{C}_1 \wedge \mathsf{F}(x = 1 \wedge \mathbf{I}_k) \wedge x.\mathsf{G}(\mathbf{C}_1 \rightarrow x \geq 2) \wedge x.\mathsf{F}(\mathbf{C}_1 \wedge x = 2)],$$
$$\varphi_{\mathbf{C}_2} := x.\mathsf{F}[\mathbf{C}_2 \wedge \mathsf{F}(x = 1 \wedge \mathbf{I}_k) \wedge x.\mathsf{G}(\mathbf{C}_2 \rightarrow x \geq 1) \wedge x.\mathsf{F}(\mathbf{C}_2 \wedge x = 1)].$$

Note that the formula $\mathsf{F}(x = 1 \wedge \mathbf{I}_k)$ in $\varphi_{\mathbf{C}_j}\,(j \in \{1,2\})$ guarantees that the $\mathbf{C}_j$ in consideration is exactly the one between $\mathbf{I}_j$ and $\mathbf{I}_k$. Hence we can get rid of the X modality.

If $\mathbf{I}_j$ operates on $\mathbf{C}_2$, then define

$$\varphi'_{I_j} := \mathbf{I}_j \rightarrow \bigvee_{\mathbf{I}_k \in S_j} (\varphi'_{\mathbf{C}_1} \wedge \varphi'_{\mathbf{C}_2}),$$

where

$$\varphi'_{\mathbf{C}_1} := x.\mathsf{F}[\mathbf{C}_1 \wedge \mathsf{F}(x = 1 \wedge \mathbf{I}_k) \wedge x.\mathsf{G}(\mathbf{C}_1 \rightarrow x \geq 1) \wedge x.\mathsf{F}(\mathbf{C}_1 \wedge x = 1)],$$
$$\varphi'_{\mathbf{C}_2} := x.\mathsf{F}[\mathbf{C}_2 \wedge \mathsf{F}(x = 1 \wedge \mathbf{I}_k) \wedge x.\mathsf{G}(\mathbf{C}_2 \rightarrow x \geq 2) \wedge x.\mathsf{F}(\mathbf{C}_2 \wedge x = 2)].$$

(8) $\mathbf{I}_j$ is a decrement instruction: if $\mathbf{C}_1 = 0$ then go to some $\mathbf{I}_k \in S_j^1$ else $\mathbf{C}_1 := \mathbf{C}_1 - 1$; go to some $\mathbf{I}_m \in S_j^2$.

- For unaMTL, define

$$\varphi_{I_j} := (\mathbf{I}_j \wedge \mathsf{F}_{=0}\,\mathbf{C}_1 \rightarrow \psi_{\mathsf{zero}}) \wedge (\mathbf{I}_j \wedge \neg\mathsf{F}_{=0}\,\mathbf{C}_1 \rightarrow \psi_{\mathsf{notzero}}),$$

where

$$\psi_{\mathsf{zero}} := \mathsf{X}(\mathbf{C}_1 \wedge \neg\mathsf{F}_{<1}\,\mathbf{C}_1 \wedge \mathsf{F}_{=1}\,\mathbf{C}_1) \wedge \mathsf{X}^2(\mathbf{C}_2 \wedge \neg\mathsf{F}_{<1}\,\mathbf{C}_2 \wedge \mathsf{F}_{=1}\,\mathbf{C}_2) \wedge \mathsf{F}_{=1} \bigvee_{\mathbf{I}_k \in S_j^1} \mathbf{I}_k,$$

and

$$\psi_{\mathsf{notzero}} := \mathsf{X}(\mathbf{C}_1 \wedge \mathsf{F}_{=0}\,\mathbf{C}_1) \wedge \mathsf{X}^2(\mathbf{C}_2 \wedge \neg\mathsf{F}_{<1}\,\mathbf{C}_2 \wedge \mathsf{F}_{=1}\,\mathbf{C}_2) \wedge \mathsf{F}_{=1} \bigvee_{\mathbf{I}_m \in S_j^2} \mathbf{I}_m.$$

Note that decrementing the value of $\mathbf{C}_1$ corresponds to not changing the data value in the encoding.

If $\mathbf{I}_j$ operates on $\mathbf{C}_2$, then define

$$\varphi_{I_j} := (\mathbf{I}_j \wedge \mathsf{F}_{=0}\,\mathbf{C}_2 \rightarrow \psi_{\mathsf{zero}}) \wedge (\mathbf{I}_j \wedge \neg\mathsf{F}_{=0}\,\mathbf{C}_2 \rightarrow \psi'_{\mathsf{notzero}}),$$

where

$$\psi'_{\mathsf{notzero}} := \mathsf{X}(\mathbf{C}_1 \wedge \neg\mathsf{F}_{<1}\,\mathbf{C}_1 \wedge \mathsf{F}_{=1}\,\mathbf{C}_1) \wedge \mathsf{X}^2(\mathbf{C}_2 \wedge \mathsf{F}_{=0}\,\mathbf{C}_2) \wedge \mathsf{F}_{=1} \bigvee_{\mathbf{I}_m \in S_j^2} \mathbf{I}_m.$$

- For $\mathsf{unaTPTL}^1$ without the X modality, define

$$\varphi'_{I_j} := (\mathbf{I}_j \wedge x.\mathsf{F}(x = 0 \wedge \mathbf{C}_1) \rightarrow \phi_{\mathsf{zero}}) \wedge (\mathbf{I}_j \wedge \neg x.\mathsf{F}(x = 0 \wedge \mathbf{C}_1) \rightarrow \phi_{\mathsf{notzero}}),$$

where

$$\phi_{\mathsf{zero}} := \bigvee_{\mathbf{I}_k \in S_j^1} \langle x.\mathsf{F}[\mathbf{C}_1 \wedge \mathsf{F}(x = 1 \wedge \mathbf{I}_k) \wedge x.\mathsf{G}(\mathbf{C}_1 \rightarrow x \geq 1) \wedge x.\mathsf{F}(\mathbf{C}_1 \wedge x = 1)]$$
$$\wedge x.\mathsf{F}[\mathbf{C}_2 \wedge \mathsf{F}(x = 1 \wedge \mathbf{I}_k) \wedge x.\mathsf{G}(\mathbf{C}_2 \rightarrow x \geq 1) \wedge x.\mathsf{F}(\mathbf{C}_2 \wedge x = 1)]\rangle,$$

$$\phi_{\mathsf{notzero}} := \bigvee_{\mathbf{I}_m \in S_j^2} \langle x.\mathsf{F}[\mathbf{C}_1 \wedge \mathsf{F}(x = 1 \wedge \mathbf{I}_m) \wedge x.\mathsf{F}(\mathbf{C}_1 \wedge x = 0)]$$
$$\wedge x.\mathsf{F}[\mathbf{C}_2 \wedge \mathsf{F}(x = 1 \wedge \mathbf{I}_m) \wedge x.\mathsf{G}(\mathbf{C}_2 \rightarrow x \geq 1) \wedge x.\mathsf{F}(\mathbf{C}_2 \wedge x = 1)]\rangle.$$

If $\mathbf{I}_j$ operates on $\mathbf{C}_2$, then define

$$\varphi'_{I_j} := (\mathbf{I}_j \wedge x.\mathsf{F}(x = 0 \wedge \mathbf{C}_2) \rightarrow \phi_{\mathsf{zero}}) \wedge (\mathbf{I}_j \wedge \neg x.\mathsf{F}(x = 0 \wedge \mathbf{C}_2) \rightarrow \phi'_{\mathsf{notzero}}),$$

where

$$\phi'_{\mathsf{notzero}} := \bigvee_{\mathbf{I}_m \in S_j^2} \langle x.\mathsf{F}[\mathbf{C}_2 \wedge \mathsf{F}(x = 1 \wedge \mathbf{I}_m) \wedge x.\mathsf{F}(\mathbf{C}_2 \wedge x = 0)]$$
$$\wedge x.\mathsf{F}[\mathbf{C}_1 \wedge \mathsf{F}(x = 1 \wedge \mathbf{I}_m) \wedge x.\mathsf{G}(\mathbf{C}_1 \rightarrow x \geq 1) \wedge x.\mathsf{F}(\mathbf{C}_1 \wedge x = 1)]\rangle.$$

In the following we define two formulas φ_{infin} and φ_{fin} (respectively, $\varphi'_{\mathsf{infin}}$ and φ'_{fin}) for unaMTL (respectively, $\mathsf{unaTPTL}^1$ without the X modality) that can capture the infinite computation of $\mathbf{M}$ that visits the initial instruction $\mathbf{I}_0$ infinitely often and the finite computation of $\mathbf{M}$ that reaches the halting instruction **halt**, respectively.

For unaMTL, define

$$\varphi_{\mathsf{infin}} := \varphi_{\mathsf{init}} \wedge \varphi_{\mathsf{prop}} \wedge \varphi_{I_0} \wedge \varphi_{\mathsf{dat}} \wedge \varphi_{\mathsf{cntdat}} \wedge \mathsf{GF}\,\mathbf{I}_0 \wedge \mathsf{G}\,(\varphi_{\mathsf{prop}} \wedge \varphi_{\mathsf{seq}} \wedge \varphi_{\mathsf{dat}} \wedge \varphi_{\mathsf{cntdat}} \wedge \neg\mathbf{halt} \wedge \bigwedge_{0 \le j < n} \varphi_{I_j}),$$

and

$$\varphi_{\mathsf{fin}} := \varphi_{\mathsf{init}} \wedge \varphi_{\mathsf{prop}} \wedge \varphi_{I_0} \wedge \varphi_{\mathsf{dat}} \wedge \varphi_{\mathsf{cntdat}} \wedge \mathsf{F}\,\varphi_{\mathsf{halt}} \wedge \mathsf{G}\,(\mathsf{F}\,\varphi_{\mathsf{halt}} \rightarrow \varphi_{\mathsf{prop}} \wedge \varphi_{\mathsf{seq}} \wedge \varphi_{\mathsf{dat}} \wedge \varphi_{\mathsf{cntdat}} \wedge \bigwedge_{0 \le j < n} \varphi_{I_j}).$$

For unaTPTL[1] without the X modality, define

$$\varphi'_{\mathsf{infin}} := \varphi'_{\mathsf{init}} \wedge \varphi_{\mathsf{prop}} \wedge \varphi'_{\mathsf{seq}} \wedge \varphi'_{I_0} \wedge \varphi_{\mathsf{dat}} \wedge \varphi_{\mathsf{cntdat}} \wedge \mathsf{GF}\,\mathbf{I}_0 \wedge \mathsf{G}\,(\varphi_{\mathsf{prop}} \wedge \varphi'_{\mathsf{seq}} \wedge \varphi_{\mathsf{dat}} \wedge \varphi_{\mathsf{cntdat}} \wedge \neg\mathbf{halt} \wedge \bigwedge_{0 \le j < n} \varphi'_{I_j}),$$

and

$$\varphi'_{\mathsf{fin}} := \varphi'_{\mathsf{init}} \wedge \varphi_{\mathsf{prop}} \wedge \varphi'_{\mathsf{seq}} \wedge \varphi'_{I_0} \wedge \varphi_{\mathsf{dat}} \wedge \varphi_{\mathsf{cntdat}} \wedge \mathsf{F}\,\varphi_{\mathsf{halt}} \wedge \mathsf{G}\,(\mathsf{F}\,\varphi_{\mathsf{halt}} \rightarrow \varphi_{\mathsf{prop}} \wedge \varphi'_{\mathsf{seq}} \wedge \varphi_{\mathsf{dat}} \wedge \varphi_{\mathsf{cntdat}} \wedge \bigwedge_{0 \le j < n} \varphi'_{I_j}).$$

We see at once that φ_{infin} and $\varphi'_{\mathsf{infin}}$ (respectively, φ_{fin} and φ'_{fin}) are satisfiable if and only if **M** has a recurring computation (respectively, a halting computation). □

Corollary 6. *For* unaTPTL, *infinitary SAT is* Σ^1_1*-complete and finitary SAT is* Σ^0_1*-complete.*

4.4 SAT for the pure fragment of MTL

In Theorem 5 of [7], the authors proved that infinitary SAT is undecidable for the pure fragment of TPTL. In this section, we consider the satisfiability problem for the pure fragment of MTL. We show that both of infinitary SAT and finitary SAT are not decidable for the pure fragment of MTL. First we give the definition for pureMTL in the following.

Definition 7. The set of pure MTL-formulas (pureMTL) is built by the following grammar:

$$\varphi ::= \top \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \mathsf{U}_I \varphi$$

In the theorem blew, we show that the propositions are not necessary to get the undecidability. The satisfiability problem is still undecidable for MTL even without any propositions.

Theorem 14. *For* pureMTL, *infinitary SAT is* Σ_1^1*-complete and finitary SAT is* Σ_1^0*-complete.*

Proof. Let $\mathbf{M}$ be a two-counter machine with instructions $\mathbf{I}_0, \ldots, \mathbf{I}_n$. For each configuration $(\mathbf{I}_j, c, d)$ of $\mathbf{M}$, where $j \in \{0, \ldots, n\}$ and $c, d \in \mathbb{N}$, we encode it into the following pure data word (we put the numbers into pairs for readability of the proof)

$$\underbrace{(0,3)\overbrace{(0,1)\cdots(0,1)(0,2)(0,1)\cdots(0,1)}^{n+1}}_{\mathbf{I}_j}\underbrace{(0,4+c)}_{\mathbf{C}_1}\underbrace{(0,4+d)}_{\mathbf{C}_2}.$$

It starts with the pair $(0,3)$, and followed by $n+1$ pairs: one is $(0,2)$, and the remaining are $(0,1)$. The pair $(0,2)$ is the $(j+1)^{\text{th}}$ pair after the pair $(0,3)$. These $n+2$ pairs encode the instruction $\mathbf{I}_j$. After that, the pair $(0,4+c)$ encodes the value of $\mathbf{C}_1$ and the pair $(0,4+d)$ encodes the value of $\mathbf{C}_2$. The encoding of a computation of $\mathbf{M}$ is a sequence of encodings for each configuration of it.

In the following we define several pureMTL-formulas which express that a pure data word encodes a computation of $\mathbf{M}$ properly.

(1) The pair $(0,3)$ identifies an instruction (i.e., the beginning of the encoding of a configuration):

$$\varphi_{\mathsf{inst}} := \mathsf{X}_{=3}\mathsf{X}_{=-3}\top.$$

(2) For each instruction $\mathbf{I}_j$ $(j \in \{0, ..., n\})$, define

$$\psi_j := \mathsf{X}_{=3}\mathsf{X}_{=-3}(\mathsf{X}_{=1}\mathsf{X}_{=-1})^j\mathsf{X}_{=2}\mathsf{X}_{=-2}(\mathsf{X}_{=1}\mathsf{X}_{=-1})^{n-j}\mathsf{X}_{\geq 4}\mathsf{X}_{\leq 4}\mathsf{X}_{\geq 4}\top,$$

where $(\mathsf{X}_{=1}\mathsf{X}_{=-1})^k$ is an abbreviation for k copies of $\mathsf{X}_{=1}\mathsf{X}_{=-1}$.

(3) The data word is a concatenation of encodings of configurations:

$$\varphi_{\mathsf{struct}} := (\bigvee_{0\leq j\leq n} \psi_j) \to \mathsf{X}^{2n+8}(\bigvee_{0\leq j\leq n} \psi_j).$$

(4) The initial configuration is $(\mathbf{I}_0, 0, 0)$:

$$\varphi_{\mathsf{init}} := \psi_0 \wedge \mathsf{X}^{2+4n}(\mathsf{X}_{=4}\mathsf{X}_{=-4})^2\top.$$

(5) For the halting instruction $\mathbf{I}_n$ (i.e., **halt**), define

$$\varphi_{\mathsf{halt}} := \mathsf{X}_{=3}\mathsf{X}_{=-3}(\mathsf{X}_{=1}\mathsf{X}_{=-1})^n\mathsf{X}_{=2}\mathsf{X}_{=-2}\,\mathsf{X}_{\geq 4}\mathsf{X}_{\leq 4}\mathsf{X}_{\geq 4}\,(\neg\mathsf{X}\top).$$

(6) For an increment instruction I_j: $\mathbf{C}_1 := \mathbf{C}_1 + 1$; go to some $I_k \in S_j$, define

$$\varphi_{I_j} := \psi_j \rightarrow (\neg\varphi_{\mathsf{inst}}\,\mathsf{U}_{=0} \bigvee_{\mathbf{I}_k \in S_j} \psi_k) \wedge \varphi_{\mathbf{C}_1} \wedge \varphi_{\mathbf{C}_2},$$

where

$$\begin{aligned}\varphi_{\mathbf{C}_1} :=& \mathsf{X}^{2n+5}[(\neg\mathsf{X}^3\varphi_{\mathsf{inst}})\,\mathsf{U}_{=1}\,(\mathsf{X}^3\,\varphi_{\mathsf{inst}})],\\ \varphi_{\mathbf{C}_2} :=& \mathsf{X}^{2n+7}[(\neg\mathsf{X}\varphi_{\mathsf{inst}})\,\mathsf{U}_{=0}\,(\mathsf{X}\,\varphi_{\mathsf{inst}})].\end{aligned}$$

If $\mathbf{I}_j$ operates on $\mathbf{C}_2$, then define:

$$\varphi_{I_j} := \psi_j \rightarrow (\neg\varphi_{\mathsf{inst}}\,\mathsf{U}_{=0} \bigvee_{\mathbf{I}_k \in S_j} \psi_k) \wedge \varphi'_{\mathbf{C}_1} \wedge \varphi'_{\mathbf{C}_2},$$

where

$$\begin{aligned}\varphi_{\mathbf{C}_1} :=& \mathsf{X}^{2n+5}[(\neg\mathsf{X}^3\varphi_{\mathsf{inst}})\,\mathsf{U}_{=0}\,(\mathsf{X}^3\,\varphi_{\mathsf{inst}})],\\ \varphi_{\mathbf{C}_2} :=& \mathsf{X}^{2n+7}[(\neg\mathsf{X}\varphi_{\mathsf{inst}})\,\mathsf{U}_{=1}\,(\mathsf{X}\,\varphi_{\mathsf{inst}})].\end{aligned}$$

(7) For a decrement instruction I_j: if $\mathbf{C}_1 = 0$ then go to some $I_k \in S_j^1$ else $\mathbf{C}_1 := \mathbf{C}_1 - 1$; go to some $I_m \in S_j^2$, define

$$\begin{aligned}\varphi_{I_j} := \psi_j \rightarrow \langle[\neg\varphi_{\mathsf{inst}}\,\mathsf{U}_{=4}\,\mathsf{X}^3\,\varphi_{\mathsf{inst}} \rightarrow (\neg\varphi_{\mathsf{inst}}\,\mathsf{U}_{=0} \bigvee_{\mathbf{I}_k \in S_j^1} \psi_k) \wedge \varphi_{\mathsf{zero}}]\wedge\\ [\neg\varphi_{\mathsf{inst}}\,\mathsf{U}_{>4}\,\mathsf{X}^3\,\varphi_{\mathsf{inst}} \rightarrow (\neg\varphi_{\mathsf{inst}}\,\mathsf{U}_{=0} \bigvee_{\mathbf{I}_m \in S_j^2} \psi_m) \wedge \varphi_{\mathsf{notzero}}]\rangle,\end{aligned}$$

where

$$\varphi_{\mathsf{zero}} := \mathsf{X}^{2n+5}[(\neg \mathsf{X}^3 \varphi_{\mathsf{inst}})\,\mathsf{U}_{=0}\,(\mathsf{X}^3\,\varphi_{\mathsf{inst}})] \wedge \mathsf{X}^{2n+7}[(\neg \mathsf{X}\varphi_{\mathsf{inst}})\,\mathsf{U}_{=0}\,(\mathsf{X}\,\varphi_{\mathsf{inst}})],$$
$$\varphi_{\mathsf{notzero}} := \mathsf{X}^{2n+5}[(\neg \mathsf{X}^3 \varphi_{\mathsf{inst}})\,\mathsf{U}_{=-1}\,(\mathsf{X}^3\,\varphi_{\mathsf{inst}})] \wedge \mathsf{X}^{2n+7}[(\neg \mathsf{X}\varphi_{\mathsf{inst}})\,\mathsf{U}_{=0}\,(\mathsf{X}\,\varphi_{\mathsf{inst}})].$$

If $\mathbf{I}_j$ operates on $\mathbf{C}_2$, then define:

$$\varphi_{I_j} := \psi_j \rightarrow \langle [\neg\varphi_{\mathsf{inst}}\,\mathsf{U}_{=4}\,\mathsf{X}\,\varphi_{\mathsf{inst}} \rightarrow (\neg\varphi_{\mathsf{inst}}\,\mathsf{U}_{=0} \bigvee_{\mathbf{I}_k \in S_j^1} \psi_k) \wedge \varphi_{\mathsf{zero}}] \wedge$$
$$[\neg\varphi_{\mathsf{inst}}\,\mathsf{U}_{>4}\,\mathsf{X}\,\varphi_{\mathsf{inst}} \rightarrow (\neg\varphi_{\mathsf{inst}}\,\mathsf{U}_{=0} \bigvee_{\mathbf{I}_m \in S_j^2} \psi_m) \wedge \varphi'_{\mathsf{notzero}}] \rangle,$$

where

$$\varphi'_{\mathsf{notzero}} := \mathsf{X}^{2n+5}[(\neg \mathsf{X}^3 \varphi_{\mathsf{inst}})\,\mathsf{U}_{=0}\,(\mathsf{X}^3\,\varphi_{\mathsf{inst}})] \wedge \mathsf{X}^{2n+7}[(\neg \mathsf{X}\varphi_{\mathsf{inst}})\,\mathsf{U}_{=-1}\,(\mathsf{X}\,\varphi_{\mathsf{inst}})].$$

We define two pureMTL-formulas φ_{infin} and φ_{fin} in the following:

$$\varphi_{\mathsf{infin}} := \varphi_{\mathsf{init}} \wedge \varphi_{I_0} \wedge \varphi_{\mathsf{struct}} \wedge \mathsf{GF}\,\psi_0 \wedge \mathsf{G}\,(\varphi_{\mathsf{struct}} \wedge \bigwedge_{0 \le j < n} \varphi_{I_j}),$$
$$\varphi_{\mathsf{fin}} := \varphi_{\mathsf{init}} \wedge \varphi_{I_0} \wedge \varphi_{\mathsf{struct}} \wedge \mathsf{F}\,\varphi_{\mathsf{halt}} \wedge \mathsf{G}\,(\mathsf{F}\,\varphi_{\mathsf{halt}} \rightarrow \varphi_{\mathsf{struct}} \wedge \bigwedge_{0 \le j < n} \varphi_{I_j}).$$

It is easy to check that φ_{infin} (respectively, φ_{fin}) is satisfiable if and only if $\mathbf{M}$ has a recurring computation (respectively, a halting computation). □

4.5 SAT for other fragments of MTL and TPTL

In this section, we consider the satisfiability problem for some other fragments of MTL and TPTL. We show that SAT is still undecidable even for unary MTL with two propositions, but for existential TPTL, SAT is NP-complete.

4.5.1 The satisfiability problem for unary MTL with two propositions

By an observation of the proof for Theorem 5 in [7], we see that no until modality is used in it. It means that this theorem also holds for the pure unary fragment of TPTL. It is of interest

to consider the satisfiability problem for the pure unary fragment of MTL. In Theorem 13 and Theorem 14, we show that SAT is undecidable for unaMTL and pureMTL, respectively. But in the proofs of them, we either use propositions (the number of propositions depends on the number of instructions, which is not fixed) or use the until modality. In the following we show that two propositions are enough for unaMTL to get the undecidability.

Theorem 15. *For the fragment of* unaMTL *with at least two propositions, infinitary SAT is* Σ_1^1*-complete and finitary SAT is* Σ_1^0*-complete.*

Proof. First we give a proof for the theorem that uses four propositions, then we show how to reduce the number of propositions to two. Let $\mathbf{P} = \{\mathbf{Zero}, \mathbf{Instr}, \mathbf{C}_1, \mathbf{C}_2\}$. Suppose that $\mathbf{M}$ is a two-counter machine with instructions $\mathbf{I}_0, \ldots, \mathbf{I}_n$, and $\pi = (\mathbf{I}_{j_0}, c_0, d_0)(\mathbf{I}_{j_1}, c_1, d_1)(\mathbf{I}_{j_2}, c_2, d_2)\ldots$ is a computation of $\mathbf{M}$, where $(\mathbf{I}_{j_0}, c_0, d_0) = (\mathbf{I}_0, 0, 0)$, $j_i \in \{0, \ldots, n\}$ and $c_i, d_i \in \mathbb{N}\,(i \geq 1)$. We encode π into a data word over $\mathbf{P}$ as follows:

$$
\begin{array}{l}
(\mathbf{Zero}, 0)(\mathbf{Instr}, 0)(\mathbf{C}_1, 0)(\mathbf{C}_2, 0) \\
(\mathbf{Zero}, n), (\mathbf{Instr}, n + j_1), (\mathbf{C}_1, n + n \cdot c_1)(\mathbf{C}_2, n + n \cdot d_1) \\
(\mathbf{Zero}, 2n)(\mathbf{Instr}, 2n + j_2)(\mathbf{C}_1, 2n + n \cdot c_2)(\mathbf{C}_2, 2n + n \cdot d_2) \\
\ldots
\end{array}
$$

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

$$(\mathbf{Zero}, i \cdot n), (\mathbf{Instr}, i \cdot n + j_i), (\mathbf{C}_1, i \cdot n + n \cdot c_i)(\mathbf{C}_2, i \cdot n + n \cdot d_i).$$

The data values in the positions where **Zero** holds are strictly monotonic and increase progressively by exactly n. We use these numbers for the zero test operation. If counter $\mathbf{C}_i\,(i \in \{1, 2\})$ is increased by 1 (decreased by 1, not changed, respectively), then the corresponding data value in the encoding is increased by $2n$ (not changed, increased by n, respectively). Hence, the data values in the positions where $\mathbf{C}_1$ (respectively, $\mathbf{C}_2$) holds are also monotonic. We exploit this monotonicity property to get rid of the U modality.

In the following we define several unaMTL-formulas which express that a data word encodes a computation of $\mathbf{M}$ properly.

(1) There is exactly one proposition from $\mathbf{P}$ that holds in every position:

$$\varphi_{\mathsf{prop}} := (\bigvee_{p \in \mathbf{P}} p) \wedge \bigwedge_{p \in \mathbf{P}} (p \rightarrow \bigwedge_{q \in \mathbf{P} \setminus \{p\}} \neg q).$$

(2) The sequence of all propositions in the data word is of the form $\mathbf{Zero}, \mathbf{Instr}, \mathbf{C}_1\mathbf{C}_2, \mathbf{Zero}, \mathbf{Instr}, \mathbf{C}_1\mathbf{C}_2, \ldots$:

$$\varphi_{\mathsf{seq}} := (\mathbf{Zero} \to \mathsf{X}\,\mathbf{Instr} \wedge \mathsf{X}^2\,\mathbf{C}_1 \wedge \mathsf{X}^3\,\mathbf{C}_2) \wedge (\mathbf{C}_2 \to \mathsf{X}\,\mathbf{Zero}).$$

(3) The data values in the positions where **Zero** holds are increasing exactly by n:

$$\varphi_{\mathsf{dat}} := \mathbf{Zero} \to (\mathsf{F}_{=n}\mathbf{Zero} \wedge \neg\mathsf{F}_{<n}\mathbf{Zero}).$$

(4) The data value in the position where **Instr** holds encodes an instruction $\mathbf{I}_j$ $(0 \leq j \leq n)$:

$$\varphi_{\mathsf{instr}} := \mathbf{Zero} \to \mathsf{X}_{[0,n]}\mathbf{Instr}.$$

(5) The data values in the positions where $\mathbf{C}_1$ (respectively, $\mathbf{C}_2$) holds are monotonic, and are no less than the previous data values in the positions where **Zero** holds:

$$\begin{aligned}\varphi_{\mathsf{cntdat}} :=& (\mathbf{C}_1 \to \neg(\mathsf{F}_{<0}\,\mathbf{C}_1)) \wedge (\mathbf{C}_2 \to \neg(\mathsf{F}_{<0}\,\mathbf{C}_2)) \\ &\wedge (\mathbf{Zero} \to \neg(\mathsf{F}_{<0}\,\mathbf{C}_1 \vee \mathsf{F}_{<0}\,\mathbf{C}_2)).\end{aligned}$$

(6) The initial configuration is $(\mathbf{I}_0, 0, 0)$:

$$\varphi_{\mathsf{init}} := \mathbf{Zero} \wedge \mathsf{X}_{=0}(\mathbf{Instr} \wedge \mathsf{X}_{=0}(\mathbf{C}_1 \wedge \mathsf{X}_{=0}\,\mathbf{C}_2)).$$

(7) For the halting instruction $\mathbf{I}_n$ (i.e., **halt**), define

$$\varphi_{\mathsf{halt}} := \mathbf{Zero} \wedge \varphi_{\mathsf{prop}} \wedge \mathsf{X}_{=n}[\mathbf{Instr} \wedge \varphi_{\mathsf{prop}} \wedge \mathsf{X}(\varphi_{\mathsf{prop}} \wedge \mathbf{C}_1 \wedge \mathsf{X}(\varphi_{\mathsf{prop}} \wedge \mathbf{C}_2 \wedge \neg\mathsf{X}\top))].$$

This formula says that the data word ends with $(\mathbf{Zero}, d)(\mathbf{Instr}, d+n)(\mathbf{C}_1, d_1)(\mathbf{C}_2, d_2)$, where $d, d_1, d_2 \in \mathbb{N}$.

(8) For an increment instruction I_j: $\mathbf{C}_1 := \mathbf{C}_1 + 1$; go to some $I_k \in S_j$, define

$$\varphi_{I_j} := \mathbf{Zero} \wedge \mathsf{X}_{=j}\,\mathbf{Instr} \to (\mathsf{F}_{=n}(\mathbf{Zero} \wedge \bigvee_{\mathbf{I}_k \in S_j} \mathsf{X}_{=k}\,\mathbf{Instr}) \wedge \varphi_{\mathbf{C}_1} \wedge \varphi_{\mathbf{C}_2}),$$

where

$$\varphi_{\mathbf{C}_1} := \mathsf{X}^2(\mathbf{C}_1 \wedge \mathsf{F}_{=2n}\,\mathbf{C}_1 \wedge \neg \mathsf{F}_{<2n}\,\mathbf{C}_1),$$
$$\varphi_{\mathbf{C}_2} := \mathsf{X}^3(\mathbf{C}_2 \wedge \mathsf{F}_{=n}\,\mathbf{C}_2 \wedge \neg \mathsf{F}_{<n}\,\mathbf{C}_2).$$

If $\mathbf{I}_j$ operates on $\mathbf{C}_2$, then define:

$$\varphi_{I_j} := \mathbf{Zero} \wedge \mathsf{X}_{=j}\,\mathbf{Instr} \rightarrow (\mathsf{F}_{=n}(\mathbf{Zero} \wedge \bigvee_{\mathbf{I}_k \in S_j} \mathsf{X}_{=k}\,\mathbf{Instr}) \wedge \varphi'_{\mathbf{C}_1} \wedge \varphi'_{\mathbf{C}_2}),$$

where

$$\varphi'_{\mathbf{C}_1} := \mathsf{X}^2(\mathbf{C}_1 \wedge \mathsf{F}_{=n}\,\mathbf{C}_1 \wedge \neg \mathsf{F}_{<n}\,\mathbf{C}_1),$$
$$\varphi'_{\mathbf{C}_2} := \mathsf{X}^3(\mathbf{C}_2 \wedge \mathsf{F}_{=2n}\,\mathbf{C}_2 \wedge \neg \mathsf{F}_{<2n}\,\mathbf{C}_2).$$

(9) For a decrement instruction I_j: if $\mathbf{C}_1 = 0$ then go to some $I_k \in S_j^1$ else $\mathbf{C}_1 := \mathbf{C}_1 - 1$; go to some $I_m \in S_j^2$, define

$$\begin{aligned}\varphi_{I_j} :=& \mathbf{Zero} \wedge \mathsf{X}_{=j}\,\mathbf{Instr} \rightarrow \\ & [(\mathsf{F}_{=0}\,\mathbf{C}_1 \rightarrow \mathsf{F}_{=n}(\mathbf{Zero} \wedge \bigvee_{\mathbf{I}_k \in S_j^1} \mathsf{X}_{=k}\,\mathbf{Instr}) \wedge \varphi_{\mathsf{zero}}) \wedge \\ & (\neg \mathsf{F}_{=0}\,\mathbf{C}_1 \rightarrow \mathsf{F}_{=n}(\mathbf{Zero} \wedge \bigvee_{\mathbf{I}_m \in S_j^2} \mathsf{X}_{=m}\,\mathbf{Instr}) \wedge \varphi_{\mathsf{notzero}})],\end{aligned}$$

where

$$\varphi_{\mathsf{zero}} := \mathsf{X}^2(\mathbf{C}_1 \wedge \mathsf{F}_{=n}\,\mathbf{C}_1 \wedge \neg \mathsf{F}_{<n}\,\mathbf{C}_1) \wedge \mathsf{X}^3(\mathbf{C}_2 \wedge \mathsf{F}_{=n}\,\mathbf{C}_2 \wedge \neg \mathsf{F}_{<n}\,\mathbf{C}_2),$$
$$\varphi_{\mathsf{notzero}} := \mathsf{X}^2(\mathbf{C}_1 \wedge \mathsf{F}_{=0}\,\mathbf{C}_1) \wedge \mathsf{X}^3(\mathbf{C}_2 \wedge \mathsf{F}_{=n}\,\mathbf{C}_2 \wedge \neg \mathsf{F}_{<n}\,\mathbf{C}_2).$$

If $\mathbf{I}_j$ operates on $\mathbf{C}_2$, then define:

$$\begin{aligned}\varphi_{I_j} :=& \mathbf{Zero} \wedge \mathsf{X}_{=j}\,\mathbf{Instr} \rightarrow \\ & [(\mathsf{F}_{=0}\,\mathbf{C}_2 \rightarrow \mathsf{F}_{=n}(\mathbf{Zero} \wedge \bigvee_{\mathbf{I}_k \in S_j^1} \mathsf{X}_{=k}\,\mathbf{Instr}) \wedge \varphi_{\mathsf{zero}}) \wedge \\ & (\neg \mathsf{F}_{=0}\,\mathbf{C}_2 \rightarrow \mathsf{F}_{=n}(\mathbf{Zero} \wedge \bigvee_{\mathbf{I}_m \in S_j^2} \mathsf{X}_{=m}\,\mathbf{Instr}) \wedge \varphi'_{\mathsf{notzero}})],\end{aligned}$$

where

$$\varphi'_{\mathsf{notzero}} := \mathsf{X}^2(\mathbf{C}_1 \wedge \mathsf{F}_{=n}\,\mathbf{C}_1 \wedge \neg \mathsf{F}_{<n}\,\mathbf{C}_1) \wedge \mathsf{X}^3(\mathbf{C}_2 \wedge \mathsf{F}_{=0}\,\mathbf{C}_2).$$

We define two formulas φ_{infin} and φ_{fin} in the following:

$$\begin{aligned}\varphi_{\mathsf{infin}} :=& \varphi_{\mathsf{init}} \wedge \varphi_{\mathsf{prop}} \wedge \varphi_{I_0} \wedge \varphi_{\mathsf{dat}} \wedge \varphi_{\mathsf{cntdat}} \wedge \mathsf{GF}\,(\mathbf{Zero} \wedge \mathsf{X}_{=0}\,\mathbf{Instr}) \wedge \\ & \mathsf{G}\,(\varphi_{\mathsf{prop}} \wedge \varphi_{\mathsf{instr}} \wedge \varphi_{\mathsf{seq}} \wedge \varphi_{\mathsf{dat}} \wedge \varphi_{\mathsf{cntdat}} \wedge \bigwedge_{0 \le j < n} \varphi_{I_j}),\end{aligned}$$

and

$$\begin{aligned}\varphi_{\mathsf{fin}} :=& \varphi_{\mathsf{init}} \wedge \varphi_{\mathsf{prop}} \wedge \varphi_{I_0} \wedge \varphi_{\mathsf{dat}} \wedge \varphi_{\mathsf{cntdat}} \wedge \mathsf{F}\,\varphi_{\mathsf{halt}} \wedge \\ & \mathsf{G}\,(\mathsf{F}\,\varphi_{\mathsf{halt}} \rightarrow \varphi_{\mathsf{prop}} \wedge \varphi_{\mathsf{instr}} \wedge \varphi_{\mathsf{seq}} \wedge \varphi_{\mathsf{dat}} \wedge \varphi_{\mathsf{cntdat}} \wedge \bigwedge_{0 \le j < n} \varphi_{I_j}).\end{aligned}$$

It is easy to check that φ_{infin} (respectively, φ_{fin}) is satisfiable if and only if **M** has a recurring computation (respectively, a halting computation).

We next show how to reduce the number of propositions to two. Let p, q be two propositions. We can replace the propositions $\mathbf{Zero}, \mathbf{Instr}, \mathbf{C}_1, \mathbf{C}_2$ by $\neg p \wedge \neg q, p \wedge q, p \wedge \neg q, \neg p \wedge q$ in the formulas φ_{infin} and φ_{fin}, respectively, and remove the formula φ_{prop}. We can get two new formulas $\varphi'_{\mathsf{infin}}$ and φ'_{fin}. Then, we replace $\mathbf{Zero}, \mathbf{Instr}, \mathbf{C}_1, \mathbf{C}_2$ by $\emptyset, \{p,q\}, \{p\}, \{q\}$ in the data word, respectively. It is easily seen that $\varphi'_{\mathsf{infin}}$ and φ'_{fin} can also capture the recurring computation and halting computation of **M** over the new data word, respectively. □

4.5.2 The satisfiability problem for existential TPTL

In this subsection, we consider the satisfiability problem of the existential fragment of TPTL, in which we only use the F and X modalities, and the negation operator ($\neg$) is only applied to atomic propositions and constraints $x \sim c$. This fragment has also been considered for MTL and TPTL over monotonic timed words [18]. 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. Without explicit state, all data words in this subsection are finite.

Definition 8. The formulas of the existential fragment of TPTL (extTPTL) is defined by the following grammar:

$$\varphi ::= \top \mid \bot \mid p \mid \neg p \mid x \sim c \mid \neg x \sim c \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \mathsf{F}\varphi \mid x.\varphi$$

By a reduction from the propositional satisfiability problem, we see at once that SAT for extTPTL is NP-hard. Actually, we can show that SAT for the fragment of extTPTL with two

register variables and without any propositions is still NP-hard. We use a reduction from the subset sum problem, which is defined as:

Input: A sequence $a_1, \ldots, a_n, b \in \mathbb{N}$ of binary encoded numbers.

Output: `yes` if $\exists\{b_1, \ldots, b_m\} \subseteq \{a_1, \ldots, a_n\}$ such that $\sum_{i=1}^{m} b_i = b$, `no` otherwise.

It is well known that the subset sum problem is NP-complete [77].

Proposition 12. *For the pure fragment of* extTPTL^2*, SAT is* NP*-hard.*

Proof. We prove the proposition by a reduction from the subset sum problem. Let $a_1, \ldots, a_n, b$ ba an instance of this problem, we construct the formula $x.y.\mathsf{F}\varphi_1$, where $\varphi_i\,(1 \leq i \leq n)$ is defined inductively by

$$\varphi_i = \begin{cases} (x = 0 \vee x = a_i) \wedge x.\mathsf{F}\,\varphi_{i+1} & \text{if } 1 \leq i < n, \\ (x = 0 \vee x = a_n) \wedge y = b & \text{if } i = n. \end{cases}$$

If $x.y.\mathsf{F}\varphi_1$ is satisfiable, then there is a data word w such that $w \models x.y.\mathsf{F}\varphi_1$. We define a subset S of $\{a_1, \ldots, a_n\}$ such that for each $a_i\,(1 \leq i \leq n)$, a_i is in S if and only if the constraint $x = a_i$ holds when we evaluate $x.y.\mathsf{F}\varphi_1$ over w. By the constraint $y = b$ we can see that the sum of all numbers in S is b. Conversely, if $a_1, \ldots, a_n, b$ is a positive instance, it is easy to construct a data word w such that $w \models x.y.\mathsf{F}\varphi_1$. □

Let φ be an extTPTL-formula. We define the set $\Gamma(\varphi)$ for φ such that it contains exactly all those formulas that can be obtained from φ by resolving the non-determinism induced by the occurrences of the boolean operator $\vee$. More precisely, $\Gamma(\varphi)$ is defined inductively by the following rules:

- $\Gamma(\varphi) := \{\varphi\}$ if φ is $\top, \bot, p, \neg p, x \sim c$ or $\neg x \sim c$.
- $\Gamma(\varphi_1 \wedge \varphi_2) := \{\psi_1 \wedge \psi_2 \mid \psi_1 \in \Gamma(\varphi_1) \text{ and } \psi_2 \in \Gamma(\varphi_2)\}$.
- $\Gamma(\varphi_1 \vee \varphi_2) := \Gamma(\varphi_1) \cup \Gamma(\varphi_2)$.
- $\Gamma(x.\varphi_1) := \{x.\psi_1 \mid \psi_1 \in \Gamma(\varphi_1)\}$.
- $\Gamma(\mathsf{F}\,\varphi_1) := \{\mathsf{F}\,\psi_1 \mid \psi_1 \in \Gamma(\varphi_1)\}$.
- $\Gamma(\mathsf{X}\,\varphi_1) := \{\mathsf{X}\,\psi_1 \mid_1 \psi \in \Gamma(\varphi_1)\}$.

Let $||\varphi||$ be the size of φ, i.e., the number of all symbols in φ. It is easily seen that $\Gamma(\varphi)$ is a finite set (maybe exponentially larger with respect to $||\varphi||$), and for each formula ψ in $\Gamma(\varphi)$, $||\psi|| \leq ||\varphi||$.

Lemma 10. *Let w be a data word. Then for every* extTPTL*-formula φ, every position i in w and every register valuation ν,*

$$(w,i,\nu) \models \varphi \text{ if and only if } (w,i,\nu) \models \psi \text{ for some } \psi \in \Gamma(\varphi).$$

Proof. We prove the lemma by induction on φ. It is easy for the cases that φ is $\top, \bot, p, \neg p$, $x \sim c$ or $\neg x \sim c$.

- If φ is $\varphi_1 \wedge \varphi_2$, then $(w,i,\nu) \models \varphi_1 \wedge \varphi_2$ if and only if $(w,i,\nu) \models \varphi_1$ and $(w,i,\nu) \models \varphi_2$, by induction hypothesis, if and only if $(w,i,\nu) \models \psi_1$ for some $\psi_1 \in \Gamma(\varphi_1)$ and $(w,i,\nu) \models \psi_2$ for some $\psi_2 \in \Gamma(\varphi_2)$, if and only if $(w,i,\nu) \models \psi_1 \wedge \psi_2$ for some $\psi_1 \in \Gamma(\varphi_1)$ and $\psi_2 \in \Gamma(\varphi_2)$. By the definition of $\Gamma(\varphi_1 \wedge \varphi_2)$, we know that $\psi_1 \wedge \psi_2 \in \Gamma(\varphi_1 \wedge \varphi_2)$.

- If φ is $\varphi_1 \vee \varphi_2$, then $(w,i,\nu) \models \varphi_1 \vee \varphi_2$ if and only if $(w,i,\nu) \models \varphi_1$ or $(w,i,\nu) \models \varphi_2$, by induction hypothesis, if and only if $(w,i,\nu) \models \psi_1$ for some $\psi_1 \in \Gamma(\varphi_1)$ or $(w,i,\nu) \models \psi_2$ for some $\psi_2 \in \Gamma(\varphi_2)$. By the definition of $\Gamma(\varphi_1 \vee \varphi_2)$, we know that ψ_1 and ψ_2 are in $\Gamma(\varphi_1 \vee \varphi_2)$.

- If φ is $x.\varphi_1$, then $(w,i,\nu) \models x.\varphi_1$ if and only if $(w,i,\nu[x \mapsto d_i]) \models \varphi_1$, by induction hypothesis, if and only if $(w,i,\nu[x \mapsto d_i]) \models \psi_1$ for some $\psi_1 \in \Gamma(\varphi_1)$, if and only if $(w,i,\nu) \models x.\psi_1$ for some $\psi_1 \in \Gamma(\varphi_1)$. By the definition of $\Gamma(x.\varphi_1)$, we know that $x.\psi_1 \in \Gamma(x.\varphi_1)$.

- If φ is $\mathsf{F}\varphi_1$, then $(w,i,\nu) \models \mathsf{F}\varphi_1$ if and only if $\exists j > i$ such that $(w,j,\nu) \models \varphi_1$, by induction hypothesis, if and only if $\exists j > i$ such that $(w,j,\nu) \models \psi_1$ for some $\psi_1 \in \Gamma(\varphi_1)$, if and only if $(w,i,\nu) \models \mathsf{F}\psi_1$ for some $\psi_1 \in \Gamma(\varphi_1)$. By the definition of $\Gamma(\mathsf{F}\varphi_1)$, we know that $\mathsf{F}\psi_1 \in \Gamma(\mathsf{F}\varphi_1)$.

- If φ is $\mathsf{X}\varphi_1$, then $(w,i,\nu) \models \mathsf{X}\varphi_1$ if and only if $(w,i+1,\nu) \models \varphi_1$, by induction hypothesis, if and only if $(w,i+1,\nu) \models \psi_1$ for some $\psi_1 \in \Gamma(\varphi_1)$, if and only if $(w,i,\nu) \models \mathsf{X}\psi_1$ for some $\psi_1 \in \Gamma(\varphi_1)$. By the definition of $\Gamma(\mathsf{X}\varphi_1)$, we know that $\mathsf{X}\psi_1 \in \Gamma(\mathsf{X}\varphi_1)$.

□

By the definition of $\Gamma(\varphi)$, all formulas in it do not contain any occurrence of $\vee$. We say that an extTPTL-formula is *simple* if it does not contain any occurrence of $\vee$. Let ψ be a simple extTPTL-formula. We define $\mathsf{sub}(\psi)$ to be the multiset of all subformulas of ψ, where two syntactically equally subformulas occurring in different positions in ψ are considered as different. Furthermore, let Val be the set of all register valuations.

Definition 9. Let w be a data word, i a position in w, let v_0 be a register valuation, and let ψ be a simple extTPTL-formula. A mapping $\theta : \mathsf{sub}(\psi) \mapsto (\{0,\ldots,|w|\} \times \mathsf{Val})$ is a *witness mapping* for (w,i,v_0) and ψ, if it satisfies the following conditions:

(1) $\theta(\psi) = (i, v_0)$.

(2) If $\psi_1 \wedge \psi_2 \in \mathsf{sub}(\psi)$ and $\theta(\psi_1 \wedge \psi_2) = (j, v)$, then $\theta(\psi_1) = (j, v)$ and $\theta(\psi_2) = (j, v)$.

(3) If $\mathsf{F}\,\psi_1 \in \mathsf{sub}(\psi)$ and $\theta(\mathsf{F}\,\psi_1) = (j_1, v)$, then $\theta(\psi_1) = (j_2, v)$ for some $j_2 > j_1$.

(4) If $\mathsf{X}\,\psi_1 \in \mathsf{sub}(\psi)$ and $\theta(\mathsf{X}\psi_1) = (j, v)$, then $\theta(\psi_1) = (j+1, v)$.

(5) If $x.\psi_1 \in \mathsf{sub}(\psi)$ and $\theta(x.\psi_1) = (j, v)$, then $\theta(\psi_1) = (j, v[x \mapsto d_j])$.

(6) If $\psi_1 \in \mathsf{sub}(\psi)$, where ψ_1 is $\top, \bot, p$ or $\neg p$, and $\theta(\psi_1) = (j, v)$, then $(w, j, v) \models \psi_1$.

(7) If $\psi_1 \in \mathsf{sub}(\psi)$, where ψ_1 is $x \sim c$ or $\neg x \sim c$, and $\theta(\psi_1) = (j, v)$ then $(w, j, v) \models \psi_1$.

Intuitively, a witness mapping θ captures the satisfaction relation between the data word w and the formula ψ, i.e., if $\psi_1 \in \mathsf{sub}(\psi)$ and $\theta(\psi_1) = (j, v)$, then $(w, j, v) \models \psi_1$. The witness mapping θ is *preserved under subformulas*, i.e., if ψ_1 is a subformula of ψ and $\theta(\psi_1) = (j, v')$, then the restriction of θ to $\mathsf{sub}(\psi_1)$ is also a witness mapping for (w, j, v') and ψ_1.

Lemma 11. *For every triple (w,i,v_0), where w is a data word, i is a position in w, v_0 is a register valuation, and every simple* extTPTL*-formula ψ, $(w,i,v_0) \models \psi$ if and only if there exists a witness mapping $\theta : \mathsf{sub}(\psi) \mapsto (\{0,\ldots,|w|\} \times \mathsf{Val})$ for (w,i,v_0) and ψ.*

Proof. If $(w,i,v_0) \models \psi$, then there exists a witness mapping θ, as is easy to check. Conversely, suppose that there exists a witness mapping θ for (w,i,v_0) and ψ, we shall show that $(w,i,v_0) \models \psi$. We now proceed by induction on ψ. The proof is easy for the cases that ψ is $\top, \bot, p, \neg p, x \sim c$ or $\neg x \sim c$.

- If ψ is $\psi_1 \wedge \psi_2$, then $(w,i,v_0) \models \psi_1 \wedge \psi_2$ if and only if $(w,i,v_0) \models \psi_1$ and $(w,i,v_0) \models \psi_2$. By the conditions (1) and (2) above we have $\theta(\psi_1) = (i, v_0)$ and $\theta(\psi_2) = (i, v_0)$. Note that θ is preserved under subformulas. Hence, by induction hypothesis, we can obtain $(w,i,v_0) \models \psi_1$ and $(w,i,v_0) \models \psi_2$.

- If ψ is $x.\psi_1$, then $(w,i,v_0) \models x.\psi_1$ if and only if $(w,i,v_0[x \mapsto d_i]) \models \psi_1$. By the conditions (1) and (5) above we have $\theta(\psi_1) = (i, v_0[x \mapsto d_i])$, and by induction hypothesis, we can obtain $(w,i,v_0[x \mapsto d_i]) \models \psi_1$.

- If ψ is $\mathsf{F}\psi_1$, then $(w,i,v_0) \models \mathsf{F}\psi_1$ if and only if $\exists j > i$ such that $(w,j,v_0) \models \psi_1$. By the conditions (1) and (3) above we have $\theta(\psi_1) = (j, v_0)$ for some $j > i$, by induction hypothesis, we can obtain $(w,j,v_0) \models \psi_1$ for some $j > i$.

- If ψ is $\mathsf{X}\psi_1$, then $(w,i,v_0) \models \mathsf{X}\psi_1$ if and only if $(w,i+1,v_0) \models \psi_1$. By the conditions (1) and (4) above we have $\theta(\psi_1) = (i+1, v_0)$, and by induction hypothesis, we can obtain $(w,i+1,v_0) \models \psi_1$.

□

In Theorem 11 we show that if a positive TPTL-formula is satisfiable, then it is satisfied by a finite data word. In the following we show that for a simple extTPTL-formula ψ, if ψ is satisfiable, then it is satisfied by a data word whose length is bounded by $||\psi||$.

Lemma 12. *Let ψ be a simple* extTPTL*-formula. If ψ is satisfiable, then it is satisfied by a data word u such that $|u| \leq ||\psi||$.*

Proof. Suppose that ψ is satisfiable, then there is a data word w such that $w \models \psi$. By Lemma 11, there is a witness mapping $\theta : \mathsf{sub}(\psi) \mapsto (\{0,\ldots,|w|\} \times \mathsf{Val})$ for $(w,0,\overline{0})$ and ψ. Let $\pi := i_0, i_1, \ldots, i_n$ be a sequence of numbers such that $i_0 = 0$, $i_j < i_{j+1}\,(0 \leq j < n)$, and a number i is in π if and only if there exist a formula $\phi \in \mathsf{sub}(\psi)$ and a valuation v such that $\theta(\phi) = (i,v)$. Let w_π be the data word $(P_{i_0}, d_{i_0})(P_{i_1}, d_{i_1}) \cdots (P_{i_n}, d_{i_n})$, where $(P_{i_j}, d_{i_j})\,(0 \leq j \leq n)$ is the $(i_j)^{\text{th}}$ pair in w. By the definition of θ, we see that the length $|w_\pi|$ of w_π is less than or equal to the size $||\psi||$ of ψ. If we show that $(w_\pi, 0, \overline{0}) \models \psi$, then the lemma follows. We prove it by showing that a more general claim:

$$\text{If } \phi \in \mathsf{sub}(\psi) \text{ and } \theta(\phi) = (i_j, v), \text{ then } (w_\pi, j, v) \models \phi.$$

We prove the claim in a bottom-up process: Suppose that it holds for all subformulas of ϕ, we show that it also holds for ϕ.

- If ϕ is $p, \neg p$ or $x \sim c$, by the definition of θ, the claim holds for ϕ.

- If ϕ is $\psi_1 \wedge \psi_2$ and $\theta(\psi_1 \wedge \psi_2) = (i_j, v)$, then $\theta(\psi_1) = (i_j, v)$ and $\theta(\psi_2) = (i_j, v)$. By induction, we have $(w_\pi, j, v) \models \psi_1$ and $(w_\pi, j, v) \models \psi_2$. This implies $(w_\pi, j, v) \models \psi_1 \wedge \psi_2$.

- If ϕ is $x.\psi_1$ and $\theta(x.\psi_1) = (i_j, v)$, then $\theta(\psi_1) = (i_j, v[x \mapsto d_{i_j}])$. By induction, we have $(w_\pi, j, v[x \mapsto d_{i_j}]) \models \psi_1$. This implies $(w_\pi, j, v) \models x.\psi_1$.

- If ϕ is $\mathsf{F}\,\psi_1$ and $\theta(\mathsf{F}\,\psi_1) = (i_j, v)$, then $\theta(\psi_1) = (i_{j'}, v)$ for some $i_{j'} > i_j$. By induction, we have $(w_\pi, j', v) \models \psi_1$. By the definition of π, we know that $j' > j$. Hence, we have $(w_\pi, j, v) \models \mathsf{F}\,\psi_1$.
- If ϕ is $\mathsf{X}\,\psi_1$ and $\theta(\mathsf{X}\,\psi_1) = (i_j, v)$, then $\theta(\psi_1) = (i_j + 1, v)$. By the definition of π we can know that $i_j + 1 = i_{j+1}$. Hence, $\theta(\psi_1) = (i_{j+1}, v)$. By induction, we have $(w_\pi, j+1, v) \models \psi_1$. This implies $(w_\pi, j, v) \models \mathsf{X}\,\psi_1$.

□

Corollary 7. *Let φ be an* extTPTL*-formula. If φ is satisfiable, then it is satisfied by a data word w such that $|u| \leq ||\varphi||$.*

Proof. If φ is satisfiable, then by Lemma 10, there exists a simple extTPTL-formula $\psi \in \Gamma(\varphi)$ such that ψ is also satisfiable. By Lemma 12, there is a data word u such that $u \models \psi$ and $|u| \leq ||\psi||$. By Lemma 10 again, we have $u \models \varphi$. Clearly, $|u| \leq ||\varphi||$. □

Theorem 16. *For* extTPTL*, SAT is* NP*-complete.*

Proof. The lower bound for this problem follows by Proposition 12. For the upper bound, we give two algorithms that can decide this problem in NP in the following.

Let φ be an extTPTL-formula, and let $n = ||\varphi||$. By Corollary 7 we can know that if φ is satisfiable, then it is satisfied by a data word whose length is bounded by n. We define $\mathbf{P}$ to be the set of all propositions occurring in φ, and define Val' to be the set of all mappings from $\{x_1, \ldots, x_k\}$ to $\{\tilde{d}_0, \ldots, \tilde{d}_{n-1}\}$, where $x_1, \ldots, x_k$ are all register variables occurring in φ and $\tilde{d}_0, \ldots, \tilde{d}_{n-1}$ are auxiliary variables.

For technical reasons, we assume that all constraints in φ are in positive form, i.e., the negation operator ($\neg$) can only be applied to atomic propositions. Note that we can replace every $\neg(x \sim c)$ by an equivalent constraint or a disjunction of two constraints, e.g, $\neg(x < a) \equiv (x \geq a)$, $\neg(x > a) \equiv x \leq a$ and $\neg(x = a) \equiv (x > a \vee x < a)$.

Algorithm 1:

---

1. Guess a formula ψ in $\Gamma(\varphi)$.
2. Guess a sequence of pairs $(P_0, \tilde{d}_0)(P_1, \tilde{d}_1)\ldots(P_{n-1}, \tilde{d}_{n-1})$, where $P_i \subseteq \mathbf{P}\,(0 \leq i < n)$.
3. Guess a mapping $\theta : \mathsf{sub}(\psi) \mapsto (\{0, \ldots, n-1\} \times \mathsf{Val}')$.

4. Check whether θ satisfies the conditions (1) - (6) in Definition 9 for the witness mapping, *reject* if no.

5. Build a set of linear inequalities $\mathscr{C}$ such that if $x \sim c \in \mathsf{sub}(\psi)$ and $\theta(x \sim c) = (i, v')$, where $v'(x) = \tilde{d}_j$, then $\tilde{d}_i - \tilde{d}_j \sim c$ is in $\mathscr{C}$.

6. Check whether $\mathscr{C}$ has a solution on $\mathbb{N}$, *accept* if yes, otherwise *reject*.

In Step 1, guessing a formula in $\Gamma(\varphi)$ can be done in polynomial time because it amounts to select which disjunct to remove from $\varphi_1 \vee \varphi_2$ for each subformula $\varphi_1 \vee \varphi_2$ of φ. It is easy to check that Steps 2 - 5 can be done in polynomial time. Finally, we need to show that whether $\mathscr{C}$ has a solution on $\mathbb{N}$ can be checked in polynomial time. The set of linear inequalities $\mathscr{C}$ can be treated as a system of difference constraints (see Section 24.4 in [27]). By Theorem 24.9 in [27], we can use the Bellman-Ford algorithm to check whether a system of difference constraints has a feasible solution, and Bellman-Ford algorithm runs in polynomial time. If the algorithm accepts, then it means that there is a data word that satisfies ψ. By Lemma 10, we know that this data word also satisfies φ.

Let $C = \max\{|c| \mid x \sim c \text{ is a constraint in } \varphi\}$. By Proposition 6 in Chapter 3, we can know that if φ is satisfiable, then it is satisfied by a data word whose data values are all bounded by $C \cdot ||\varphi||$. Using this fact we give another algorithm that computes the data word directly if φ is satisfiable. Let Val'' be the set of all mappings from $\{x_1, \ldots, x_k\}$ to $\{0, \ldots, C \cdot ||\varphi||\}$.

Algorithm 2:

---

(1) Guess a formula ψ in $\Gamma(\varphi)$.

(2) Guess a data word $w := (P_0, d_0)(P_1, d_1)\ldots(P_{n-1}, d_{n-1})$, where $P_i \subseteq \mathbf{P}$ and $d_i \leq C \cdot ||\varphi||$ for each $0 \leq i < n$.

(3) Guess a mapping $\theta : \mathsf{sub}(\psi) \mapsto (\{0, \ldots, n-1\} \times \mathsf{Val}'')$.

(4) Check whether θ is a witness mapping for $(w, 0, \overline{0})$ and ψ, *accept* if yes, otherwise *reject*.

If the algorithm accepts, then by Lemma 11 we can know that $w \models \psi$, and by Lemma 10 we have $w \models \varphi$. □

Define the existential fragment of MTL (extMTL) to be the set of MTL-formulas built by following grammar:

$$\varphi ::= \top \mid \bot \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{X}_I \varphi \mid \mathsf{F}_I \varphi$$

It is obvious that every extMTL-formula is equivalent to an extTPTL-formula. So we can get the following corollary.

Corollary 8. *For* extMTL*, SAT is* NP*-complete.*

Proof. The lower bound can be obtained by a reduction from the propositional satisfiability problem. The upper bound follows from Theorem 16. □

4.6 Summary of satisfiability results

In this section, we give a summary of the computational complexity of satisfiability for different fragments of MTL and TPTL in Table 4.1. On finite data words, the satisfiability of most fragments of MTL and TPTL is Σ_1^0-complete. On infinite data words, the satisfiability of most fragments of MTL and TPTL is Σ_1^1-complete, whereas for the positive fragments is still Σ_1^0-complete. Additionally, for the unary fragments, the same results can also be obtained for unaTPTL where only one register variable and the F modality are allowed, and for unaMTL where at most two propositions are allowed. For the existential fragment of MTL and TPTL, satisfiability over both finite and infinite data words is NP-complete.

	MTL	pureMTL	unaMTL, unaTPTL	posMTL, posTPTL	extMTL, extTPTL
finite	Σ_1^0-complete	Σ_1^0-complete	Σ_1^0-complete	Σ_1^0-complete	NP-complete
infinite	Σ_1^1-complete	Σ_1^1-complete	Σ_1^1-complete	Σ_1^0-complete	NP-complete

Table 4.1 Computational complexity of satisfiability

Chapter 5

The path checking problems for MTL and TPTL

In this chapter, we study the complexity of path checking problems for MTL and TPTL over data words. In Section 5.1 we prove several upper complexity bounds, and in Section 5.2 we prove several lower complexity bounds. In Section 5.4, we extend these results to deterministic one-counter machines. For a logic $\mathscr{L}$ and a class of data words $\mathbf{C}$, we consider the *path checking problem for* $\mathscr{L}$ *over* $\mathbf{C}$:

Input: A data word $w \in \mathbf{C}$ and a formula $\varphi \in \mathscr{L}$.

Output: `yes` if $w \models \varphi$, `no` otherwise.

Data words can be (i) finite or infinite, (ii) monotonic or non-monotonic, (iii) pure or non-pure, and (iv) unary encoded or binary encoded. All infinite data words in this chapter are of the form $u_1(u_2)^{\omega}_{+k}$, where u_1, u_2 are finite data words and $k \in \mathbb{N}$, unless explicitly stated otherwise. For complexity considerations, it makes a difference, whether the numbers c in constraints $x \sim c$ are encoded in binary or unary notation, and similarly for the interval borders in MTL. We write TPTL^{r}_{u}, TPTL_{u}, and MTL_{u} (respectively, TPTL^{r}_{b}, TPTL_{b}, and MTL_{b}) if we want to emphasize that numbers in constraints are encoded in unary (respectively, binary) notation. All upper bounds that hold for a logic $\mathscr{L}$ where constraint numbers (or interval borders) are encoded in binary notation also hold for $\mathscr{L}$ if constraint numbers (or interval borders) are given in unary notation. Conversely, all lower bounds that hold for $\mathscr{L}$ where constraint numbers (or interval borders) are encoded in unary notation also hold for $\mathscr{L}$ if constraint numbers (or interval borders) are given in binary notation.

5.1 The upper complexity bounds

In this section we prove our upper complexity bounds for TPTL, TPTL^r and MTL. All upper bounds that hold for infinite (respectively, binary encoded) data words also hold for finite (respectively, unary encoded) data words.

5.1.1 Polynomial space upper bound for TPTL

For technical reasons, we define a *relative semantics* for TPTL in the following.

Definition 10. Let w be a data word and $i \in \mathbb{N}$ be a position in w, and let δ be a register valuation. The relative satisfaction relation for TPTL, denoted by $\models^{\mathsf{rel}}$, is defined as follows:

- $(w,i,\delta) \models^{\mathsf{rel}} \top$.
- $(w,i,\delta) \models^{\mathsf{rel}} p$ if and only if $p \in P_i$.
- $(w,i,\delta) \models^{\mathsf{rel}} \neg\varphi$ if and only if $(w,i,\nu) \not\models^{\mathsf{rel}} \varphi$.
- $(w,i,\delta) \models^{\mathsf{rel}} \varphi_1 \wedge \varphi_2$ if and only if $(w,i,\nu) \models^{\mathsf{rel}} \varphi_1$ and $(w,i,\nu) \models^{\mathsf{rel}} \varphi_2$.
- $(w,i,\delta) \models^{\mathsf{rel}} \varphi_1 \mathsf{U} \varphi_2$ if and only if there is a position j with $i < j < |w|$ such that $(w,j,\delta + (d_j - d_i)) \models^{\mathsf{rel}} \varphi_2$, and for all positions t with $i < t < j$, $(w,t,\delta + (d_t - d_i)) \models^{\mathsf{rel}} \varphi_1$.
- $(w,i,\delta) \models^{\mathsf{rel}} x \sim c$ if and only if $\delta(x) \sim c$.
- $(w,i,\delta) \models^{\mathsf{rel}} x.\varphi$ if and only if $(w,i,\delta[x \mapsto 0]) \models^{\mathsf{rel}} \varphi$.

We say that data word w satisfies formula φ under the relative semantics, written $w \models^{\mathsf{rel}} \varphi$, if $(w,0,\tilde{0}) \models \varphi$, where $\tilde{0}$ denotes the valuation function that maps all register variables to 0. We show below that $w \models \varphi$ if and only if $w \models^{\mathsf{rel}} \varphi$, which allows to work with the relative semantics. Its main advantage is the following: In TPTL, a constraint $x \sim c$ is `true` under a valuation ν in a position with data value d, if $d - \nu(x) \sim c$ holds. In contrast, under the relative semantics, a constraint $x \sim c$ is `true` under a valuation δ, if $\delta(x) \sim c$ holds.

Lemma 13. *Let w be a data word and d_i be the data value in position i, and let δ, ν be two valuations. If $\delta(x) = d_i - \nu(x)$ for all register variables x, then for all* TPTL*-formulas φ, $(w,i,\nu) \models \varphi$ if and only if $(w,i,\delta) \models^{\mathsf{rel}} \varphi$.*

Proof. We prove this lemma by induction on the formula φ.

- If φ is $\top$ or a proposition p, then $(w,i,\nu) \models \varphi$ if and only if $(w,i,\delta) \models^{\mathsf{rel}} \varphi$.

- If φ is $\neg\varphi_1$, then $(w,i,\nu) \models \neg\varphi_1$ if and only if $(w,i,\nu) \not\models \varphi_1$ if and only if $(w,i,\delta) \not\models^{\mathsf{rel}} \varphi_1$ if and only if $(w,i,\delta) \models^{\mathsf{rel}} \neg\varphi_1$.

- If φ is $\varphi_1 \wedge \varphi_2$, then $(w,i,\nu) \models \varphi_1 \wedge \varphi_2$ if and only if $(w,i,\nu) \models \varphi_1$ and $(w,i,\nu) \models \varphi_2$, if and only if $(w,i,\delta) \models^{\mathsf{rel}} \varphi_1$ and $(w,i,\delta) \models^{\mathsf{rel}} \varphi_2$ if and only if $(w,i,\delta) \models^{\mathsf{rel}} \varphi_1 \wedge \varphi_2$.

- If $\varphi = x \sim c$, then $(w,i,\nu) \models x \sim c$ if and only if $d_i - \nu(x) \sim c$. Since $\delta(x) = d_i - \nu(x)$, the latter holds if and only if $\delta(x) \sim c$, if and only if $(w,i,\delta) \models^{\mathsf{rel}} x \sim c$.

- If $\varphi = x.\varphi_1$, then $(w,i,\nu) \models x.\varphi_1$ if and only if $(w,i,\nu[x \mapsto d_i]) \models \varphi_1$. Since $d_i - \nu[x \mapsto d_i](x) = 0$, $\delta[x \mapsto 0](x) = 0$, and $\delta[x \mapsto 0](y) = d_i - \nu[x \mapsto d_i](y)$ for all $y \neq x$. By induction hypothesis, the latter holds if and only if $(w,i,\delta[x \mapsto 0]) \models^{\mathsf{rel}} \varphi_1$, i.e., $(w,i,\delta) \models^{\mathsf{rel}} x.\varphi_1$.

- If $\varphi = \varphi_1 \mathsf{U} \varphi_2$, then $(w,i,\nu) \models \varphi_1 \mathsf{U} \varphi_2$ if and only if there is a position j with $i < j < |w|$ such that $(w,j,\nu) \models \varphi_2$ and $(w,t,\nu) \models \varphi_1$ for all positions t with $i < t < j$. By induction hypothesis, this holds if and only if there is a position j with $i < j < |w|$ such that $(w,j,\delta + (d_j - d_i)) \models^{\mathsf{rel}} \varphi_2$ and $(w,t,\delta + (d_t - d_i)) \models^{\mathsf{rel}} \varphi_1$ for all positions t with $i < t < j$. This is equivalent to $(w,i,\delta) \models^{\mathsf{rel}} \varphi_1 \mathsf{U} \varphi_2$.

□

Corollary 9. *For all data words w and all* TPTL*-formulas φ, we have $w \models \varphi$ if and only if $w \models^{\mathsf{rel}} \varphi$.*

Lemma 14. *Let w be a data word, and let $k \in \mathbb{N}$. Then for all register valuations ν, all* TPTL*-formulas φ and all positions i in w, $(w,i,\nu) \models \varphi$ if and only if $(w_{+k},i,\nu+k) \models \varphi$.*

Proof. Given an arbitrary TPTL-formula φ, suppose $\varphi \in \mathsf{TPTL}^{r,S}$, where $r \in \mathbb{N}$ and $S \subseteq \mathbb{Z}$ is a finite set. By Lemma 6 in Chapter 3, we have $(w,i,\nu) \equiv_n^{r,S} (w_{+k},i,\nu+k)$ for every valuation ν, every $i \in \mathbb{N}$ and every $n \in \mathbb{N}$. This implies $(w,i,\nu) \models \varphi$ if and only if $(w_{+k},i,\nu+k) \models \varphi$. □

For the next two lemmas, we always assume that u_1 and u_2 are finite data words, $k \geq 0$, $w := u_1(u_2)^{\omega}_{+k}$, $i \geq |u_1|$, and ϕ is a TPTL-formula.

Lemma 15. *For all register valuations δ, $(w,i,\delta) \models^{\mathsf{rel}} \phi$ if and only if $(w,i+|u_2|,\delta) \models^{\mathsf{rel}} \phi$.*

Proof. Let δ be a register valuation. Define two register valuations ν, ν' by $\nu(x) = d_i - \delta(x)$ and $\nu'(x) = \nu(x) + k$ for every register variable x. By Lemma 13, we have $(w,i,\delta) \models^{\mathsf{rel}} \phi$ if and only if $(w,i,\nu) \models \phi$, and $(w,i+|u_2|,\delta) \models^{\mathsf{rel}} \phi$ if and only if $(w,i+|u_2|,\nu') \models \phi$, since $\nu'(x) = \nu(x) + k = d_i - \delta(x) + k = d_{i+|u_2|} - \delta(x)$. We prove that $(w,i,\nu) \models \phi$ if and only

if $(w, i+|u_2|, v') \models \phi$; the lemma then follows. By Lemma 14, $(w,i,v) \models \phi$ if and only if $(w_{+k}, i, v+k) \models \phi$. Since $i \geq |u_1|$, we have $w_{+k}[i:] = w[(i+|u_2|):]$. So the latter holds if and only if $(w, i+|u_2|, v') \models \phi$. □

For a TPTL-formula ϕ and a finite data word v we define:

$$C_\phi = \max\{c \in \mathbb{Z} \mid x \sim c \text{ is a constraint in } \phi\} \tag{5.1}$$

$$M_v = \max\{d_i - d_j \mid d_i \text{ and } d_j \text{ are data values in } v\} \geq 0 \tag{5.2}$$

We may always assume that $C_\phi \geq 0$ (we can add a dummy constraint $x \geq 0$). Note that in the infinite data word v^ω_{+k}, for all positions $j \geq i$ we have $d_j - d_i + M_v \geq 0$ (where as usual d_l is the data value in position l).

Lemma 16. *Let δ be a register valuation and define the register valuation δ' by $\delta'(x) = \min\{\delta(x), C_\phi + M_{u_2} + 1\}$ for all x. For every subformula θ of ϕ, we have $(w,i,\delta) \models^{\mathsf{rel}} \theta$ if and only if $(w,i,\delta') \models^{\mathsf{rel}} \theta$.*

Proof. Define two register valuations v, v' by $v(x) = d_i - \delta(x)$ and $v'(x) = d_i - \delta'(x)$ for every register variable x. Let $j \geq i$. For every constraint $x \sim c$ in ϕ, by Lemma 13, we have $(w,j,v) \models x \sim c$ if and only if $(w,j,d_j - d_i + \delta) \models^{\mathsf{rel}} x \sim c$ if and only if $(d_j - d_i + \delta) \sim c$, and $(w,j,v') \models x \sim c$ if and only if $(w,j,d_j - d_i + \delta') \models^{\mathsf{rel}} x \sim c$ if and only if $(d_j - d_i + \delta') \sim c$.

We prove that $(w,j,v) \models x \sim c$ if and only if $(w,j,v') \models x \sim c$, then by Lemma 6, this lemma follows. If $\delta(x) \leq C_\phi + M_{u_2} + 1$, then $\delta'(x) = \delta(x)$. So assume $\delta(x) > C_\phi + M_{u_2} + 1$, and hence $\delta'(x) = C_\phi + M_{u_2} + 1$. Then $d_j - d_i + \delta(x) > d_j - d_i + C_\phi + M_{u_2} + 1 \geq C_\phi + 1$ and $d_j - d_i + \delta'(x) = d_j - d_i + C_\phi + M_{u_2} + 1 \geq C_\phi + 1$. This implies $(d_j - d_i + \delta) \sim c$ if and only if $(d_j - d_i + \delta') \sim c$ since $c \leq C_\phi$. □

We can now prove a PSPACE upper bound for our most general path checking problem: path checking for TPTL_b over infinite binary encoded data words.

Theorem 17. *Path checking for TPTL_b over infinite binary encoded data words is in* PSPACE.

Proof. Fix two finite data words u_1, u_2, a number $k \in \mathbb{N}$ and a TPTL-formula ψ, and let $w = u_1(u_2)^\omega_{+k}$. We show that one can decide in APTIME (= PSPACE) whether $w \models \psi$ holds. We first deal with the case $k > 0$ and later sketch the necessary adaptations for the (simpler) case $k = 0$. Without loss of generality, we further assume ψ to be in negation normal form, i.e., negations only appear in front of atomic propositions. Recall that we define $\varphi_1 \mathsf{R} \varphi_2 := \neg(\neg\varphi_1 \mathsf{U} \neg\varphi_2)$. So we can use the release operator R to translate a TPTL-formula into negation normal form. Define $C := C_\psi$ and $M := M_{u_2}$ by (5.1) and (5.2).

The non-trivial cases in our alternating polynomial time algorithm are the ones for the formulas of form $\varphi_1 \mathsf{U} \varphi_2$ and $\varphi_1 \mathsf{R} \varphi_2$. Consider a position i and a register valuation δ. We have $(w,i,\delta) \models^{\mathsf{rel}} \varphi_1 \mathsf{U} \varphi_2$ if and only if there exists $j > i$ such that $(w,j,\delta + d_j - d_i) \models^{\mathsf{rel}} \varphi_2$ and for all $i < t < j$, $(w,t,\delta + d_t - d_i) \models^{\mathsf{rel}} \varphi_1$. Because w is an infinite word, j could be arbitrarily large. Our first goal is to derive a bound on j. Suppose that $0 \leq i \leq |u_1| + |u_2| - 1$; this is no restriction by Lemma 15. Define

$$m_\delta = \min\{\delta(x) \mid x \text{ is a register variable in } \psi\}, \tag{5.3}$$

$$m_1 = \max\{d_i - d_j \mid d_i \text{ and } d_j \text{ are data values in } u_1 u_2\} \text{ and} \tag{5.4}$$

$$m_2 = \min\{d \mid d \text{ is a data value in } u_2\}. \tag{5.5}$$

Let $n \geq 2$ be the minimal number such that $m_\delta + m_2 + (n-1)k - d_i \geq C + M + 1$, i.e. (here we assume $k > 0$),

$$n = \max\{2, \left\lceil \frac{C + M + 1 + d_i - m_\delta - m_2}{k} \right\rceil + 1\}. \tag{5.6}$$

Let $h \geq |u_1| + (n-1)|u_2|$, then for every register variable x from ψ we have

$$\delta(x) + d_h - d_i \geq m_\delta + d_h - d_i \geq m_\delta + m_2 + (n-1)k - d_i \geq C + M + 1.$$

By Lemmas 15 and 16, for every $h \geq |u_1| + (n-1)|u_2|$ we have

$$(w,h,\delta + d_h - d_i) \models^{\mathsf{rel}} \varphi_2 \Leftrightarrow (w, h + |u_2|, \delta + d_{h+|u_2|} - d_i) \models^{\mathsf{rel}} \varphi_2.$$

Therefore, the position j witnessing $(w,j,\delta + d_j - d_i) \models^{\mathsf{rel}} \varphi_2$ can be bounded by $|u_1| + n|u_2|$. Similarly, we can get the same result for $\varphi_1 \mathsf{R} \varphi_2$.

We sketch an alternating Turing machine **TM** that, given a TPTL_b-formula ψ and a data word w, has an accepting run if and only if $w \models \psi$. The machine **TM** first computes and stores the value $C + M + 1$. In every configuration, **TM** stores a triple (i, δ, φ), where i is a position in the data word, δ is a register valuation (with respect to the relative semantics), and φ is a subformula of ψ. By Lemma 15, we can restrict i to the interval $[0, |u_1| + |u_2| - 1]$, and by Lemma 16, we can restrict the range of δ to the interval $[-m_1, \max\{m_1, C + M + 1\}]$. The machine **TM** starts with the triple $(0, \tilde{0}, \psi)$, where $\tilde{0}(x) = 0$ for each register variable x. Then, **TM** branches according to the following rules, where we define the function $\rho : \mathbb{N} \to [0, |u_1| + |u_2| - 1]$ by

$$\rho(z) = \begin{cases} z & \text{if } z < |u_1|, \\ ((z - |u_1|) \bmod |u_2|) + |u_1| & \text{otherwise.} \end{cases}$$

If φ is of the form $\top$, $\bot$, p, $\neg p$, or $x \sim c$, then accept if $(w, i, \delta) \models^{\mathsf{rel}} \varphi$, and reject otherwise.

If $\varphi = \varphi_1 \wedge \varphi_2$, then branch universally to (i, δ, φ_1) and (i, δ, φ_2).

If $\varphi = \varphi_1 \vee \varphi_2$, then branch existentially to (i, δ, φ_1) and (i, δ, φ_2).

If $\varphi = x.\varphi_1$, then go to $(i, \delta[x \mapsto 0], \varphi_1)$.

If $\varphi = \varphi_1 \mathsf{U} \varphi_2$, then first compute the value n according to (5.3), (5.5), and (5.6), and then branch existentially to each value $j \in [i+1, |u_1| + n|u_2|]$, and finally branch universally to each triple from $\{(\rho(t), \delta_t, \varphi_1) \mid i < t < j\} \cup \{(\rho(j), \delta_j, \varphi_2)\}$, where for all register variables x in ψ:

$$\delta_j(x) = \begin{cases} \min\{\delta(x) + d_j - d_i, C + M + 1\} & \text{if } j \geq |u_1|, \\ \delta(x) + d_j - d_i & \text{otherwise,} \end{cases}$$

$$\delta_t(x) = \begin{cases} \min\{\delta(x) + d_t - d_i, C + M + 1\} & \text{if } t \geq |u_1|, \\ \delta(x) + d_t - d_i & \text{otherwise.} \end{cases}$$

If $\varphi = \varphi_1 \mathsf{R} \varphi_2$, then first compute the value n according to (5.3), (5.5), and (5.6) and then branch existentially to the following two alternatives:

- Branch universally to all triples from $\{(\rho(j), \delta_j, \varphi_2) \mid i < j \leq |u_1| + n|u_2|\}$, where

$$\delta_j(x) = \begin{cases} \min\{\delta(x) + d_j - d_i, C + M + 1\} & \text{if } j \geq |u_1|, \\ \delta(x) + d_j - d_i & \text{otherwise.} \end{cases}$$

- Branch existentially to each value $j \in [i+1, |u_1| + n|u_2|]$, and then branch universally to all triples from $\{(\rho(t), \delta_t, \varphi_2) \mid i < t \leq j\} \cup \{(\rho(j), \delta_j, \varphi_1)\}$, where for all register variables x in ψ:

$$\delta_j(x) = \begin{cases} \min\{\delta(x) + d_j - d_i, C + M + 1\} & \text{if } j \geq |u_1|, \\ \delta(x) + d_j - d_i & \text{otherwise,} \end{cases}$$

$$\delta_t(x) = \begin{cases} \min\{\delta(x) + d_t - d_i, C + M + 1\} & \text{if } t \geq |u_1|, \\ \delta(x) + d_t - d_i & \text{otherwise.} \end{cases}$$

The machine **TM** clearly works in polynomial time.

Let us briefly discuss the necessary changes for the case $k = 0$ (i.e., $w = u_1(u_2)^\omega$). The main difficulty in the above algorithm is to find the upper bound of the witnessing position j for the formulas $\varphi_1 \mathsf{U} \varphi_2$ and $\varphi_1 \mathsf{R} \varphi_2$. If $k = 0$, then it is easily seen that for all positions $i \geq |u_1|$, formulas φ and valuations ν, $(w,i,\nu) \models \varphi$ if and only if $(w, i+|u_2|, \nu) \models \varphi$. We see at once that the witnessing position j can be bounded by $|u_1| + 2|u_2|$. It is straightforward to implement the necessary changes in the above algorithm. □

It is easy to adapt the proof in the above theorem to finite data words. We have the following theorem.

Theorem 18. *Path checking for* TPTL_b *over finite data words is in* PSPACE.

5.1.2 Polynomial time upper bound for TPTL$^{\mathbf{r}}$

In this subsection, we consider the complexity of path checking for the logic TPTL^r ($r \in \mathbb{N}$), which is a fragment of TPTL where the number of register variables is bounded by r.

In Theorem 17, we show that path checking for TPTL_b over infinite binary encoded data words is in PSPACE. For TPTL^r, if all input numbers are encoded in unary notation, then we can show that the path checking problem is in P.

Theorem 19. *For every fixed* $r \in \mathbb{N}$, *path checking for* TPTL^r_u *over infinite unary encoded data words is in* P.

Proof. In the algorithm from the proof of Theorem 17, if all numbers are given in unary, then the numbers $C + M + 1$, m_1, m_2 and n can be computed in logarithmic space and are bounded polynomially in the input size. Moreover, a configuration triple (i, δ, φ) needs only logarithmic space: Clearly, the position $i \in [0, |u_1| + |u_2| - 1]$ and the subformula φ only need logarithmic space. The valuation δ is an r-tuple over $[-m_1, \max\{m_1, C + M + 1\}]$ and hence needs logarithmic space too, since r is a constant. Hence, the alternating machine from the proof of Theorem 17 works in logarithmic space. The theorem follows, since ALOGSPACE = P. □

Later (see Theorem 31), we will see that path checking for TPTL^2_u over infinite binary encoded data words is PSPACE-complete. Hence, we cannot (unless P = PSPACE) extend Theorem 19 to infinite binary encoded data words. But if we consider infinite binary encoded monotonic data words, then we can do so. Actually, a condition slightly weaker than monotonicity suffices. We define quasi-monotonic data words in the following. Recall that we use $\min(u)$ and $\max(u)$ to denote the minimal data value and the maximal data value in the data word u, respectively.

Definition 11. Let u_1 and u_2 be two finite data words, and let $k \in \mathbb{N}$. The infinite data word $u_1(u_2)^{\omega}_{+k}$ is quasi-monotonic if $\max(u_1) \leq \max(u_2) \leq \min(u_2) + k$.

Note that if $u_1(u_2)^{\omega}_{+k}$ is monotonic, then $u_1(u_2)^{\omega}_{+k}$ is also quasi-monotonic.

Theorem 20. *For every fixed $r \in \mathbb{N}$, path checking for TPTL^r_u over infinite binary encoded quasi-monotonic data words is in* P.

Proof. Let ψ be a TPTL^r_u-formula, $k \in \mathbb{N}$, and u_1, u_2 two finite data words such that $u_1(u_2)^{\omega}_{+k}$ is quasi-monotonic. We construct in polynomial time two unary encoded finite data words v_1, v_2 and a number $l \in \mathbb{N}$ encoded in unary notation such that $u_1(u_2)^{\omega}_{+k} \models \psi$ if and only if $v_1(v_2)^{\omega}_{+l} \models \psi$. Then we can apply Theorem 19.

Define $S = \{c \mid x \sim c \text{ is a constraint in } \psi\}$ and $C = \max\{|c| \mid c \in S\}$. Suppose that $|u_1u_2| = n$. Let $a_1, \ldots, a_n$ be an enumeration of all data values in u_1u_2 such that $a_j \leq a_{j+1}$ for all $1 \leq j < n$. For each $1 < i \leq n$, define $\delta_i = d_i - d_{i-1}$. We define a new sequence $b_1, \ldots, b_n$ inductively as follows: $b_1 = 0$ and for all $1 < i \leq n$,

$$b_i = \begin{cases} b_{i-1} + \delta_i & \text{if } \delta_i \leq C, \\ b_{i-1} + C + 1 & \text{if } \delta_i > C. \end{cases}$$

We obtain the new data words v_1 and v_2 by replacing in u_1 and u_2 every data value a_i by b_i ($1 \leq i \leq n$). Note that $b_n \leq (C+1) \cdot (n-1)$. Since C is given in unary notation, we can compute in polynomial time the unary encodings of the numbers $b_1, \ldots, b_n$.

To define the number l, note that $\delta = \min(u_2) + k - \max(u_2)$ is the difference between the smallest data value in $(u_2)_{+k}$ and the largest data value in u_2 (which is also the largest data value of u_1u_2). Since $u_1(u_2)^{\omega}_{+k}$ is quasi-monotonic, we have $\delta \geq 0$. We define the number l as

$$l = \begin{cases} \max(v_2) - \min(v_2) + \delta & \text{if } \delta \leq C, \\ \max(v_2) - \min(v_2) + C + 1 & \text{if } \delta > C. \end{cases}$$

Again, the unary encoding of l can be computed in polynomial time. Let d_i (respectively, d'_i) be the data value in the i^{th} position of $u_1(u_2)^{\omega}_{+k}$ (respectively, $v_1(v_2)^{\omega}_{+l}$). Similar to the proof of Proposition 6 in Chapter 3, we have $d_{j_2} - d_{j_1} \overset{S}{\equiv} d'_{j_2} - d'_{j_1}$ for any j_1, j_2 such that $0 \leq j_1 < j_2$. By Lemma 6, this implies that $u_1(u_2)^{\omega}_{+k} \models \psi$ if and only if $v_1(v_2)^{\omega}_{+l} \models \psi$. Applying Theorem 19, we can check whether $v_1(v_2)^{\omega}_{+l} \models \psi$ holds in polynomial time. □

For finite data words, we obtain a polynomial time algorithm also for binary encoded non-monotonic data words (assuming again a fixed number of register variables):

Theorem 21. *For every fixed $r \in \mathbb{N}$, path checking for TPTL_b^r over finite binary encoded data words is in* P.

Proof. Let the input data word w be of length n and let $d_1, \ldots, d_n$ be the data values appearing in w. Moreover, let $x_1, \ldots, x_r$ be the register variables appearing in the input formula ψ. Then, we only have to consider the n^r many valuation mappings $\delta : \{x_1, \ldots, x_r\} \to \{d_1, \ldots, d_n\}$. For each of these mappings δ, every subformula φ of ψ, and every position i in w we check whether $(w, i, \delta) \models \varphi$. This information is computed bottom-up (with respect to the structure of ψ) in the usual way. □

For infinite data words, we have to reduce the number of register variables to one in order to get a polynomial time complexity for binary encoded non-monotonic data words. First we prove several lemmas in the following.

Let φ be a TPTL-formula. We say that a register variable x *occurs free in* φ if there exists a occurrence of x in φ is not within the scope of the corresponding freeze quantifier. A TPTL-formula φ is *closed* if there are no free register variables in φ. We show that when checking the satisfiability of a formula over a data word only the values for free register variables matter in the following.

Lemma 17. *Let w be a data word and i be a position in w. For all* TPTL*-formulas φ and register valuations ν_1 and ν_2, if $\nu_1(x) = \nu_2(x)$ for every register variable x that occurs free in φ, then $(w, i, \nu_1) \models \varphi$ if and only if $(w, i, \nu_2) \models \varphi$.*

Proof. We prove this lemma by induction on φ. The proof for the cases that φ is $\top$, p, $\neg\varphi_1$ or $\varphi_1 \wedge \varphi_2$ is easy. We only give the proof for the other cases.

- If φ is $x \sim c$, then $(w, i, \nu_1) \models x \sim c$ if and only if $d_i - \nu_1(x) \sim c$. Since ν_1 and ν_2 coincide on free register variables in φ, the latter holds if and only if $d_i - \nu_2(x) \sim c$ if and only if $(w, i, \nu_2) \models x \sim c$.

- If φ is $x.\varphi_1$, then $(w, i, \nu_1) \models x.\varphi_1$ if and only if $(w, i, \nu_1[x \mapsto d_i]) \models \varphi_1$. Note that $\nu_1[x \mapsto d_i]$ and $\nu_2[x \mapsto d_i]$ again satisfy the premise of the lemma, by induction hypothesis, the latter holds if and only if $(w, i, \nu_2[x \mapsto d_i]) \models \varphi_1$ if and only if $(w, i, \nu_2) \models x.\varphi_1$.

- If φ is $\varphi_1 \mathsf{U} \varphi_2$, then $(w, i, \nu_1) \models \varphi_1 \mathsf{U} \varphi_2$ if and only if there is a position j with $i < j < |w|$ such that $(w, j, \nu_1) \models \varphi_2$, and $(w, t, \nu_1) \models \varphi_1$ for all positions t with $i < t < j$. By induction hypothesis this holds, if and only if there is a position j with $i < j < |w|$ such that $(w, j, \nu_2) \models \varphi_2$, and $(w, t, \nu_2) \models \varphi_1$ for all positions t with $i < t < j$. This is equivalent to $(w, i, \nu_2) \models \varphi_1 \mathsf{U} \varphi_2$.

□

Corollary 10. *Let w be a data word and i be a position in w, and let φ be a closed* TPTL*-formula. Then for all register valuations v_1 and v_2, $(w,i,v_1) \models \varphi$ if and only if $(w,i,v_2) \models \varphi$.*

Lemma 18. *The following problem belongs to* P *(in fact, to* $\mathsf{AC}^1(\mathsf{LogDCFL})$*):*

Input: *An* LTL*-formula ψ, finite words $u_1, u_2, \ldots, u_l, u_{l+1}$ and numbers $N_1, \ldots, N_l \in \mathbb{N}$ that are encoded in binary notation.*

Question: *Does $u_1^{N_1} u_2^{N_2} \cdots u_l^{N_l} u_{l+1}^{\omega} \models \psi$ hold?*

Proof. The crucial point is that for all finite words u, v, every infinite word w and every number $N \geq ||\psi||$ where $||\psi||$ is the number of all symbols in ψ, we have $uv^N w \models \psi$ if and only if $uv^{||\psi||} w \models \psi$. This can be shown by using the Ehrenfeucht-Fraïssé game for LTL from [37]. It is similar to the EF-game for MTL where we do not need to consider the data values. Let us briefly explain this game. Let w_0, w_1 be two infinite words. A game configuration is a pair of positions $(i_0, i_1) \in \mathbb{N} \times \mathbb{N}$, where i_0 is a position in w_0 and i_1 is a position in w_1. The game is played by two players: Spoiler and Duplicator. In each round, Spoiler chooses an index $a \in \{0,1\}$ and a position $j_a > i_a$. Then Duplicator has to respond with a position $j_{1-a} > i_{1-a}$. Then Spoiler chooses between one of the following two options:

- The new configuration becomes (j_0, j_1).
- Spoiler chooses a position $i_{1-a} < j'_{1-a} < j_{1-a}$, then Duplicator has to respond with a position $i_a < j'_a < j_a$, and the new configuration becomes (j'_0, j'_1).

Duplicator wins the 0-round EF-game from configuration (i_0, i_1) if $w_0[i_0] = w_1[i_1]$. Duplicator wins the $(k+1)$-round EF-game ($k \geq 0$) from configuration (i_0, i_1) if $w_0[i_0] = w_1[i_1]$, and for every choice of moves of Spoiler in the first round, Duplicator wins the k-round EF-game from the successor configuration (j_0, j_1) (or (j'_0, j'_1)). It was shown in [37] that Duplicator wins the k-round EF-game from position $(0,0)$ if and only if for every LTL-formula φ whose until rank is at most k, we have $w_0 \models \varphi$ if and only if $w_1 \models \varphi$.

Now assume that $w_0 = uv^{m_0} w$ and $w_1 = uv^{m_1} w$, where $m_0, m_1 \geq k$, and u, v are finite words, and w is an infinite word. It is then obvious that Duplicator can win the k-round EF-game starting from position $(0,0)$. The point is that Duplicator can enforce that after the first round the new configuration (i_0, i_1) satisfies one of the following two conditions:

- $w_0[i_0 :] = w_1[i_1 :]$, which implies that Duplicator can win for every number of rounds starting from (i_0, i_1).

- $w_0[i_0:]$ (respectively, $w_1[i_1:]$) has the form $u'v^{n_0}w$ (respectively, $u'v^{n_1}w$), where $n_0, n_1 \geq k-1$. Hence, by induction Duplicator can win the $(k-1)$-round EF-game from configuration (i_0, i_1).

Hence, $uv^{m_0}w$ and $uv^{m_1}w$ satisfy the same LTL-formulas whose until rank is at most k. It follows that two infinite words $u_1^{m_1}u_2^{m_2}\cdots u_l^{m_l}u_{l+1}^{\omega}$ and $u_1^{n_1}u_2^{n_2}\cdots u_l^{n_l}u_{l+1}^{\omega}$ satisfy the same LTL-formulas whose until rank is at most k if all $m_i, n_i\,(1 \leq i \leq l)$ are at least k.

Now, the proof of the lemma is obvious. We can replace the binary encoded exponents N_i in the word $u_1^{N_1}u_2^{N_2}\cdots u_l^{N_l}u_{l+1}^{\omega}$ by numbers $N_i' = \min\{N_i, ||\psi||\}\,(1 \leq i \leq l)$. By Theorem 3.6 of [59], infinite path checking for LTL can be reduced to finite path checking for LTL. And finite path checking for LTL is in $\mathsf{AC}^1(\mathsf{LogDCFL})$ [55]. So checking whether $u_1^{n_1}u_2^{n_2}\cdots u_l^{n_l}u_{l+1}^{\omega} \models \psi$ is in $\mathsf{AC}^1(\mathsf{LogDCFL})$. □

Remark 4. One can generalize Lemma 18 to so called exponential expressions of constant exponentiation depth. Let **P** be a finite set of atomic propositions. Exponential expressions are inductively defined as follows:

- Every $P \subseteq \mathbf{P}$ is an exponential expression.
- If e_1 and e_2 are exponential expressions, then $e_1 \cdot e_2$ is an exponential expression.
- If e is an exponential expression and $n \geq 1$, then e^n is an exponential expression.

In the last point, e^n has to be viewed as a formal expression, encoded for instance by the pair (e, n). The number n is assumed to be binary encoded. The length of an exponential expression is defined inductively by (i) $|P| = 1$, (ii) $|e_1 \cdot e_2| = |e_1| + |e_2|$, and (iii) $|e^n| = |e| + \lceil \log_2 n \rceil$. The exponentiation depth $d(e)$ of e is defined inductively by (i) $d(P) = 0$, (ii) $d(e_1 \cdot e_2) = \max\{d(e_1), d(e_2)\}$, and (iii) $d(e^n) = d(e) + 1$. Every exponential expression produces a word $\mathsf{val}(e)$ in the obvious way. The length of this word can be exponential in the length of e.

The exponential expressions appearing in Lemma 18 have exponentiation depth 1. But the proof works for exponential expressions of exponentiation depth at most d for every fixed constant d. Hence, path checking for LTL is in $\mathsf{AC}^1(\mathsf{LogDCFL})$ if the input word is given by an exponential expression of bounded exponentiation depth. This is interesting since by Theorem 5.1 of [59] it was shown that path checking for LTL becomes PSPACE-complete if the input word is represented by a straight-line context-free grammar, i.e., a context-free grammar that produces exactly one string. Every exponential expression can be converted in logarithmic space into an equivalent straight-line context-free grammar. This leaves the question whether path checking for LTL is below PSPACE if the input word is given by an exponential expressions of unbounded exponentiation depth.

Lemma 19. *The following problem belongs to* P *(in fact, to* $\mathsf{AC}^1(\mathsf{LogDCFL})$*):*

Input: *A* TPTL_b*-formula* ψ*, which only contains free register variables, and an infinite binary encoded data word* $u_1(u_2)^\omega_{+k}$*.*

Question: *Does* $u_1(u_2)^\omega_{+k} \models \psi$ *hold?*

Proof. Let $w = u_1(u_2)^\omega_{+k}$. We reduce the question, whether $w \models \psi$ in logarithmic space to an instance of the succinct LTL path checking problem from Lemma 18. Let $n_1 = |u_1|$ and $n_2 = |u_2|$. We can assume that only one register variable x appears in ψ (since we do not use the freeze quantifier in ψ all register variables remain at the initial value d_0).

In order to construct an LTL-formula from ψ, it remains to eliminate occurrences of constraints $x \sim c$ in ψ. We can assume that all constraints are of the form $x < c$ or $x > c$. Let $x \sim_1 c_1, \ldots, x \sim_m c_m$ be a list of all constraints that appear in ψ. We introduce for every $1 \leq j \leq m$ a new atomic proposition p_j and let $\mathbf{P}' = \mathbf{P} \cup \{p_1, \ldots, p_m\}$, where $\mathbf{P}$ is the set of atomic propositions occurring in ψ. Let ψ' be obtained from ψ by replacing every occurrence of $x \sim_j c_j$ by p_j, and let $w' \in (2^{\mathbf{P}'})^\omega$ be the infinite word such that $w'[i] = P_i \cup \{p_j \mid 1 \leq j \leq m, d_i - d_0 \sim_j c_j\}$. Clearly $w \models \psi$ if and only if $w' \models \psi'$. We will show that the word w' can be written in the form considered in Lemma 18.

First of all, we can write w' as $w' = u'_1 u'_{2,0} u'_{2,1} u'_{2,2} \cdots$, where $|u'_1| = n_1$ and $|u'_{2,z}| = n_2$. The word u'_1 can be computed in logarithmic space by evaluating all constraints in all positions of u_1. Moreover, every word $u'_{2,z}$ is obtained from u_2 (without the data values) by adding the new propositions p_j at the appropriate positions. Consider the equivalence relation $\equiv$ on $\mathbb{N}$ such that $a \equiv b$ if and only if $u'_{2,a} = u'_{2,b}$. The crucial observations are that (i) every equivalence class of $\equiv$ is an interval (let us call these intervals $\equiv$-intervals), and (ii) the index of $\equiv$ is bounded by $1 + n_2 \cdot m$ (one plus length of u_2 times number of constraints). To see this, consider a position $0 \leq i \leq n_2 - 1$ in the word u_2 and a constraint $x \sim_j c_j$ ($1 \leq j \leq m$). Then, the truth value of "proposition p_j is present at the i^{th} position of $u'_{2,z}$" switches (from `true` to `false` or from `false` to `true`) at most once when z grows. The reason for this is that the data value in position $n_1 + i + n_2 \cdot z$ is $d_{n_1+i+n_2 \cdot z} = d_{n_1+i} + k \cdot z$ for $z \geq 0$, i.e., it grows monotonically with z. Hence, the truth value of $d_{n_1+i} + k \cdot z - d_0 \sim_j c_j$ switches at most once, when z grows. So, we get at most $n_2 \cdot m$ many "switching points" in $\mathbb{N}$ which produce at most $1 + n_2 \cdot m$ many intervals.

Let $I_1, \ldots, I_l$ be a list of all $\equiv$-intervals, where $a < b$ whenever $a \in I_i$, $b \in I_j$ and $i < j$ (note that I_l must be infinite). The borders of these intervals can be computed in logarithmic space using basic arithmetic on binary encoded numbers. Recall that all arithmetical operations (addition, subtraction, multiplication and division with remainder) can be carried out in logarithmic space on binary encoded numbers [46]. Hence, we can compute in logarithmic

space the size $N_i = |I_i|$ of the i^{th} interval, where $N_l = \omega$. Also, for every $1 \leq i \leq l$ we can compute in logarithmic space the unique word v_i such that $v_i = u'_{2,a}$ for all $a \in I_i$. Finally, we have $w' = u'_1 v_1^{N_1} \cdots v_l^{N_l}$. We are now in the position to apply Lemma 18. □

We now come to the path checking for TPTL^1 over infinite data words.

Theorem 22. *Path checking for* TPTL^1_b *over infinite binary encoded data words is in* P.

Proof. Given a TPTL^1_b-formula ψ, finite binary encoded data words u_1, u_2, and binary encoded number $k \in \mathbb{N}$. Let $w = u_1(u_2)^{\omega}_{+k}$. We give two proofs to show that how to check in polynomial time whether $w \models \psi$ holds in the following. The first one (**# 1**) is based on Lemmas 17, 18 and 19. The second one (**# 2**) is an algorithm-based proof which decides this problem directly.

 # 1: For a closed formula φ, by Corollary 10, we can write $(w,i) \models \varphi$ for $(w,i,\nu) \models \varphi$, where ν is an arbitrary register valuation. By Lemma 14, we can get the following claim:

Claim 1: If φ is closed and $i \geq |u_1|$, then $(w,i) \models \varphi$ if and only if $(w, i+|u_2|) \models \varphi$.

Let $n = |u_1| + |u_2|$. It suffices to compute for every (necessarily closed) subformula $x.\varphi$ of ψ the set of all positions $i \in [0, n-1]$ such that $(w,i) \models x.\varphi$, or equivalently $w[i:] \models \varphi$. We do this in a bottom-up process. Consider a subformula $x.\varphi$ of ψ and a position $i \in [0, n-1]$. We have to check whether $w[i:] \models \varphi$. Let $x.\varphi_1, \ldots, x.\varphi_l$ be a list of all subformulas of φ that are not in the scope of another freeze quantifier within φ. We can assume that for every $1 \leq s \leq l$ we have already determined the set of positions $j \in [0, n-1]$ such that $(w,j) \models x.\varphi_s$. We can therefore replace every subformula $x.\varphi_s$ of φ by a new atomic proposition p_s and add in the data words u_1 (respectively, u_2) the proposition p_s to all positions j_1 (respectively, j_2) such that $(w, j_1) \models x.\varphi_s$ (respectively, $(w, |u_1| + j_2) \models x.\varphi_s$), where $j_1 \in [0, |u_1| - 1]$ and $j_2 \in [0, |u_2| - 1]$. Here, we make use of Claim 1. We denote the resulting formula and the resulting data word with φ' and $w' = u'_1 (u'_2)^{\omega}_{+k}$, respectively. Next, it is easy to compute from u'_1 and u'_2 new finite data words v_1 and v_2 such that $v_1 (v_2)^{\omega}_{+k} = w'[i:]$: If $i < |u'_1|$ then we take $v_1 = u'_1[i:]$ and $v_2 = u'_2$; If $|u'_1| \leq i \leq n-1$, then we take $v_1 = u'_2[i:]$ and $v_2 = (u'_2)_{+k}$. Finally, using Lemma 19 we can check in polynomial time whether $w'[i:] \models \varphi'$ holds. The first proof is complete.

 # 2: Let ν be a register valuation, and let φ be a TPTL^1-formula. We define a tuple of sets $S_{\varphi,\nu} = (S_0, S_1, \ldots, S_{|u_2|})$ where $S_0 \subseteq \{0, \ldots, |u_1| - 1\}$ and $S_h \subseteq \mathbb{N} (1 \leq h \leq |u_2|)$, such that for all $0 \leq i < |u_1|$, $i \in S_0$ if and only if $(w,i,\nu) \models \varphi$, and for each $1 \leq h \leq |u_2|$ and all $j \geq 0$, $j \in S_h$ if and only if $(w, |u_1| + j \cdot |u_2| + h - 1, \nu) \models \varphi$, i.e., S_h contains all the numbers j such that φ holds in the h^{th} position of the j^{th} repetition of u_2 in w. We use $S^h_{\varphi,\nu}$ to denote the h^{th}

($0 \le h \le |u_2|$) component S_h of $S_{\varphi,\nu}$. If φ is a closed formula, then by Corollary 10, we can skip the subscript ν and write S_φ (respectively, S^h_φ) for $S_{\varphi,\nu}$ (respectively, $S^h_{\varphi,\nu}$).

For every $0 \le i < |u_1u_2|$, let ν_i denote the register valuation with $\nu_i(x) = d_i$ (we only need to consider the valuation for the register variable x here). We will compute for every $0 \le i < |u_1u_2|$ and every subformula φ of ψ the tuple S_{φ,ν_i}. Every set S^h_{φ,ν_i} will be represented by a union of polynomially many intervals that are pairwise disjoint, each of which is either a closed interval $[a,b]$ or a half closed interval $[a,+\infty)$, where $a,b \in \mathbb{N}$. Note that $w \models \psi$ if and only if $0 \in S^0_{\psi,\nu_0}$ (we assume without loss of generality that u_1 is not the empty word).

For a set $S \subseteq \mathbb{N}$, let $S_{-1} = \{a-1 \mid a \ge 1, a \in S\}$. We compute the tuples $S_{\varphi,\nu_i}\,(0 \le i < |u_1u_2|)$ bottom-up with respect to the structure of the formula φ as follows:

Case 1. φ is $\top$. Then $S^0_\varphi = \{0,\ldots,|u_1|-1\}$, and $S^h_\varphi = \mathbb{N}$ for $1 \le h \le |u_2|$.

Case 2. φ is an atomic proposition p. For each $0 \le s < |u_1|$, $s \in S^0_\varphi$ if and only if $p \in u_1[s]$. For each $1 \le h \le |u_2|$, S^h_φ is either $\mathbb{N}$ if $p \in u_2[h-1]$or $\emptyset$ otherwise.

Case 3. φ is a constraint formula $x \sim c$. For each $0 \le s < |u_1|$, $s \in S^0_{\varphi,\nu_i}$ if and only if $(u_1,s,\nu_i) \models x \sim c$. For each S^h_{φ,ν_i} $(1 \le h \le |u_2|)$, note that the sequence of data values of w in positions $|u_1| + n \cdot |u_2| + h - 1\,(n \ge 0)$ is a non-decreasing arithmetic progression $d_{|u_1|+h-1}, d_{|u_1|+h-1}+k, d_{|u_1|+h-1}+2k, \ldots$. Then, the interval borders for S^h_{φ,ν_i} can be easily computed. For example, suppose $\varphi = (x \ge c)$. We need to find the minimal number $n \ge 0$ such that $d_{|u_1|+h-1} + nk - \nu_i(x) \ge c$, which is

$$n = \max\{\left\lceil \frac{c + \nu_i(x) - d_{|u_1|+h-1}}{k} \right\rceil, 0\}.$$

Then, we set $S^h_{\varphi,\nu_i} = [n,+\infty)$. Similar calculation works for the other constraint formulas.

Case 4. $\varphi = \varphi_1 \wedge \varphi_2$. Then $S^h_{\varphi,\nu_i} = S^h_{\varphi_1,\nu_i} \cap S^h_{\varphi_2,\nu_i}$.

Case 5. $\varphi = \neg\varphi_1$. Then, $S^0_{\varphi,\nu_i} = \{0,\ldots,|u_1|-1\} \setminus S^0_{\varphi_1,\nu_i}$ and $S^h_{\varphi,\nu_i} = \mathbb{N} \setminus S^j_{\varphi_1,\nu_i}\,(1 \le h \le |u_2|)$.

Case 6. $\varphi = x.\varphi_1$. Then $S^0_\varphi = \{i \mid 0 \le i < |u_1|, i \in S^0_{\varphi_1,\nu_i}\}$ and, for each $1 \le h \le |u_2|$, $S^h_\varphi = \mathbb{N}$ if $|u_1| + h - 1 \in S^h_{\varphi_1,\nu_{|u_1|+h-1}}$, and $S^h_\varphi = \emptyset$ otherwise.

Case 7. $\varphi = \varphi_1 \mathsf{U} \varphi_2$. First, for each $0 \le s < |u_1|$, we have $s \in S^0_{\varphi,\nu_i}$ if and only if one of the following two cases holds:

- There exists $j \in [s+1, |u_1|-1]$ such that $j \in S^0_{\varphi_2,\nu_i}$ and $[s+1, j-1] \subseteq S^0_{\varphi_1,\nu_i}$.
- $[s+1,|u_1|-1] \subseteq S^0_{\varphi_1,\nu_i}$ and there are $m_1 \ge 0$ and $1 \le m_2 \le |u_2|$ such that $[0,m_1-1] \subseteq S^1_{\varphi_1,\nu_i} \cap \cdots \cap S^h_{\varphi_1,\nu_i}$ and $m_1 \in S^1_{\varphi_1,\nu_i} \cap \cdots \cap S^{m_2-1}_{\varphi_1,\nu_i} \cap S^{m_2}_{\varphi_2,\nu_i}$.

Both cases can be easily checked in polynomial time.

In order to compute the sets S^h_{φ,ν_i} $(1 \leq h \leq |u_2|)$, let

$$R_1 = S^{h+1}_{\varphi_1,\nu_i} \cap \cdots \cap S^{|u_2|}_{\varphi_1,\nu_i} \cap (S^1_{\varphi_1,\nu_i} \cap \cdots \cap S^h_{\varphi_1,\nu_i})_{-1}.$$

Let $R^1_2 = S^{h+1}_{\varphi_2,\nu_i}$ $((S^1_{\varphi_2,\nu_i})_{-1}$, if $h = |u_2|$). For each $2 \leq s \leq |u_2| - h$, let

$$R^s_2 = S^{h+1}_{\varphi_1,\nu_i} \cap \cdots \cap S^{h+s-1}_{\varphi_1,\nu_i} \cap S^{h+s}_{\varphi_2,\nu_i}.$$

Let

$$R^{|u_2|-h+1}_2 = S^{h+1}_{\varphi_1,\nu_i} \cap \cdots \cap S^{|u_2|}_{\varphi_1,\nu_i} \cap (S^1_{\varphi_2,\nu_i})_{-1},$$

and for each $|u_2| - h + 2 \leq s \leq |u_2|$, let

$$R^s_2 = S^{h+1}_{\varphi_1,\nu_i} \cap \cdots \cap S^{|u_2|}_{\varphi_1,\nu_i} \cap (S^1_{\varphi_1,\nu_i} \cap \cdots \cap S^{s+h-|u_2|-1}_{\varphi_1,\nu_i} \cap S^{s+h-|u_2|}_{\varphi_2,\nu_i})_{-1}.$$

Let

$$R_2 = R^1_2 \cup \cdots \cup R^{|u_2|}_2.$$

Define

$$S^h_{\varphi,\nu_i} = R_2 \cup \bigcup \{[a,b] \mid [a,b] \text{ is contained in } R_1 \setminus R_2 \text{ and } b+1 \in R_2\}.$$

A number j is in R_1 if and only if φ_1 holds (w.r.t. valuation ν_i) in the interval of length $|u_2|$ starting at the $(h+1)^{\text{th}}$ position of the j^{th} iteration of u_2 in w (or at the first position the $(j+1)^{\text{th}}$ iteration of u_2 in w if $h = |u_2|$). A number j is in R_2 if and only if φ_2 (w.r.t. valuation ν_i) holds in a position that is at most $|u_2|$ positions after the h^{th} position of the j^{th} iteration of u_2 in w, and φ_1 holds (w.r.t. valuation ν_i) between these two positions.

If $[a,b]$ is contained in $R_1 \setminus R_2$ and $b+1 \in R_2$, then it is easily seen that $\varphi_1 \mathsf{U} \varphi_2$ holds in each position $|u_1| + j \cdot |u_2| + h - 1$ for $j \in [a,b]$. Conversely, if $\varphi_1 \mathsf{U} \varphi_2$ holds in position $t = |u_1| + j \cdot |u_2| + h - 1$, then either φ_2 holds in a position that is at most $|u_2|$ positions after position t and φ_1 holds between t and that position (hence $j \in R_2$), or there exists $j' > j$ such that $j' \in R_2$ and φ_1 holds from position $t+1$ up to position $|u_1| + j' \cdot |u_2| + h - 1$. In the latter case, we can choose j' minimal with this property. This implies $[j, j'-1] \subseteq R_1 \setminus R_2$.

Finally, we need to show that each S^h_{φ,ν_i} contains only polynomially many intervals. It is sufficient to show that the number of all interval borders in the algorithm above is polynomially bounded. We use four kinds of set operations: union, intersection, complementation and subtraction. Union and intersection do not add any new interval borders. So we only need to consider complementation and subtraction of 1.

For a set $B \subseteq \mathbb{N}$ and $k \geq 1$, write $B_{-k} = \{a-k \mid a \geq k, a \in B\}$ and $B_{+k} = \{a+k \mid a \in B\}$. Let $\gamma_1, \ldots, \gamma_m$ be all constraints $x \sim c$ that appear in ψ. Let

$$B_0 = \bigcup_{i=0}^{|u_1u_2|-1} \bigcup_{j=1}^{m} \bigcup_{h=1}^{|u_2|} \{a \in \mathbb{N} \mid a \text{ is a border of an interval in } S^h_{\gamma_j, v_i}\}.$$

Thus, B_0 is the set of all interval borders that arise from constraint subformulas. Without loss of generality, we assume that $0 \in B_0$. For $n \geq 1$ define

$$B_n = B_{n-1} \cup (B_{n-1})_{-1} \cup (B_{n-1})_{+1}.$$

By induction on n, one can show that

$$B_n = B_0 \cup \bigcup_{k=1}^{n} (B_0)_{-k} \cup (B_0)_{+k}.$$

Hence $|B_n| \leq (2n+1)|B_0|$, where $|B_i|$ be the cardinality of B_i. Subtraction decreases each interval border by 1, and complementation may decrease or increase an interval border by 1. Suppose that all interval borders are in B_n. If we do complementation or subtraction, then the new interval borders are in B_{n+1}. There are polynomially many complementation and subtraction operations and $|B_0|$ is polynomially bounded. So the number of all interval borders is polynomially bounded. □

Since for every MTL_b-formula, we can compute in logarithmic space an equivalent TPTL^1_b-formula. The next corollary follows from Theorem 22.

Corollary 11. *Path checking for* MTL_b *over infinite binary encoded data words is in* P.

5.1.3 $\mathsf{AC}^1(\mathsf{logDCFL})$ upper bound for MTL

By Corollary 11, we know that path checking for MTL over infinite data words is in P. It is shown that path checking for MTL over finite monotonic data words is in $\mathsf{AC}^1(\mathsf{logDCFL})$ [21]. In this subsection, we consider path checking for MTL over infinite monotonic data words of the form $(u)^{\omega}_{+k}$. We can show that the complexity for this problem is still in $\mathsf{AC}^1(\mathsf{logDCFL})$. First we prove several lemmas.

Lemma 20. *Let u be a data word, and let $k \in \mathbb{N}$. Then for every* MTL*-formula φ and $i \geq 0$, $(u,i) \models \varphi$ if and only if $(u_{+k}, i) \models \varphi$.*

Proof. The lemma follows by Corollary 1 in Chapter 3. □

Corollary 12. *Let u be a finite data word, and let $k \in \mathbb{N}$. Then for every* MTL*-formula φ and $i \geq 0$, $((u)^{\omega}_{+k}, i) \models \varphi$ if and only if $((u)^{\omega}_{+k}, i+|u|) \models \varphi$.*

Lemma 21. *Let u be a finite data word, and let $k \in \mathbb{N}$. For every* MTL*-formula φ, if there exists $i \geq 0$ such that for all $i \leq j < i+|u|$, $((u)^{\omega}_{+k}, j) \models \varphi$, then for all $i' \geq 0$, $((u)^{\omega}_{+k}, i') \models \varphi$.*

Proof. Let φ be an MTL-formula. Suppose that there exists $i \geq 0$ such that for all $i \leq j < i+|u|$, $((u)^{\omega}_{+k}, j) \models \varphi$. Then for all $i' \geq 0$, there exists $i \leq j' < i+|u|$ and $n \in \mathbb{Z}$ such that $i' = j' + n|u|$. By Corollary 12, we have $((u)^{\omega}_{+k}, i') \models \varphi$. □

The next several results are proved over monotonic data words. Let w ba a monotonic data word. For every MTL-formula $\varphi_1 \mathsf{U}_I \varphi_2$, it is easily seen that $w \models \varphi_1 \mathsf{U}_I \varphi_2$ if and only if $w \models \varphi_1 \mathsf{U}_{I \cap [0,+\infty)} \varphi_2$, i.e., we can make the constraint interval I be non-negative. Every open interval over $\mathbb{Z}$ is equivalent to a closed (or half-closed) interval over $\mathbb{Z}$, e.g., $(a,b) = [a+1, b-1]$ and $(a,+\infty) = [a+1,+\infty)$, where $a, b \in \mathbb{Z}$. So for technical reasons, we assume all constraint intervals in the MTL-formulas have the form $[a,b]$ or $[a,+\infty)$, where $a, b \geq 0$, in the following. This restriction does not influence the results.

Lemma 22. *Let $(u)^{\omega}_{+k}$ be an infinite monotonic data word. For every* MTL*-formula $\varphi_1 \mathsf{U}_I \varphi_2$ and $i \geq 0$, where I is the infinite interval $[a,+\infty)$ or the finite interval $[a,b]$ $(0 \leq a \leq b)$, we have:*

(1) $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,+\infty)} \varphi_2$ *if and only if* $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,a+k]} \varphi_2$.

(2) If $b-a > k$, *then* $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,b]} \varphi_2$ *if and only if* $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,a+k]} \varphi_2$.

(3) If $a > k$, *then* $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,b]} \varphi_2$ *if and only if* $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a',b']} \varphi_2$, *where*

$$a' = \begin{cases} k + (a \bmod k) & \text{if } (a \bmod k) \neq 0, \\ 2k & \text{otherwise,} \end{cases}$$

and $b' = a' + b - a$.

Proof. (1) Because $[a, a+k] \subseteq [a,+\infty)$, the direction "$\Leftarrow$" is trivial. For the direction "$\Rightarrow$", suppose $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,+\infty)} \varphi_2$. Then there exists $j > i$ such that $((u)^{\omega}_{+k}, j) \models \varphi_2$, $d_j - d_i \geq a$ and $((u)^{\omega}_{+k}, t) \models \varphi_1$ for all $i < t < j$. Since $(u)^{\omega}_{+k}$ is monotonic, there exists $i < j' \leq i+|u|$ and $n \in \mathbb{N}$ such that $j' = j - n|u|$, and $d_{j'} - d_i \in [a, a+k]$. By Corollary 12, we have $((u)^{\omega}_{+k}, j') \models \varphi_2$. In addition, we have $((u)^{\omega}_{+k}, t') \models \varphi_1$ for all $i < t' < j'$. Therefore, $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,a+k]} \varphi_2$.

(2) The proof is the same as that of (1), where we only need to replace $+\infty$ by b.

(3) Let $a' = k + (a \bmod k)$ and $b' = a' + b - a$. Suppose $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,b]} \varphi_2$. Then there exists $j > i$ such that $((u)^{\omega}_{+k}, j) \models \varphi_2$, $d_j - d_i \in [a,b]$, and $((u)^{\omega}_{+k}, t) \models \varphi_1$ for all $i < t < j$. Since $(u)^{\omega}_{+k}$ is monotonic, there exists $i < j' \leq i + |u|$ and $n \geq 0$ such that $j' = j - n|u|$, and $d_{j'} - d_i \in [a', b']$. By Corollary 12, we have $((u)^{\omega}_{+k}, j') \models \varphi_2$. In addition, we have $((u)^{\omega}_{+k}, t') \models \varphi_1$ for all $i < t' < j'$, since $j' \leq j$. Therefore, $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a',b']} \varphi_2$.

Conversely, if $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a',b']} \varphi_2$, then there exists $j' > i$ such that $((u)^{\omega}_{+k}, j') \models \varphi_2$, $d_{j'} - d_i \in [a', b']$ and $((u)^{\omega}_{+k}, t') \models \varphi_1$ for all $i < t' < j'$. Since $(u)^{\omega}_{+k}$ is monotonic, there exists $j \geq j'$ and $n \geq 0$ such that $j = j' + n|u|$, and $d_j - d_i \in [a,b]$. By Corollary 12, we have $((u)^{\omega}_{+k}, j) \models \varphi_2$. Since $a' > k$, we see that $j' > i + |u|$. Because $((u)^{\omega}_{+k}, t') \models \varphi_1$ for all $i < t' < j'$, by Lemma 21, we have $((u)^{\omega}_{+k}, t) \models \varphi_1$ for all $i < t < j$. Therefore, $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,b]} \varphi_2$.

□

By Lemma 22, given an infinite monotonic data word $w = (u)^{\omega}_{+k}$, for every MTL-formula φ, we can construct an equivalent formula φ' with respect to w by replacing each constraint interval I in φ by a finite constraint interval I'. More precisely, all numbers in I' can be bounded by $3k$. Because w is monotonic, when checking the formula $\varphi_1 \mathsf{U}_{I'} \varphi_2$ over w in position i, we only need to consider the positions j with $i < j \leq i + 3|u|$ to check whether φ_2 holds in j and φ_1 holds in the positions between i and j.

Given a number $k > 0$, we define a function f_k from intervals to finite intervals as follows:

$$f_k([a, +\infty)) = \begin{cases} [a, a+k] & \text{if } a \leq k, \\ [a', a'+k] & \text{otherwise,} \end{cases}$$

and

$$f_k([a,b]) = \begin{cases} [a,b] & \text{if } a \leq k \text{ and } b - a \leq k, \\ [a, a+k] & \text{if } a \leq k \text{ and } b - a > k, \\ [a', a'+b-a] & \text{if } a > k \text{ and } b - a \leq k, \\ [a', a'+k] & \text{if } a > k \text{ and } b - a > k, \end{cases}$$

where

$$a' = \begin{cases} k + (a \bmod k) & \text{if } (a \bmod k) \neq 0, \\ 2k & \text{otherwise.} \end{cases}$$

It is clear that, for each interval I, $f_k(I) \subseteq [0,3k]$ and $f_k(f_k(I)) = f_k(I)$.

By Lemma 22, we can obtain the following corollary.

Corollary 13. *Let $(u)^{\omega}_{+k}$ be an infinite monotonic data word. For every* MTL*-formula $\varphi_1 \mathsf{U}_I \varphi_2$ and $i \geq 0$, $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_I \varphi_2$ if and only if $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{f_k(I)} \varphi_2$.*

Proof. If I is the infinite interval $[a,+\infty)$, then by (1) of Lemma 22, $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,+\infty)} \varphi_2$ if and only if $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,a+k]} \varphi_2$. If $a > k$, by (3) of Lemma 22, the latter holds if and only if $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a',a'+k]} \varphi_2$, where a' is $k + (a \bmod k)$ if $(a \bmod k) \neq 0$, or $2k$ otherwise.

If I is the finite interval $[a,b]$, $a \leq k$ and $b - a > k$, then by (2) of Lemma 22, $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,b]} \varphi_2$ if and only if $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,a+k]} \varphi_2$. If I is $[a,b]$ and $a > k$, then by (3) of Lemma 22, $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,b]} \varphi_2$ if and only if $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a',a'+b-a]} \varphi_2$, where a' is $k + (a \bmod k)$ if $(a \bmod k) \neq 0$, or $2k$ otherwise. If $b - a > k$, by (2) of Lemma 22, the latter holds if and only if $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{[a,a+k]} \varphi_2$. □

Given the function f_k, for every MTL-formula φ, we recursively build an MTL-formula $\overline{\varphi}$ as follows:

- $\overline{\top} = \top$,
- $\overline{p} = p$,
- $\overline{\neg\varphi} = \neg\overline{\varphi}$,
- $\overline{\varphi_1 \wedge \varphi_2} = \overline{\varphi_1} \wedge \overline{\varphi_2}$,
- $\overline{\varphi_1 \mathsf{U}_I \varphi_2} = \overline{\varphi_1} \mathsf{U}_{f_k(I)} \overline{\varphi_2}$.

Lemma 23. *Let $(u)^{\omega}_{+k}$ be an infinite monotonic data word, for every* MTL*-formula φ and $i \geq 0$, $((u)^{\omega}_{+k}, i) \models \varphi$ if and only if $((u)^{\omega}_{+k}, i) \models \overline{\varphi}$.*

Proof. We prove the lemma by induction on φ.

- If φ is $\top$ or a proposition p, then $((u)^{\omega}_{+k}, i) \models \varphi$ if and only if $((u)^{\omega}_{+k}, i) \models \overline{\varphi}$.
- If φ is $\neg\varphi_1$, then $((u)^{\omega}_{+k}, i) \models \neg\varphi_1$ if and only if $((u)^{\omega}_{+k}, i) \not\models \varphi_1$ if and only if $((u)^{\omega}_{+k}, i) \not\models \overline{\varphi_1}$ if and only if $((u)^{\omega}_{+k}, i) \models \neg\overline{\varphi_1}$ if and only if $((u)^{\omega}_{+k}, i) \models \overline{\neg\varphi_1}$.
- If φ is $\varphi_1 \wedge \varphi_2$, then $((u)^{\omega}_{+k}, i) \models \varphi_1 \wedge \varphi_2$ if and only if $((u)^{\omega}_{+k}, i) \models \varphi_1$ and $((u)^{\omega}_{+k}, i) \models \varphi_2$, if and only if $((u)^{\omega}_{+k}, i) \models \overline{\varphi_1}$ and $((u)^{\omega}_{+k}, i) \models \overline{\varphi_2}$, if and only if $((u)^{\omega}_{+k}, i) \models \overline{\varphi_1} \wedge \overline{\varphi_2}$ if and only if $((u)^{\omega}_{+k}, i) \models \overline{\varphi_1 \wedge \varphi_2}$.

- If φ is $\varphi_1 \mathsf{U}_I \varphi_2$, then by Corollary 13, $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_I \varphi_2$ if and only if $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_{f_k(I)} \varphi_2$, if and only if there exists $j > i$ such that $((u)^{\omega}_{+k}, j) \models \varphi_2$, $d_j - d_i \in f_k(I)$, and $((u)^{\omega}_{+k}, t) \models \varphi_1$ for all $i < t < j$. By induction hypothesis this holds, if and only if there exists $j > i$ such that $((u)^{\omega}_{+k}, j) \models \overline{\varphi_2}$, $d_j - d_i \in f_k(I)$, and $((u)^{\omega}_{+k}, t) \models \overline{\varphi_1}$ for all $i < t < j$. This is equivalent to $((u)^{\omega}_{+k}, i) \models \overline{\varphi_1} \mathsf{U}_{f_k(I)} \overline{\varphi_2}$, which holds if and only if $((u)^{\omega}_{+k}, i) \models \overline{\varphi_1 \mathsf{U}_I \varphi_2}$.

□

We now reduce path checking for MTL over infinite data words to path checking for MTL over finite data words.

Proposition 13. *Let $(u)^{\omega}_{+k}$ be an infinite monotonic data word, and let φ be an* MTL*-formula. For all $n \geq 3\mathsf{Rank}(\varphi) + 1$ and $0 \leq i < |u|$, $((u)^{\omega}_{+k}, i) \models \varphi$ if and only if $((u)^{n}_{+k}, i) \models \overline{\varphi}$.*

Proof. By Lemma 23, it suffices to show that $((u)^{\omega}_{+k}, i) \models \overline{\varphi}$ if and only if $((u)^{n}_{+k}, i) \models \overline{\varphi}$. We now proceed by induction on $\overline{\varphi}$.

- If $\overline{\varphi}$ is $\top$ or a proposition p, then $((u)^{\omega}_{+k}, i) \models \overline{\varphi}$ if and only if $(u, i) \models \overline{\varphi}$.

- If $\overline{\varphi}$ is $\neg\varphi_1$, then $((u)^{\omega}_{+k}, i) \models \neg\varphi_1$ if and only if $((u)^{\omega}_{+k}, i) \not\models \varphi_1$. By induction hypothesis this holds, if and only if $((u)^{n}_{+k}, i) \not\models \varphi_1$ if and only if $((u)^{n}_{+k}, i) \models \neg\varphi_1$.

- If $\overline{\varphi}$ is $\varphi_1 \wedge \varphi_2$, then $((u)^{\omega}_{+k}, i) \models \varphi_1 \wedge \varphi_2$ if and only if $((u)^{\omega}_{+k}, i) \models \varphi_1$ and $((u)^{\omega}_{+k}, i) \models \varphi_2$. By induction hypothesis, this holds if and only if $((u)^{n}_{+k}, i) \models \varphi_1$ and $((u)^{n}_{+k}, i) \models \varphi_2$, if and only if $((u)^{n}_{+k}, i) \models \varphi_1 \wedge \varphi_2$.

- If $\overline{\varphi}$ is $\varphi_1 \mathsf{U}_I \varphi_2$, then $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_I \varphi_2$ if and only if there exists $j > i$ such that $((u)^{\omega}_{+k}, j) \models \varphi_2$, $d_j - d_i \in I$, and $((u)^{\omega}_{+k}, t) \models \varphi_1$ for all $i < t < j$. Suppose $j = m|u| + r$ for some $m \geq 0$ and $0 \leq r < |u|$. By Corollary 12, $((u)^{\omega}_{+k}, j) \models \varphi_2$ if and only if $((u)^{\omega}_{+k}, r) \models \varphi_2$. Since $0 \leq i < |u|$ and $I \subseteq [0, 3k]$, we know that $0 \leq j < 4|u|$, hence $0 \leq m \leq 3$ and $n - m \geq 3\mathsf{Rank}(\varphi_1 \mathsf{U}_I \varphi_2) - 2 \geq 3\mathsf{Rank}(\varphi_2) + 1$. By induction hypothesis, $((u)^{\omega}_{+k}, r) \models \varphi_2$ if and only if $((u)^{n-m}_{+k}, r) \models \varphi_2$. By Lemma 20 this holds, if and only if $((u)^{n}_{+k}, m|u| + r) \models \varphi_2$ if and only if $((u)^{n}_{+k}, j) \models \varphi_2$. Therefore, we have $((u)^{\omega}_{+k}, j) \models \varphi_2$ if and only if $((u)^{n}_{+k}, j) \models \varphi_2$. Similarly, we can show that $((u)^{\omega}_{+k}, t) \models \varphi_1$ if and only if $((u)^{n}_{+k}, t) \models \varphi_1$ for all $i < t < j$. Hence $((u)^{\omega}_{+k}, i) \models \varphi_1 \mathsf{U}_I \varphi_2$ if and only if $((u)^{n}_{+k}, i) \models \varphi_1 \mathsf{U}_I \varphi_2$.

□

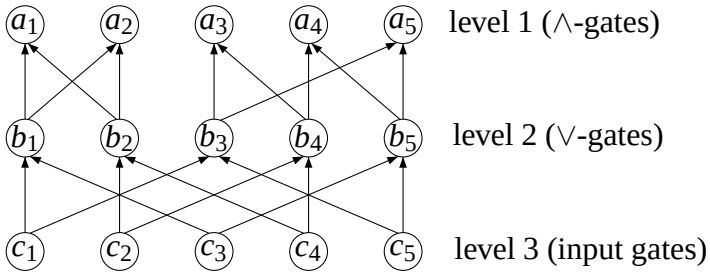

Fig. 5.1 An SAM2-circuit

Theorem 23. *Path checking for* MTL_b *over infinite monotonic binary encoded data words of the form* $(u)^{\omega}_{+k}$ *is in* $\mathsf{AC}^1(\mathsf{logDCFL})$.

Proof. Given an infinite monotonic data word $(u)^{\omega}_{+k}$ and an MTL-formula φ where all input numbers are encoded in binary notation, the finite data word $(u)^{3\mathsf{Rank}(\varphi)+1}_{+k}$ and the formula $\overline{\varphi}$ are computable in logarithmic space. By Proposition 13, we know that $(u)^{\omega}_{+k} \models \varphi$ if and only if $(u)^{3\mathsf{Rank}(\varphi)+1}_{+k} \models \overline{\varphi}$. Since path checking for MTL_b over finite monotonic binary encoded data word is in $\mathsf{AC}^1(\mathsf{logDCFL})$ [21], and $\mathsf{L} \subseteq \mathsf{AC}^1 \subseteq \mathsf{AC}^1(\mathsf{logDCFL})$, the theorem follows. □

5.2 The lower complexity bounds

In this section, we will prove several P-hardness and PSPACE-hardness results for path checking problems. All lower bounds also hold for non-pure and non-monotonic data words (and we will not mention this explicitly in the theorems). But we have to distinguish (i) whether data words are unary or binary encoded, and (ii) whether data words are finite or infinite. And all lower bounds that hold for unary (respectively, finite) data words also hold for binary (respectively, infinite) data words.

5.2.1 P-hardness for MTL and TPTL[1]

We prove our P-hardness results by a reduction from a restricted version of the Boolean circuit value problem. A *Boolean circuit* is a finite oriented directed acyclic graph, where each node is called a gate. An *input* gate is a node with indegree 0. All other gates are labeled with one of $\vee$, $\wedge$ or $\neg$ (the logical OR, AND and NOT operations). An *output* gate is a node with outdegree 0. A Boolean circuit is *monotone* if it does not have $\neg$ gates.

Definition 12. A synchronous alternating monotone circuit with fanin 2 and fanout 2 (briefly, SAM2-circuit) is a monotone circuit divided into levels $1,\ldots,l$ $(l \geq 2)$ such that the following properties hold:

- All wires go from a gate in level $i+1$ to a gate from level i ($1 \leq i < l$).
- All output gates are in level 1 and all input gates are in level l.
- All gates in the same level are of the same type ($\wedge$, $\vee$ or input) and the levels alternate between $\wedge$-levels and $\vee$-levels.
- All gates except the output gates have outdegree 2 and all gates except the input gates have indegree 2. The two input gates for a gate at level $i < l$ are different.

By the restriction to fanin 2 and fanout 2, we know that each level contains the same number of gates. Fig. 5.1 shows an example of an SAM2-circuit (the node names a_i, b_i, c_i will be needed later). The circuit value problem for SAM2-circuits, called SAM2CVP in [44], is the following problem:

Input: An SAM2-circuit α, inputs $x_1, \ldots, x_n \in \{0,1\}$, and a designated output gate y.

Output: `yes` if output y of α evaluates to 1 on inputs $x_1, \ldots, x_n$, `no` otherwise.

It is shown in [44] that SAM2CVP is P-complete.

Recall that path checking for MTL over finite monotonic data words is in the parallel complexity class $\mathsf{AC}^1(\mathsf{LogDCFL})$ [21]. We will show that for both (i) MTL_u over non-monotonic data words and (ii) TPTL_u^1 over monotonic data words the path checking problem becomes P-hard (and hence P-complete). Actually, we prove the results for their pure unary fragments, where the "pure unary" means that we do not use any propositions and use only the unary modalities X, F and G instead of U in the formula.

Theorem 24. *Path checking for* $\mathsf{pureUnaMTL}_u$ *over finite unary encoded pure data words is* P*-hard.*

Proof. We reduce from SAM2CVP. Let α be the input circuit. We first encode each two consecutive levels of α into a data word, and combine these data words into a data word w, which is the encoding of the whole circuit. Then we construct a $\mathsf{pureUnaMTL}_u$-formula ψ such that $w \models \psi$ if and only if α evaluates to 1. The data word w that we are constructing contains gate names of α (and some copies of the gates) as atomic propositions. These propositions will be only needed for the construction. At the end, we can remove all propositions from the data word w and hence obtain a pure data word. The whole construction can be done in logarithmic space. The reader might look at Example 6, where the construction is carried out for the circuit from Fig. 5.1.

Let α be an SAM2-circuit with $l \geq 2$ levels and n gates in each level. By the restriction to fanin 2 and fanout 2 we know that the induced undirected subgraph which contains the nodes

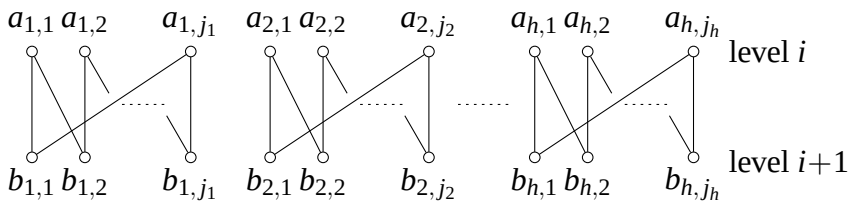

Fig. 5.2 The induced subgraph between level i and $i+1$

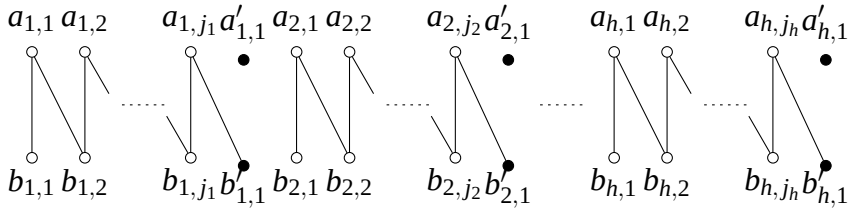

Fig. 5.3 The graph obtained from the induced subgraph

in level i and level $i+1$ $(1 \leq i < l)$ is comprised of several cycles; see Fig. 5.2. For instance, for the circuit in Fig. 5.1 the number of cycles between level 1 and 2 (respectively, level 2 and 3) is 2.

We can enumerate in logarithmic space the gates of level i and level $i+1$ such that they occur in the order shown in Fig. 5.2. To see this, let $a_1, \ldots, a_n$ (respectively, $b_1, \ldots, b_n$) be the nodes in level i (respectively, $i+1$) in the order in which they occur in the input description. We start with a_1 and enumerate the nodes in the cycle containing a_1 (from a_1 we go to the smaller neighbor among $b_1, \ldots, b_n$, then the next node on the cycle is uniquely determined since the graph has degree 2). Thereby we store the current node in the cycle and the starting node a_1. As soon as we come back to a_1, we know that the first cycle is completed. To find the next cycle, we search for the first node from the list $a_2, \ldots, a_n$ that is not reachable from a_1 (reachability in undirected graphs is in LOGSPACE [71]), and continue in this way.

So, assume that the nodes in level i and $i+1$ are ordered as in Fig. 5.2. In particular, we have h cycles. For each $1 \leq t \leq h$, we add a new node $a'_{t,1}$ (respectively, $b'_{t,1}$) after a_{t,j_t} (respectively, b_{t,j_t}). Then we replace the edge $(a_{t,j_t}, b_{t,1})$ by the edge $(a_{t,j_t}, b'_{t,1})$ $(1 \leq t \leq h)$. In this way we obtain the graph from Fig. 5.3. Again, the construction can be done in logarithmic space by adding the new nodes and new edges once a cycle was completed in the enumeration procedure from the previous paragraph.

By adding dummy nodes, we can assume that for every $1 \leq i \leq l-1$, the subgraph between level i and $i+1$ has the same number (say h) of cycles (this is only done for notational convenience, and we still suppose that there are n gates in each level). Thus, after the above step

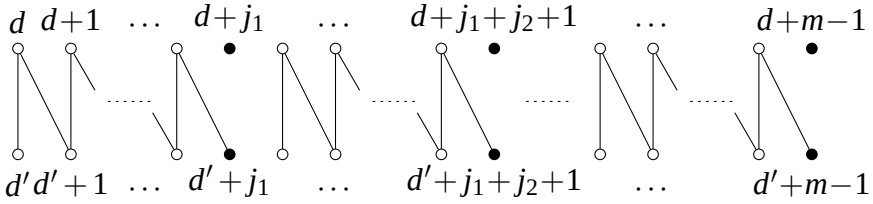

Fig. 5.4 Labeling the new graph

we have $m = n+h$ many nodes in each level. Let $d = (i-1)\cdot 2m$ and $d' = d+m$. In Fig. 5.3, we label the nodes in level i (respectively, $i+1$) with the numbers $d, d+1, \ldots, d+m-1$ (respectively $d', d'+1 \ldots, d'+m-1$) in this order, see Fig. 5.4. By this labeling, the difference between two connected nodes in level i and level $i+1$ is always m or $m+1$. So we can use the modality $\mathsf{F}_{[m,m+1]}$ (respectively, $\mathsf{G}_{[m,m+1]}$) to jump from an $\vee$-gate (respectively, $\wedge$-gate) in level i to a successor gate in level $i+1$. We now obtain in logarithmic space the data word $w_i = w_{i,1}w_{i,2}$, where

$$w_{i,1} = \begin{cases} (a_{1,1},d)(a_{1,2},d+1)\cdots(a_{1,j_1},d+j_1-1) \\ (a_{2,1},d+j_1+1)(a_{2,2},d+j_1+2)\cdots(a_{2,j_2},d+j_1+j_2)\cdots \\ (a_{h,1},d+\sum_{t=1}^{h-1} j_t + h - 1)(a_{h,2},d+\sum_{t=1}^{h-1} j_t + h)\cdots(a_{h,j_h},d+m-2) \end{cases}$$

$$w_{i,2} = \begin{cases} (b_{1,1},d')\cdots(b_{1,j_1},d'+j_1-1)(b'_{1,1},d'+j_1) \\ (b_{2,1},d'+j_1+1)\cdots(b_{2,j_2},d'+j_1+j_2)(b'_{2,1},d'+j_1+j_2+1)\cdots \\ (b_{h,1},d'+\sum_{t=1}^{h-1} j_t + h - 1)\cdots(b_{h,j_h},d'+m-2)(b'_{h,1},d'+m-1) \end{cases}$$

which is the encoding of the wires between level i and level $i+1$ from Fig. 5.4. Note that the new nodes $a'_{1,1}, a'_{2,1}, \ldots, a'_{h,1}$ in level i of the graph in Fig. 5.3 do not occur in $w_{i,1}$.

Suppose now that all data words w_i ($1 \leq i \leq l-1$) are constructed. We then combine them to obtain the data word w for the whole circuit as follows. Suppose that

$$w_{i,2} = (\tilde{b}_1, y_1)\cdots(\tilde{b}_m, y_m) \text{ and } w_{i+1,1} = (b_1, z_1)\cdots(b_n, z_n).$$

Note that every $\tilde{b}_s$ is either one of the b_j or b'_j (the copy of b_j). Let

$$v_{i+1,1} = (\tilde{b}_1, z'_1)\cdots(\tilde{b}_m, z'_m),$$

where the data values z'_s are determined as follows: If $\tilde{b}_s = b_j$ or $\tilde{b}_s = b'_j$, then $z'_s = z_j$. Then, the data word w is $w = w_{1,1}w_{1,2}v_{2,1}w_{2,2}\cdots v_{l-1,1}w_{l-1,2}$.

Let us explain the idea. Consider a gate a_j of level $2 \leq i \leq l-1$, and assume that level i consists of $\vee$-gates. Let b_{j_1} and b_{j_2} (from level $i+1$) be the two input gates for a_j. In the above data word $v_{i,1}$ there is a unique position where the proposition a_j occurs, and possibly a position where the copy a'_j occurs. If both positions exist, then they carry the same data value. Let us point to one of these positions. Using an MTL-formula, we want to branch (existentially) to the positions in the factor $v_{i+1,1}$, where the propositions $b_{j_1}, b'_{j_1}, b_{j_2}, b'_{j_2}$ occur (where b'_{j_1} and b'_{j_2} possibly do not exist). For this, we use the modality $\mathsf{F}_{[m,m+1]}$. By construction, this modality branches existentially to positions in the factor $w_{i,2}$, where the propositions $b_{j_1}, b'_{j_1}, b_{j_2}, b'_{j_2}$ occur. Then, using the iterated modality X^m (which is an abbreviation for m copies of the MTL-modality $X_{\mathbb{Z}}$), we jump to the corresponding positions in $v_{i+1,1}$.

In the above argument, we assumed that $2 \leq i \leq l-1$. If $i = 1$, then we can argue similarly, if we assume that we are pointing to the unique a_j-labeled position of the prefix $w_{1,1}$ of w. Now consider level $l-1$. Suppose that

$$w_{l-1,2} = (\tilde{d}_1, v_1)\ldots(\tilde{d}_m, v_m).$$

Let $d_1, \ldots, d_n$ be the original gates of level l, which all belong to $\{\tilde{d}_1, \ldots, \tilde{d}_m\}$, and let $x_i \in \{0,1\}$ be the input value for gate d_i. Define

$$I = \{ j \mid j \in [1,m], \exists i \in [1,n] : \tilde{d}_j \in \{d_i, d'_i\}, x_i = 1\}. \tag{5.7}$$

Let the designated output gate be the k^{th} node in level 1. We construct the pureUnaMTL-formula $\psi = \mathsf{X}^{k-1}\varphi_1$, where φ_i ($1 \leq i \leq l-1$) is defined inductively as follows:

$$\varphi_i = \begin{cases} \mathsf{F}_{[m,m+1]}\mathsf{X}^m\varphi_{i+1} & \text{if } i < l-1 \text{ and level } i \text{ is a } \vee\text{-level,} \\ \mathsf{G}_{[m,m+1]}\mathsf{X}^m\varphi_{i+1} & \text{if } i < l-1 \text{ and level } i \text{ is a } \wedge\text{-level,} \\ \mathsf{F}_{[m,m+1]}(\bigvee_{j\in I}\mathsf{X}^{m-j}\neg\mathsf{X}\top) & \text{if } i = l-1 \text{ and level } i \text{ is a } \vee\text{-level,} \\ \mathsf{G}_{[m,m+1]}(\bigvee_{j\in I}\mathsf{X}^{m-j}\neg\mathsf{X}\top) & \text{if } i = l-1 \text{ and level } i \text{ is a } \wedge\text{-level.} \end{cases}$$

The formula $\neg\mathsf{X}\top$ is only `true` in the last position of a data word. Suppose data word w is the encoding of the circuit. From the above consideration, it follows that $w \models \psi$ if and only if the circuit α evaluates to 1. Note that we only use the unary modalities $\mathsf{F}, \mathsf{G}, \mathsf{X}$ and do not use any propositions in ψ. So we can ignore the propositional part in the data word w to get a pure data word. Since the number m is bounded by $2n$, and all data values in w are bounded by $4nl$,

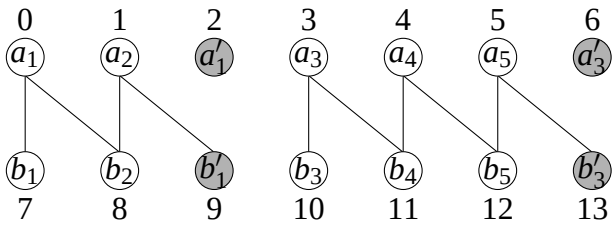

Fig. 5.5 The labeling for level 1 and 2

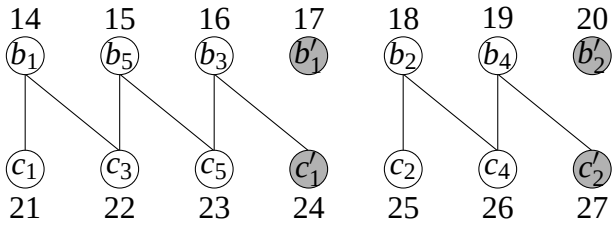

Fig. 5.6 The labeling for level 2 and 3

where n is the number of gates in each level and l is the number of levels. We can compute the formula ψ and data word w where the interval borders and data values are encoded in unary notation in logarithmic space. □

Corollary 14. *Path checking for* MTL_u *over finite unary encoded data words is* P*-hard.*

Example 6. Let α be the SAM2-circuit from Fig. 5.1. It has 3 levels and 5 gates in each level. Level 1 contains $\wedge$-gates and level 2 contains $\vee$-gates. There are 2 cycles in the subgraph between level 1 and 2, and also 2 cycles in the subgraph between level 2 and 3. The encoding for level 1 and level 2 is

$$\begin{aligned}(a_1,0)(a_2,1)(a_3,3)(a_4,4)(a_5,5)\\ (b_1,7)(b_2,8)(b'_1,9)(b_3,10)(b_4,11)(b_5,12)(b'_3,13),\end{aligned} \tag{5.8}$$

which can be obtained from Fig. 5.5. The new nodes a'_1 and a'_3 in level 1 are not used for the final encoding in the data word. The encoding for level 2 and 3 is

$$\begin{aligned}(b_1,14)(b_5,15)(b_3,16)(b_2,18)(b_4,19)\\ (c_1,21)(c_3,22)(c_5,23)(c'_1,24)(c_2,25)(c_4,26)(c'_2,27),\end{aligned} \tag{5.9}$$

which can be obtained from Fig. 5.6. We skip the new nodes b'_1 and b'_2 in level 2 in this encoding.

We combine (5.8) and (5.9) to obtain the following data word (5.10) which is the encoding of the circuit α. The encoding for level 1 and 2 determines the order of the propositional part

of the third line in (5.10), and the encoding for level 2 and 3 determines its data values.

$$\begin{aligned}(a_1,0)(a_2,1)(a_3,3)(a_4,4)(a_5,5)\\ (b_1,7)(b_2,8),(b_1',9)(b_3,10)(b_4,11)(b_5,12)(b_3',13)\\ (b_1,14)(b_2,18)(b_1',14)(b_3,16)(b_4,19)(b_5,15)(b_3',16)\\ (c_1,21)(c_3,22)(c_5,23)(c_1',24)(c_2,25)(c_4,26)(c_2',27).\end{aligned} \tag{5.10}$$

Let the designated output gate be a_3 in level 1, and assume that the input gates c_1, c_4, c_5 (respectively, c_2, c_3) receive the value 0 (respectively, 1). Then the set I from (5.7) is $I = \{2,5,7\}$ and the formula ψ is

$$\psi = \mathsf{X}^2(\mathsf{G}_{[7,8]}\mathsf{X}^7(F_{[7,8]}(\bigvee_{j\in\{2,5,7\}} \mathsf{X}^{7-j}\neg\mathsf{X}\top))).$$

We extend Theorem 24 to path checking for MTL_u over infinite data words in the following.

Theorem 25. *Path checking for* unaMTL_u *over infinite unary encoded data words is* P*-hard.*

Proof. The proof is adapted from the proof of Theorem 24. In that proof, let p be an atomic proposition that is not used in the data word w. Define the infinite data word $w' = w(p,5ml)^{\omega}_{+0}$, and redefine the formula φ_i by:

$$\varphi_i = \begin{cases} \mathsf{F}_{[m,m+1]}\mathsf{X}^m\varphi_{i+1} & \text{if } i < l-1 \text{ and level } i \text{ is a } \vee\text{-level,}\\ \mathsf{G}_{[m,m+1]}\mathsf{X}^m\varphi_{i+1} & \text{if } i < l-1 \text{ and level } i \text{ is a } \wedge\text{-level,}\\ \mathsf{F}_{[m,m+1]}(\bigvee_{j\in I}\mathsf{X}^{m-j}(\neg p\wedge\mathsf{X}p)) & \text{if } i = l-1 \text{ and level } i \text{ is a } \vee\text{-level,}\\ \mathsf{G}_{[m,m+1]}(\bigvee_{j\in I}\mathsf{X}^{m-j}(\neg p\wedge\mathsf{X}p)) & \text{if } i = l-1 \text{ and level } i \text{ is a } \wedge\text{-level.}\end{cases}$$

It is easily seen that $w' \models \psi$ if and only if the circuit α evaluates to 1. □

Corollary 15. *Path checking for* MTL_u *over infinite unary encoded data words is* P*-hard.*

Note that the construction in the proof of Theorem 24 uses non-monotonic data words. This is unavoidable since it was shown in [21] that path checking for MTL over finite monotonic data words belongs to $\mathsf{AC}^1(\mathsf{LogDCFL})$. But if we consider a succinct version of MTL that still has the same expressive power, the data words constructed can be monotonic.

Our next P-hardness result will be shown for monotonic data words. First we define an extension of MTL.

Definition 13. In the definition of MTL, if we replace the modality U_I by $\mathsf{U}_{I_1 \cup I_2 \cup \cdots \cup I_n}$, where $I_1 \cup I_2 \cup \cdots \cup I_n$ a finite union of intervals $I_i \subseteq \mathbb{Z} (1 \leq i \leq n)$, then we call this logic succinct MTL (SMTL).

Formally, the syntax and the semantics of SMTL are the same as that for MTL, except that the set I in U_I can be a finite union $I = I_1 \cup I_2 \cup \cdots \cup I_n$ of intervals $I_i \subseteq \mathbb{Z}$. We use unaSMTL to denote the unary fragment of SMTL that uses only the unary modalities $\mathsf{X}, \mathsf{F}, \mathsf{G}$, and use pureUnaSMTL to denote the pure fragment of unaSMTL, i.e., there are no propositions are used in the formula.

Let $I = \bigcup_{i=1}^{n} I_i$. It is easily seen that

$$\varphi_1 \mathsf{U}_I \varphi_2 \equiv \bigvee_{i=1}^{n} \varphi_1 \mathsf{U}_{I_i} \varphi_2 \equiv x.\varphi_1 \mathsf{U}((\bigvee_{i=1}^{n} x \in I_i) \wedge \varphi_2).$$

We have the following two facts:

Fact 1. *Every* SMTL*-formula is equivalent to an* MTL*-formula which can be exponentially larger.*

Fact 2. *Every* SMTL*-formula is equivalent to a* TPTL^1*-formula of polynomial size, which, moreover, can be computed in logarithmic space.*

By Fact 2 and Theorem 22, we can know that path checking for SMTL_b over infinite binary encoded data words is in P. In the following we show that path checking for SMTL_u over finite unary encoded data words is P-hard.

Theorem 26. *Path checking for* pureUnaSMTL$_u$ *over finite unary encoded strictly monotonic pure data words is* P*-hard.*

Proof. We reduce from SAM2CVP. Let α be an SAM2-circuit with $l \geq 2$ levels and n gates in each level. The idea will be to encode the wires between two consecutive levels by a suitably shifted version of the data word

$$w_n = \prod_{i=1}^{n} i \cdot \prod_{i=1}^{n} i(n+1) = (1, 2, \ldots, n, \, 1 \cdot (n+1), 2 \cdot (n+1), \ldots, n \cdot (n+1)).$$

Note that for all $i_1, i_2 \in \{1, \ldots, n\}$ and $j_1, j_2 \in \{1 \cdot (n+1), 2 \cdot (n+1), \ldots, n \cdot (n+1)\}$, we have the following: If $j_1 - i_1 = j_2 - i_2$ then $i_1 = i_2$ and $j_1 = j_2$. This is best seen by viewing numbers in their base $(n+1)$ expansion. Let us denote with $\Delta = n(n+1) - 1$ the maximal difference between a number from $\{1, \ldots, n\}$ and a number from $\{1 \cdot (n+1), 2 \cdot (n+1), \ldots, n \cdot (n+1)\}$.

We define the pure and strictly monotonic data word $w_{n,l}$ as

$$w_{n,l} = \prod_{j=0}^{l-2} (w_n)_{+j\cdot n(n+2)}.$$

The offset number $j \cdot n(n+2)$ is chosen such that the difference between a number from $\{1+j\cdot n(n+2),\ldots,n+j\cdot n(n+2)\}$ and a number from $\{1+(j+1)\cdot n(n+2),\ldots,n+(j+1)\cdot n(n+2)\}$ is larger than Δ for every $j \geq 0$.

Note that all data values in $w_{n,l}$ are bounded by $(l+1)n(n+2)$. The unary encoding of the data word $w_{n,l}$ is computable in logarithmic space from the circuit. For each $1 \leq j < l$, define

$$S_j = \{i_2(n+1) - i_1 \mid \text{ the } i_1^{\text{th}} \text{ gate in level } j \text{ connects to the } i_2^{\text{th}} \text{ gate in level } j+1\}.$$

Suppose $o_k\,(1 \leq k \leq n)$ is the designated output gate. Let I be the set of all $i \in [1,n]$ such that the i^{th} gate in level l is set to the Boolean value 1. We construct the pureUnaSMTL-formula $\psi = \mathsf{X}^{k-1}\varphi_1$, where φ_j $(1 \leq j \leq l-1)$ is defined inductively as follows:

$$\varphi_j = \begin{cases} \mathsf{F}_{S_j}\mathsf{X}^n\varphi_{j+1} & \text{if } j < l-1 \text{ and level } j \text{ is a } \vee\text{-level,} \\ \mathsf{G}_{S_j}\mathsf{X}^n\varphi_{j+1} & \text{if } j < l-1 \text{ and level } j \text{ is a } \wedge\text{-level,} \\ \mathsf{F}_{S_j}(\bigvee_{i\in I}\mathsf{X}^{n-i}\neg\mathsf{X}\top) & \text{if } j = l-1 \text{ and level } j \text{ is a } \vee\text{-level,} \\ \mathsf{G}_{S_j}(\bigvee_{i\in I}\mathsf{X}^{n-i}\neg\mathsf{X}\top) & \text{if } j = l-1 \text{ and level } j \text{ is a } \wedge\text{-level.} \end{cases}$$

The purpose of the prefix X^n in front of φ_{j+1} is to move from a certain position within the second half of the j^{th} copy of w_n to the corresponding position within the first half of the $(j+1)^{\text{th}}$ copy of w_n in $w_{n,l}$. Note that no propositions are used, and only the unary modalities $\mathsf{F},\mathsf{G},\mathsf{X}$ are used in ψ. It is straightforward to check that $w_{n,l} \models \psi$ if and only if the circuit α evaluates to 1. □

Corollary 16. *Path checking for* SMTL_u *over finite unary encoded data words is* P*-hard.*

By Fact 2, we can obtain the following corollaries.

Corollary 17. *Path checking for* $\mathsf{pureUnaTPTL}_u^1$ *over finite unary encoded strictly monotonic pure data words is* P*-hard.*

Corollary 18. *Path checking for* TPTL_u^1 *over finite unary encoded data words is* P*-hard.*

Example 7. Let α be the SAM2-circuit from Fig. 5.1. The encoding for level 1 and level 2 is

$$(1,2,3,4,5,\ 6,12,18,24,30). \tag{5.11}$$

The encoding for level 2 and level 3 is

$$(36,37,38,39,40,\ 41,47,53,59,65). \tag{5.12}$$

We concatenate (5.11) and (5.12) to obtain the following data word $w_{5,3}$ which is the encoding of the circuit α:

$$w_{5,3} = (1,2,3,4,5,\ 6,12,18,24,30)(36,37,38,39,40,\ 41,47,53,59,65).$$

Define

$$S_1 = \{4,5,10,11,13,15,20,21,25,26\}$$

and

$$S_2 = \{3,5,8,10,13,17,20,22,25,27\}.$$

Let the designated output gate be a_3 in level 1, and assume that the input gates c_1, c_4, c_5 (respectively, c_2, c_3) receive the value 0 (respectively, 1). Then the set I is $I = \{2,3\}$ and the pureUnaSMTL-formula ψ is

$$\psi = \mathsf{X}^2(\mathsf{G}_{S_1}\mathsf{X}^5(F_{S_2}(\bigvee_{j\in\{2,3\}} \mathsf{X}^{5-j}\neg\mathsf{X}\top))).$$

Similar to the proof of Theorem 25, we can adapt the proof of Theorem 26 and extend the results to infinite data words.

Theorem 27. *Path checking for* $\mathsf{unaSMTL}_u$ *and* $\mathsf{unaTPTL}_u^1$ *over infinite unary encoded data words are* P*-hard.*

Corollary 19. *Path checking for* SMTL_u *and* TPTL_u^1 *over infinite unary encoded data words are* P*-hard.*

5.2.2 PSPACE-hardness for TPTL

In this subsection, we will prove several PSPACE lower bounds for TPTL and TPTL^r $(r \geq 2)$. For TPTL and TPTL_b^r $(r \geq 2)$, we prove the lower bounds for their pure unary fragments.

Theorem 28. *Path checking for* $\mathsf{pureUnaTPTL}_u$ *over finite unary encoded strictly monotonic pure data words is* PSPACE*-hard.*

Proof. We prove PSPACE-hardness by a reduction from the PSPACE-complete quantified Boolean formula problem (QBF, for short). Let $\Psi = Q_1 x_1 \cdots Q_n x_n \phi$ be a quantified Boolean formula, where $Q_i \in \{\forall, \exists\}$ and ϕ is a quantifier-free propositional formula. We construct the finite pure strictly monotonic data word

$$w = 0, 1, 2, \ldots, 2n-1, 2n, 2n+1.$$

For every $i \in \{1, \ldots, n\}$, the subword $2i-1, 2i$ is used to quantify over the Boolean variable x_i. We use a corresponding register variable x_i. If we assign to this register variable the data value $2i-b$, then the corresponding Boolean variable x_i is set to $b \in \{0,1\}$.

We define the pureUnaTPTL-formula $x.\Psi'$, where Ψ' is defined inductively by the following rules.

- If $\Psi = \forall x_i \Phi$, then $\Psi' = \mathsf{G}((x_i = 2i-1 \vee x_i = 2i) \rightarrow x_i.\Phi')$.
- If $\Psi = \exists x_i \Phi$, then $\Psi' = \mathsf{F}((x_i = 2i-1 \vee x_i = 2i) \wedge x_i.\Phi')$.
- If Ψ is a quantifier-free formula, then

  $$\Psi' = \mathsf{F}(x = 2n+1 \wedge \Psi[x_1/x_1 = 2n, \ldots, x_i/x_i = 2(n-i)+2, \ldots, x_n/x_n = 2]).$$

  Here, $\Psi[x_1/x_1 = a_0, \ldots, x_n/x_n = a_n]$ denotes the TPTL-formula obtained from Ψ by replacing every occurrence of x_i by $x_i = a_i$ $(1 \leq i \leq n)$.

Recall from the semantics of TPTL that the subformula $x_i = 2i-1 \vee x_i = 2i$ is `true` if and only if the difference between the current data value and the value to which x_i is bound (which is initially 0) is $2i-1$ or $2i$. Hence, the subformula is only `true` at the two positions where the data values are $2i-1$ and $2i$, respectively. It is easy now to see that the quantified Boolean formula Ψ is `true` if and only if $w \models x.\Psi'$. □

Corollary 20. *Path checking for* TPTL_u *over finite unary encoded data words is* PSPACE*-hard.*

It is shown that model checking for freezeLTL over one-deterministic counter machine is PSPACE-hard [30]. Since every infinite computation of a deterministic one-counter machine is of the form $u_1(u_2)^{\omega}_{+k}$ (see Section 5.4). We can know that path checking for TPTL over infinite data words is PSPACE-hard. Moreover, we can show that the path checking is still PSPACE-hard even over the infinite strictly monotonic pure data word $(0)^{\omega}_{+1} = 0, 1, 2, 3, 4, \ldots$

Theorem 29. *Path checking for* TPTL_u *over the infinite data word* $(0)^{\omega}_{+1}$ *is* PSPACE*-hard.*

Proof. In the proof of Theorem 28, let $w' = w(2n+2)^{\omega}_{+1} = 0,1,2,3,4,\ldots$ Analysis similar to that proof shows that the quantified Boolean formula Ψ is `true` if and only if $w' \models x.\Psi'$. □

Corollary 21. *Path checking for* TPTL_u *over infinite unary encoded data words is* PSPACE*-hard.*

In the following we consider path checking for the fragment of TPTL where the number of register variables are bounded by a fixed number $r \geq 2$.

The quantified subset sum problem (QSS) is PSPACE-complete [79]:

Input: A sequence $a_1, a_2, \ldots, a_{2n}, b \in \mathbb{N}$ of binary encoded numbers.

Output: `yes` if $\forall x_1 \in \{0, a_1\} \exists x_2 \in \{0, a_2\} \cdots \forall x_{2n-1} \in \{0, a_{2n-1}\} \exists x_{2n} \in \{0, a_{2n}\}$ such that $\sum_{i=1}^{2n} x_i = b$, `no` otherwise.

We define a variant of QSS in the following:

Input: A sequence $a_1, a_2, \ldots, a_{2n}, b \in \mathbb{N} \setminus \{0\}$ of binary encoded numbers.

Output: `yes` if $\forall x_1 \in \{1, a_1\} \exists x_2 \in \{1, a_2\} \cdots \forall x_{2n-1} \in \{1, a_{2n-1}\} \exists x_{2n} \in \{1, a_{2n}\}$ such that $\sum_{i=1}^{2n} x_i = b$, `no` otherwise.

We call this problem positive quantified subset sum problem (PQSS). PQSS is also PSPACE-complete. It is easy to check that for every instance $a_1, a_2, \ldots, a_{2n}, b$ of QSS, the answer is `yes` for this input if and only if the answer for the PQSS-input $(a_1+1, a_2+1, \ldots, a_{2n}+1, b+2n)$ is `yes`.

Theorem 30. *Path checking for* $\mathsf{pureUnaTPTL}^2_b$ *over the infinite data word* $(0)^{\omega}_{+1}$ *is* PSPACE*-hard.*

Proof. The theorem is proved by a reduction from PQSS. Given an instance $a_1, a_2, \ldots, a_{2n}, b$ of PQSS, we construct the $\mathsf{pureUnaTPTL}^2$-formula $x.\varphi_1$, where the formula $\varphi_i\,(1 \leq i \leq 2n+1)$ is defined inductively by

$$\varphi_i = \begin{cases} y.\mathsf{G}((y = 1 \vee y = a_i) \rightarrow \varphi_{i+1}) & \text{for } i < 2n \text{ odd}, \\ y.\mathsf{F}((y = 1 \vee y = a_i) \wedge \varphi_{i+1}) & \text{for } i \leq 2n \text{ even}, \\ x = b & \text{for } i = 2n+1. \end{cases}$$

The intuition is the following: Note that in the data word w the data value is increasing by one in each step. Assume we want to evaluate $y.\mathsf{G}((y = 1 \vee y = a_i) \rightarrow \varphi_{i+1})$ in a position where the data value is currently d. The initial freeze quantifier sets y to d. Then, $\mathsf{G}((y = 1 \vee y =$

$a_i) \to \varphi_{i+1})$ means that in every future position, where the current data value is either $d+1$ (in such a position $y=1$ holds by the TPTL-semantics) or $d+a_i$ (in such a position $y=a_i$ holds), the formula φ_{i+1} has to hold. In this way, we simulate the quantifier $\forall x_i \in \{1, a_i\}$. At the end, we have to check that the current data value is b, which can be done with the constraint $x=b$ (note that x is initially set to 0 and never reset). We can show that $(0)^{\omega}_{+1} \models x.\varphi_1$ if and only if the answer for the PQSS-input $(a_1, a_2, \ldots, a_{2n}, b)$ is `yes`. □

Corollary 22. *Path checking for* TPTL^2_b *over infinite unary encoded strictly monotonic pure data words is* PSPACE*-hard.*

Recall from Theorem 20 that for every fixed r, path checking for TPTL^r_u over infinite binary encoded quasi-monotonic data words can be solved in polynomial time. The following result shows that quasi-monotonicity is important for Theorem 20. First we prove a lower bound for $\mathsf{freezeLTL}^2$, which is a fragment of TPTL^2_u.

Theorem 31. *Path checking for* $\mathsf{freezeLTL}^2$ *over infinite binary encoded pure data words is* PSPACE*-hard.*

Proof. The theorem is proved by a reduction from PQSS.

We first prove the theorem for non-pure data words, and then show how to get rid of the atomic propositions. Given an instance $a_1, a_2, \ldots, a_{2n}, b$ of PQSS, we construct the infinite data word

$$w = (\emptyset, b)\left((q,0)\prod_{i=1}^{2n}(p,1)(p,a_i)(\emptyset,0)\right)^{\omega}_{+1}$$

and the formula $x.y.\mathsf{X}\varphi_1$, where the formulas φ_i $(1 \le i \le 2n+1)$ are defined as follows. First of all, for a proposition p and a formula ψ we use the abbreviations

$$\mathsf{F}_p\psi = p\mathsf{U}(p \wedge \psi) \text{ and } \mathsf{G}_p\psi = \neg\mathsf{F}_p\neg\psi.$$

Thus, $\mathsf{F}_p\psi$ holds in position j means that in the future there is a time point t such that ψ holds at time t and the proposition p holds at all time points $j < s \le t$. Similarly, $\mathsf{G}_p\psi$ holds in position j means that for all future time points t such that p holds at all time points $j < s \le t$, ψ holds at t. Then we define:

$$\varphi_i = \begin{cases} \mathsf{X}^{3(i-1)}\mathsf{G}_p y.\mathsf{F}(q \wedge y = 0 \wedge \varphi_{i+1}) & \text{for } i < 2n \text{ odd}, \\ \mathsf{X}^{3(i-1)}\mathsf{F}_p y.\mathsf{F}(q \wedge y = 0 \wedge \varphi_{i+1}) & \text{for } i \le 2n \text{ even}, \\ x = 0 & \text{for } i = 2n+1. \end{cases}$$

Let us explain the formula $\mathsf{X}^{3(i-1)}\mathsf{F}_p y.\mathsf{F}(q \wedge y = 0 \wedge \varphi_{i+1})$. We will only evaluate this formula in positions where the proposition q holds (i.e., the starting positions of the periodic part of w). Let d be the data value at this position (meaning that we are at the first position of the $(d+1)^{\text{th}}$ iteration of the periodic part). With $\mathsf{X}^{3(i-1)}$ we move to the position j which precedes the block $(p,1)(p,a_i)(\emptyset,0)$. The data value at the next position $j+1$ is $d+1$, whereas the data value at the position $j+2$ is $d+a_i$. With the modality F_p we either move to the position $j+1$ or move to the position $j+2$; the choice is made existentially (if the modality is G_p, then the choice is made universally). Next, y is set to the current data value. Hence, we existentially set y to either $d+1$ or $d+a_i$. With the final part $\mathsf{F}(q \wedge y = 0 \wedge \varphi_{i+1})$ we go to the unique position in the future, where q holds and the data value at this position is equal to the value which was assigned to y before ($d+1$ or $d+a_i$). In this way we simulate the quantifier $\exists x_i \in \{1, a_i\}$.

Finally, note that initially, the register variable x is set to b (the data value at the first position of w). Hence, in the formula φ_{2n} we express by the constraint $x = 0$ that the current data value has to be b. This shows that $w \models x.y.\mathsf{X}\varphi_1$ if and only if the instance $a_1, a_2, \ldots, a_{2n}, b$ is positive.

We can get rid of the propositions p and q by encoding them into a pure data word. We use $(0,1,1)$ (respectively, $(0,0,0)$) to denote q (respectively, p). Then the data word w can be replaced by the following pure data word (for better readability we underline the positions that correspond to the old data word w)

$$w' = b\left(0,1,1,\underline{0},\prod_{i=1}^{2n}(0,0,0,\underline{1},0,0,0,\underline{a_i},0)\right)^{\omega}_{+1}.$$

Define $\varphi_q = x.\mathsf{X}(\neg(x=0) \wedge x.\mathsf{X}(x=0))$ and $\varphi_p = x.\mathsf{X}(x = 0 \wedge \mathsf{X}(x=0))$. We replace the formula $\mathsf{F}_p\psi$ by

$$\mathsf{F}'_p\psi = [\varphi_p \vee \mathsf{X}^3(\varphi_p \wedge \mathsf{X}\neg\varphi_p) \vee \mathsf{X}^2(\varphi_p \wedge \mathsf{X}\neg\varphi_p) \vee \mathsf{X}(\varphi_p \wedge \mathsf{X}\neg\varphi_p)]\mathsf{U}[\varphi_p \wedge \psi]$$

and define $\mathsf{G}'_p\psi = \neg\mathsf{F}'_p\neg\psi$. Then we define:

$$\varphi'_i = \begin{cases} \mathsf{X}^{9(i-1)}\,\mathsf{G}'_p\mathsf{X}^3 y.\mathsf{F}(\varphi_q \wedge \mathsf{X}^4\varphi_p \wedge \mathsf{X}^3(y=0) \wedge \mathsf{X}^3\varphi'_{i+1}) & \text{for } i < 2n \text{ odd,} \\ \mathsf{X}^{9(i-1)}\,\mathsf{F}'_p\mathsf{X}^3 y.\mathsf{F}(\varphi_q \wedge \mathsf{X}^4\varphi_p \wedge \mathsf{X}^3(y=0) \wedge \mathsf{X}^3\varphi'_{i+1}) & \text{for } i \leq 2n \text{ even,} \\ x = 0 & \text{for } i = 2n+1. \end{cases}$$

Analysis similar to above shows that $w' \models x.y.\mathsf{X}^4\varphi'_1$ if and only if the instance $a_1, a_2, \ldots, a_{2n}, b$ is positive. □

Corollary 23. *Path checking for* $\mathsf{TPTL}_u^r\,(r \geq 2)$ *over infinite binary encoded pure data words is* PSPACE*-hard.*

Remark 5. Theorems 24, 26 and 28 are showed for the logic where all constraint numbers (or interval borders) are encoded in unary notation, and also hold if all constraint numbers (or interval borders) are given in binary notation. The constructions in the proofs of these theorems use only finite data words, and are easily adapted to infinite data words. Whereas Theorems 30, 31 and Corollary 23 only hold for infinite data words, since by Theorem 21 the path checking for TPTL^r over finite data words is in P. Furthermore, by Theorem 19, the constraint numbers in the logic of Theorem 30 and the data words in Theorem 31 have to be encoded in binary notation.

It is interesting to note that all lower bounds hold for the corresponding unary fragments except Theorem 31 and Corollary 23. The proof for Theorem 31 for $\mathsf{freezeLTL}^2$ needs the until operator. It is not clear, whether path checking for the unary fragment of $\mathsf{freezeLTL}^2$ over infinite binary encoded data words is still PSPACE-hard.

5.3 Summary of path checking results

We prove several upper and lower complexity bounds in Section 5.1 and Section 5.2, respectively. We summarizes our complexity results in this section.

Table 5.1 is an overview of our complexity results. We see that for $\mathsf{TPTL}, \mathsf{TPTL}^1, \mathsf{SMTL}$ and MTL, the type (finite or infinite) of the input data words, and the encoding (unary or binary) of data values and constraint numbers (or interval borders) do not influence the complexity. In fact, the complexity results still hold for their unary fragments, and if we consider the infinite monotonic data words of the form $(u)^{\omega}_{+k}$, then path checking of MTL is in $\mathsf{AC}^1(\mathsf{logDCFL})$. For $\mathsf{TPTL}^r\,(r \geq 2)$, the complexity depends on the input data words, and the encoding of the data words and constraint numbers.

Fig. 5.7 shows our complexity results, which depicts the relationship of different logics with respect to their expressive power (here the superscript $< \infty$ is a place holder for any number $r \geq 2$), over different classes of data words. Whether data words are pure or not does not change the complexity in all cases. Moreover, for finite data words, the complexity does not depend upon the encoding of data words (unary or binary) and the fact whether data words are monotonic or non-monotonic (for TPTL and SMTL). On the other hand, for infinite data words, these distinctions influence the complexity: For binary and non-monotonic data words we get another picture than for unary encoded or (quasi-)monotonic data words.

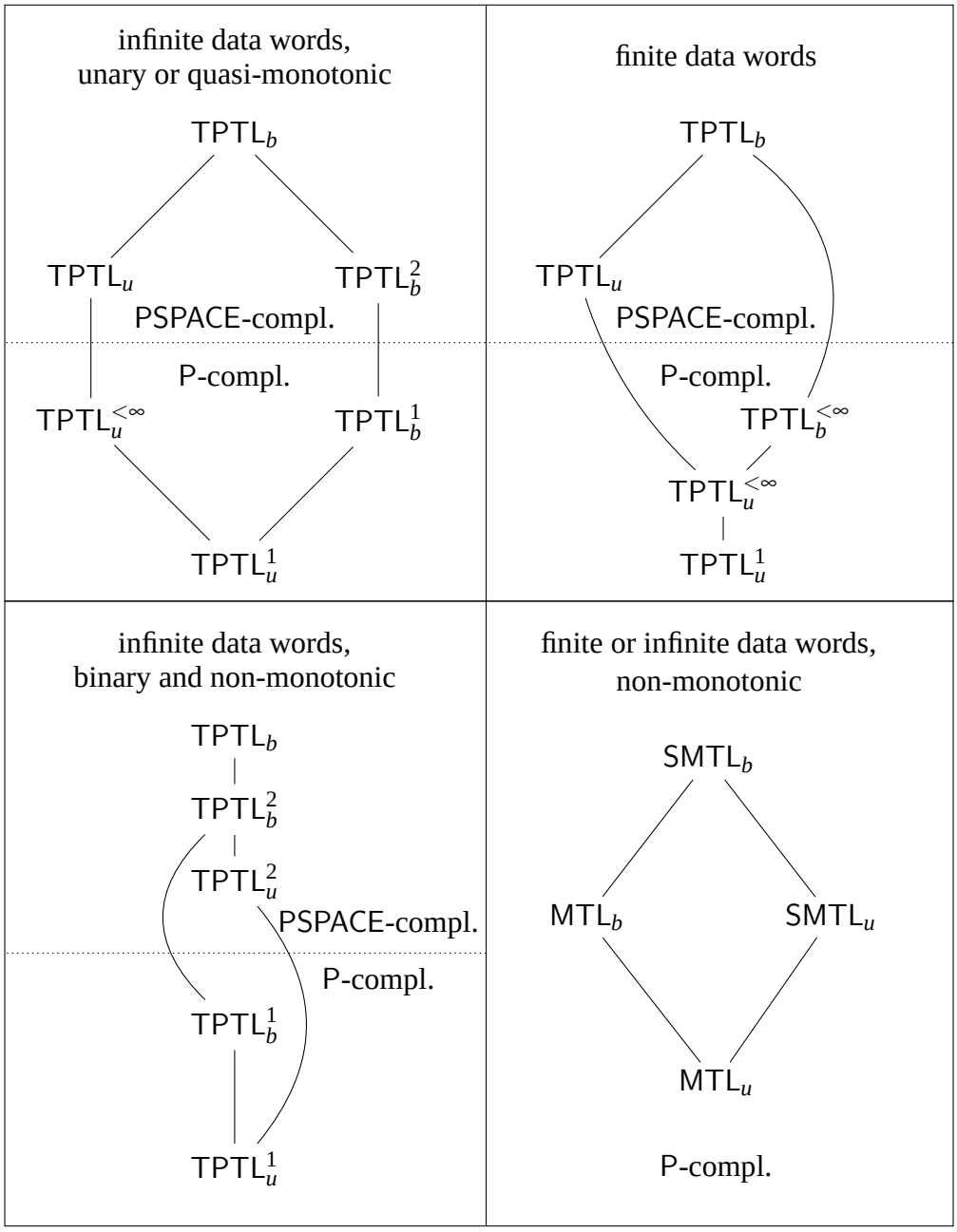

Fig. 5.7 Complexity results of path checking

	TPTL	$\mathsf{TPTL}_b^r\,(r \geq 2)$	$\mathsf{TPTL}_u^r\,(r \geq 2)$	$\mathsf{TPTL}^1, \mathsf{SMTL}, \mathsf{MTL}$
finite	PSPACE-compl.	P-compl.	P-compl.	P-compl.
infinite unary	PSPACE-compl.	PSPACE-compl.	P-compl.	P-compl.
infinite binary	PSPACE-compl.	PSPACE-compl.	PSPACE-compl.	P-compl.

Table 5.1 Complexity results of path checking

5.4 Model checking for deterministic one-counter machines

In this section, we consider the model checking problem over deterministic one-counter machines. We show that it is equivalent to the path checking problem over infinite unary encoded data words with respect to logarithmic space reductions.

A *one-counter machine* (OCM) $\mathscr{A}$ is a triple $\mathscr{A} = (Q, q_0, \Delta)$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, and $\Delta = Q \times \{-1, 0, 1\} \times Q$ is the transition relation. A configuration is a pair $(q, n) \in Q \times \mathbb{N}$. For configurations (p, m) and (q, n) we write $(p, m) \vdash_{\mathscr{A}} (q, n)$ if one of the following three cases holds:

- $(p, -1, q) \in \Delta$ and $n = m - 1$
- $(p, 1, q) \in \Delta$ and $n = m + 1$
- $(p, 0, q) \in \Delta$ and $n = m = 0$

An infinite run of $\mathscr{A}$ is an infinite sequence

$$(q_0, 0) \vdash_{\mathscr{A}} (q_1, n_1) \vdash_{\mathscr{A}} (q_2, n_2) \vdash_{\mathscr{A}} (q_3, n_3) \vdash_{\mathscr{A}} \cdots.$$

A finite run of $\mathscr{A}$ is a finite sequence

$$(q_0, 0) \vdash_{\mathscr{A}} (q_1, n_1) \vdash_{\mathscr{A}} (q_2, n_2) \vdash_{\mathscr{A}} \cdots \vdash_{\mathscr{A}} (q_l, n_l)$$

such that there does not exist a configuration (q,n) with $(q_l,n_l) \vdash_{\mathscr{A}} (q,n)$. We identify this run with the finite data word $(q_0,0)(q_1,n_1)(q_2,n_2)\cdots(q_l,n_l)$, and an infinite run is viewed as an infinite data word in the same way.

An OCM is *deterministic* (and called a DOCM) if for every state $p \in Q$, either there is exactly one outgoing transition (p,a,q) or there are exactly two outgoing transitions, which have to be of the form $(p,0,q_1)$ and $(p,-1,q_2)$ for states $q_1,q_2 \in Q$. This implies that $\mathscr{A}$ has a unique run (either finite or infinite), which we denote with $\mathsf{run}(\mathscr{A})$ and is viewed as a data word as explained above.

Lemma 24. *For a given DOCM $\mathscr{A}$ one can check in logarithmic space, whether $\mathsf{run}(\mathscr{A})$ is finite or infinite. Moreover, the following holds:*

- *If $\mathsf{run}(\mathscr{A})$ is finite, then the corresponding data word in unary encoding can be computed in logarithmic space.*
- *If $\mathsf{run}(\mathscr{A})$ is infinite, then one can compute in logarithmic space two unary encoded data words u_1 and u_2 and a unary encoded number k such that $\mathsf{run}(\mathscr{A}) = u_1(u_2)^\omega_{+k}$.*

Proof. In [31], the following statement was shown: If $\mathsf{run}(\mathscr{A})$ is infinite, then $\mathsf{run}(\mathscr{A}) = u_1(u_2)^\omega_{+k}$ with $k \leq |Q|$ and $|u_1u_2| \leq |Q|^3$. Hence, in order to check whether $\mathsf{run}(\mathscr{A})$ is infinite, we have to simulate $\mathscr{A}$ for at most $|Q|^3$ many steps. Thereby we check whether a configuration (q,n) is reached such that before, already a configuration (q,m) with $m < n$ has been reached. We store the current configuration with the counter value in binary together with a step counter t, which only needs logarithmic space (since the counter and the step counter are bounded by $|Q|^3$). Each time we produce a new configuration (q,n) (at step t), we have to check whether we have seen a configuration (q,m) with $m < n$ before. Since we cannot store the whole sequence of configuration, we have to "freeze" the simulation of $\mathscr{A}$ at the configuration (q,n) and then start a new simulation from the initial configuration for at most t steps. Thereby, the current configuration is compared with (q,n).

In a similar way, we can produce the data word $\mathsf{run}(\mathscr{A})$ itself in logarithmic space. We only have to print out the current configuration. Internally, our machine stores counter values in binary encoding. Since we want the output data word to be unary encoded, we have to transform the binary encoded counter values into unary encoding, which can be done with a logarithmic space machine. □

Let $\mathscr{L}$ be a logic. The *model checking problem for $\mathscr{L}$ over DOCM* is defined as follows:

Input: A DOCM $\mathscr{A}$ and a formula $\varphi \in \mathscr{L}$.

Output: `yes` if $\mathsf{run}(\mathscr{A}) \models \varphi$, `no` otherwise.

Theorem 32. *Let $\mathscr{L}$ be one of the logic* $\mathsf{TPTL}, \mathsf{TPTL}^r, \mathsf{SMTL}$ *or* MTL. *The model checking problem for $\mathscr{L}$ over DOCM is equivalent with respect to logarithmic space reductions to the path checking problem for $\mathscr{L}$ over infinite unary encoded data words.*

Proof. The reduction from the model checking problem for $\mathscr{L}$ over DOCM to the path checking problem for $\mathscr{L}$ over infinite unary encoded data words follows from Lemma 24. For the other direction take a unary encoded infinite data word $w = u_1(u_2)^{\omega}_{+k}$ and a formula $\psi \in \mathscr{L}$. Of course, w does not have to be of the form $\mathsf{run}(\mathscr{A})$ for a DOCM $\mathscr{A}$, since in a data word $\mathsf{run}(\mathscr{A})$ the data value can only change by at most 1 for neighboring positions. On the other hand, the latter can be easily enforced by inserting dummy positions in between the positions of w. Let $w' = v_1(v_2)^{\omega}_{+k}$ be the resulting data words. Then, we can easily construct in logarithmic space a DOCM $\mathscr{A}$ such that the sequence of counter values produced by $\mathscr{A}$ is the sequence of data values of w'. Moreover, no state of $\mathscr{A}$ repeats among the first $|v_1v_2|-1$ many positions. It is then straightforward to construct a formula $\psi' \in \mathscr{L}$ such that $w \models \psi$ if and only if $\mathsf{run}(\mathscr{A}) \models \psi'$. □

By Theorem 32, the complexity results from Table 5.1 and Fig. 5.7 proved for $\mathscr{L}$ over infinite unary encoded data words also shows the complexity of model checking for $\mathscr{L}$ over DOCM.

Chapter 6

Conclusion and future work

In this thesis, we studied the expressive power, satisfiability problems and path checking problems for MTL and TPTL over data words. In [6], the authors showed that over infinite monotonic data words, MTL and TPTL have the same expressive power and the satisfiability problem is decidable. We considered MTL and TPTL over infinite data words and finite data words separately. Now we briefly summarize the contents of this thesis and mention some directions for the future work.

In Chapter 2, we gave some basic definitions and notations about data words, metric temporal logic, timed propositional temporal logic, the relative expressive power, computational complexity and two-counter machines.

In Chapter 3, we introduced Ehrenfeucht–Fraïssé games for MTL and TPTL over data words, respectively, which are quantitative extensions of the EF-game for LTL over words defined in [36]. Every MTL-formula is equivalent to a TPTL-formula where only one register variable is used. Using the EF-game for MTL, we showed that TPTL is strictly more expressive than MTL over both infinite data words and finite data words. Actually, we showed that TPTL^1 is strictly more expressive than MTL by proving that the TPTL^1-formula $x.\mathsf{XX}(x=0)$ is not definable in MTL. Furthermore, we showed that the MTL definability problem: whether a TPTL-formula is definable in MTL, is undecidable over both infinite and finite data words by reductions of recurrent state problem and halting problem of two-counter machines. The register variables in TPTL play an important role in reaching its greater expressive power compared to MTL. When restricting the number of register variables, we were able to show that there is a strict increase in expressiveness when allowing two register variables instead of just one, i.e., TPTL^2 is strictly more expressive than TPTL^1. It is still open for the general case that whether $\mathsf{TPTL}^r \prec \mathsf{TPTL}^{r+1}$, where $r \geq 2$. We conjecture that the register variable hierarchy for TPTL is strict.

We considered the expressive power of several fragments of MTL and TPTL by restriction of the until rank and the set of constraint numbers (or interval borders). We showed that the until rank hierarchies for MTL and TPTL are strict over both infinite and finite data words, and whether an MTL-formula (respectively, TPTL-formula) is definable in MTL_k (respectively, TPTL_k) is undecidable for every $k \in \mathbb{N}$. We also obtained linear constraint hierarchies and lattice constraint hierarchies for MTL and TPTL when the set of constraint numbers (or interval borders) is restricted.

We also considered the expressive power of MTL that uses the non-strict semantics for the until modality. We showed that, over non-monotonic data words, MTL with strict semantics is strictly more expressive than MTL with non-strict semantics.

In Chapter 4, we considered infinitary SAT and finitary SAT for MTL and some fragments of MTL and TPTL. We showed that for MTL, the unary fragment of MTL and the pure fragment of MTL, infinitary SAT is Σ_1^1-complete and finitary SAT is Σ_1^0-complete. This still holds even for the unary fragment of MTL with two propositions and for the unary fragment of TPTL^1 without the X modality. We proved the undecidability of infinitary SAT (respectively, finitary SAT) by a reduction from the recurrent state problem (respectively, halting problem) of two-counter machines. However, it is an open problem whether undecidability also holds for the unary fragment of MTL in which the X modality is not allowed.

For the positive fragments of MTL and TPTL, we showed that a positive formula is satisfiable if and only it is satisfied by a finite data word. Finitary SAT and infinitary SAT coincide for positive MTL and positive TPTL. Both of them are Σ_1^0-complete. For existential TPTL (respectively, existential MTL) that is the fragment of positive TPTL (respectively, positive MTL) in which we only use the F and X modalities, we showed that SAT is NP-complete.

In Chapter 5, we considered the complexity of path checking problems for MTL and TPTL over data words. We showed that path checking for TPTL is PSPACE-complete, and for MTL is P-complete. The type (finite or infinite) of the input data words, and the encoding (unary or binary) of data values and constraint numbers (or interval borders) do not influence the complexity. If the number of register variables allowed in a formula is restricted, we obtained path checking for TPTL^1 is P-complete over both infinite and finite data words; for TPTL^2 is PSPACE-complete over infinite data words; and for $\mathsf{TPTL}^r\,(r \geq 2)$ is P-complete over finite data words. If the encoding of constraint numbers of the input TPTL-formula is in unary notation, we showed that path checking for $\mathsf{TPTL}^r\,(r \geq 2)$ is P-complete over infinite unary encoded data words or infinite binary encoded quasi-monotonic data words.

For MTL, we proved the P-hardness over non-monotonic data words. This is unavoidable by the result in [21] that path checking for MTL over monotonic data words belongs to $\mathsf{AC}^1(\mathsf{logDCFL})$. We introduced SMTL which is a succinct version of MTL. For SMTL,

we showed that path checking over monotonic data words is P-complete. We also showed that path checking for MTL over infinite monotonic data words of the form $(u)^{\omega}_{+k}$ belongs to $\mathsf{AC}^1(\mathsf{logDCFL})$.

In the last section of this chapter, we extended these results to model checking over deterministic one-counter machines.

References

[1] Parosh Aziz Abdulla and Karlis Cerans. Simulation is decidable for one-counter nets (extended abstract). In *Concurrency Theory, 9th International Conference, Nice, France, September 8-11, 1998*, volume 1466 of *LNCS*, pages 253–268. Springer, 1998.

[2] Sara Adams, Joël Ouaknine, and James Worrell. Undecidability of universality for timed automata with minimal resources. In *Formal Modeling and Analysis of Timed Systems, 5th International Conference, FORMATS 2007, Salzburg, Austria, October 3-5, 2007*, volume 4763 of *LNCS*, pages 25–37. Springer, 2007.

[3] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[4] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, January 1996.

[5] Rajeev Alur and Thomas A. Henzinger. Logics and models of real time: A survey. In J. W. de Bakker, Cornelis Huizing, Willem P. de Roever, and Grzegorz Rozenberg, editors, *REX Workshop*, volume 600 of *LNCS*, pages 74–106. Springer, 1991.

[6] Rajeev Alur and Thomas A. Henzinger. Real-time logics: Complexity and expressiveness. *Inf. Comput.*, 104(1):390–401, 1993.

[7] Rajeev Alur and Thomas A. Henzinger. A really temporal logic. *J. ACM*, 41(1):181–204, 1994.

[8] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, September 2002.

[9] Rajeev Alur and P. Madhusudan. Decision problems for timed automata: A survey. In *Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT 2004, Bertinoro, Italy, September 13-18, 2004, Revised Lectures*, volume 3185 of *LNCS*, pages 1–24. Springer, 2004.

[10] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[11] Mikolaj Bojanczyk. The common fragment of ACTL and LTL. In *Foundations of Software Science and Computational Structures, 11th International Conference, FOSSACS 2008, Budapest, Hungary, March 29 - April 6, 2008*, volume 4962 of *LNCS*, pages 172–185. Springer, 2008.

[12] Mikolaj Bojanczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Log.*, 12(4):27, 2011.

[13] Benedikt Bollig. An automaton over data words that captures EMSO logic. In *Concurrency Theory - 22nd International Conference, CONCUR 2011, Aachen, Germany, September 6-9, 2011*, volume 6901 of *LNCS*, pages 171–186. Springer, 2011.

[14] Benedikt Bollig, Aiswarya Cyriac, Paul Gastin, and K. Narayan Kumar. Model checking languages of data words. In *Foundations of Software Science and Computational Structures - 15th International Conference, FOSSACS 2012, Tallinn, Estonia, March 24 - April 1, 2012*, volume 7213 of *LNCS*, pages 391–405. Springer, 2012.

[15] Rémi Bonnet. Decidability of LTL for vector addition systems with one zero-test. In *Reachability Problems - 5th International Workshop, Genoa, Italy, September 28-30, 2011*, volume 6945 of *LNCS*, pages 85–95. Springer, 2011.

[16] Ahmed Bouajjani, Javier Esparza, and Oded Maler. Reachability analysis of pushdown automata: Application to model-checking. In *CONCUR '97: Concurrency Theory, 8th International Conference, Warsaw, Poland, July 1-4, 1997*, volume 1243 of *LNCS*, pages 135–150. Springer, 1997.

[17] Patricia Bouyer. A logical characterization of data languages. *Inf. Process. Lett.*, 84(2):75–85, 2002.

[18] Patricia Bouyer, Fabrice Chevalier, and Nicolas Markey. On the expressiveness of TPTL and MTL. *Inf. Comput.*, 208(2):97–116, 2010.

[19] Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, and Nicolas Markey. Timed automata with observers under energy constraints. In *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2010, Stockholm, Sweden, April 12-15, 2010*, pages 61–70. ACM, 2010.

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

[21] Daniel Bundala and Joël Ouaknine. On the complexity of temporal-logic path checking. In *Automata, Languages, and Programming - 41st International Colloquium, Copenhagen, Denmark, July 8-11, 2014*, volume 8573 of *LNCS*, pages 86–97, 2014.

[22] John P. Burgess and Yuri Gurevich. The decision problem for linear temporal logic. *Notre Dame Journal of Formal Logic*, 26(2):115–128, 04 1985.

[23] Claudia Carapelle, Shiguang Feng, Oliver Fernandez Gil, and Karin Quaas. On the expressiveness of TPTL and MTL over ω-data words. In *Proceedings 14th International Conference on Automata and Formal Languages, AFL 2014, Szeged, Hungary, May 27-29, 2014*, volume 151 of *EPTCS*, pages 174–187, 2014.

[24] Claudia Carapelle, Shiguang Feng, Oliver Fernandez Gil, and Karin Quaas. Satisfiability for MTL and TPTL over non-monotonic data words. In *Language and Automata Theory and Applications - 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014*, volume 8370 of *LNCS*, pages 248–259, 2014.

[25] Claudia Carapelle, Shiguang Feng, Alexander Kartzow, and Markus Lohrey. Satisfiability of ECTL* with tree constraints. In Lev D. Beklemishev and Daniil V. Musatov, editors, *Computer Science - Theory and Applications - 10th International Computer Science Symposium in Russia, CSR 2015, Listvyanka, Russia, July 13-17, 2015*, volume 9139 of *LNCS*, pages 94–108. Springer, 2015.

[26] Joelle Cohen, Dominique Perrin, and Jean-Eric Pin. On the expressive power of temporal logic. *J. COMPUT. SYSTEM SCI*, 46:271–294, 1993.

[27] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001.

[28] Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3), 2009.

[29] Stéphane Demri, Ranko Lazic, and David Nowak. On the freeze quantifier in constraint LTL: Decidability and complexity. *Inf. Comput.*, 205(1):2–24, 2007.

[30] Stéphane Demri, Ranko Lazic, and Arnaud Sangnier. Model checking freeze LTL over one-counter automata. In *Foundations of Software Science and Computational Structures, 11th International Conference, FOSSACS 2008, Budapest, Hungary, March 29 - April 6, 2008*, volume 4962 of *LNCS*, pages 490–504. Springer, 2008.

[31] Stéphane Demri, Ranko Lazić, and Arnaud Sangnier. Model checking memoryful linear-time logics over one-counter automata. *Theoretical Computer Science*, 411(22-24):2298–2316, 2010.

[32] Stéphane Demri and Arnaud Sangnier. When model-checking freeze LTL over counter machines becomes decidable. In *Foundations of Software Science and Computational Structures, 13th International Conference, FOSSACS 2010, Paphos, Cyprus, March 20-28, 2010*, volume 6014 of *LNCS*, pages 176–190. Springer, 2010.

[33] Deepak D'Souza and Pavithra Prabhakar. On the expressiveness of MTL in the pointwise and continuous semantics. *STTT*, 9(1):1–4, 2007.

[34] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 1999.

[35] Javier Esparza. Decidability and complexity of Petri net problems - an introduction. In *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets, held in Dagstuhl, September 1996*, volume 1491 of *LNCS*, pages 374–428. Springer, 1996.

[36] Kousha Etessami and Thomas Wilke. An until hierarchy for temporal logic. In *11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 108–117. IEEE Computer Society, 1996.

[37] Kousha Etessami and Thomas Wilke. An until hierarchy and other applications of an Ehrenfeucht-Fraïssé game for temporal logic. *Inf. Comput.*, 160(1-2):88–108, 2000.

[38] John Fearnley and Marcin Jurdziński. Reachability in two-clock timed automata is PSPACE-complete. In *Proceedings of the 40th International Conference on Automata, Languages, and Programming - Volume Part II*, ICALP'13, pages 212–223, Berlin, Heidelberg, 2013. Springer-Verlag.

[39] Shiguang Feng, Markus Lohrey, and Karin Quaas. Path checking for MTL and TPTL over data words. In Igor Potapov, editor, *Developments in Language Theory - 19th International Conference, DLT 2015, Liverpool, UK, July 27-30, 2015*, volume 9168 of *LNCS*, pages 326–339. Springer, 2015.

[40] Diego Figueira and Luc Segoufin. Future-looking logics on data words and trees. In *Mathematical Foundations of Computer Science 2009, 34th International Symposium, MFCS 2009, Novy Smokovec, High Tatras, Slovakia, August 24-28, 2009*, volume 5734 of *LNCS*, pages 331–343. Springer, 2009.

[41] Carlo Alberto Furia and Matteo Rossi. On the expressiveness of MTL variants over dense time. In *Proceedings of the 5th International Conference on Formal Modeling and Analysis of Timed Systems*, FORMATS'07, pages 163–178, Berlin, Heidelberg, 2007. Springer-Verlag.

[42] Stefan Göller, Christoph Haase, Joël Ouaknine, and James Worrell. Model checking succinct and parametric one-counter automata. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010*, volume 6199 of *LNCS*, pages 575–586. Springer, 2010.

[43] V. Goranko. Hierarchies of modal and temporal logics with reference pointers. *Journal of Logic, Language and Information*, 5(1):1–24, 1996.

[44] Raymond Greenlaw, H. James Hoover, and Walter L. Ruzzo. *Limits to Parallel Computation: P-completeness Theory*. Oxford University Press, Inc., New York, NY, USA, 1995.

[45] Tom Henzinger and Vinayak Prabhu. Timed alternating-time temporal logic. In Patricia Asarin, Eugene; Bouyer, editor, *Formal Modeling and Analysis of Timed Systems, 4th International Conference, FORMATS 2006, Paris, France, LCNS 4202*, pages 1–17, September 2006.

[46] William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *J. Comput. Syst. Sci.*, 65(4):695–716, 2002.

[47] Yoram Hirshfeld and Alexander Rabinovich. Continuous time temporal logic with counting. *Inf. Comput.*, 214:1–9, May 2012.

[48] Yoram Hirshfeld and Alexander Moshe Rabinovich. Logics for real time: Decidability and complexity. *Fundam. Inform.*, 62(1):1–28, 2004.

[49] Yoram Hirshfeld and Alexander Moshe Rabinovich. Expressiveness of metric modalities for continuous time. In *Computer Science - Theory and Applications, First International Computer Science Symposium in Russia, CSR 2006, St. Petersburg, Russia, June 8-12, 2006*, volume 3967 of *LNCS*, pages 211–220. Springer, 2006.

[50] Ian Hodkinson, Frank Wolter, and Michael Zakharyaschev. Decidable fragments of first-order temporal logics. *Annals of Pure and Applied Logic*, 106(1-3):85–134, 2000.

[51] Paul Hunter. When is metric temporal logic expressively complete? In *Computer Science Logic 2013, CSL 2013, September 2-5, 2013, Torino, Italy*, volume 23 of *LIPIcs*, pages 380–394. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013.

[52] Paul Hunter, Joël Ouaknine, and James Worrell. Expressive completeness for metric temporal logic. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 349–357. IEEE Computer Society, 2013.

[53] Joxan Jaffar, Michael J. Maher, Peter J. Stuckey, and Roland H. C. Yap. Beyond finite domains. In *Principles and Practice of Constraint Programming, Second International Workshop, PPCP'94, Rosario, Orcas Island, Washington, USA, May 2-4, 1994*, volume 874 of *LNCS*, pages 86–94. Springer, 1994.

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

[55] Lars Kuhtz and Bernd Finkbeiner. Efficient parallel path checking for linear-time temporal logic with past and bounds. *Logical Methods in Computer Science*, 8(4), 2012.

[56] François Laroussinie, Nicolas Markey, and Ph. Schnoebelen. On model checking durational Kripke structures. In *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002*, volume 2303 of *LNCS*, pages 264–279. Springer, 2002.

[57] Étienne Lozes. Adjuncts elimination in the static ambient logic. *Electr. Notes Theor. Comput. Sci.*, 96:51–72, 2004.

[58] Monika Maidl. The common fragment of CTL and LTL. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 643–652. IEEE Computer Society, 2000.

[59] Nicolas Markey and Ph. Schnoebelen. Model checking a path. In *Concurrency Theory, 14th International Conference, Marseille, France, September 3-5, 2003*, volume 2761 of *LNCS*, pages 248–262. Springer, 2003.

[60] Marvin L. Minsky. Recursive unsolvability of Post's problem of tag and other topics in theory of Turing machines. *Annals of Mathematics*, 74(3):437–455, November 1961.

[61] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, NJ, 1967.

[62] Joël Ouaknine and James Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland*, pages 54–63. IEEE Computer Society, 2004.

[63] Joël Ouaknine and James Worrell. On metric temporal logic and faulty Turing machines. In *Foundations of Software Science and Computation Structures, 9th International Conference, FOSSACS 2006, Vienna, Austria, March 25-31, 2006*, volume 3921 of *LNCS*, pages 217–230. Springer, 2006.

[64] Joël Ouaknine and James Worrell. Safety metric temporal logic is fully decidable. In *Tools and Algorithms for the Construction and Analysis of Systems, 12th International Conference, TACAS 2006, Vienna, Austria, March 25 - April 2, 2006*, volume 3920 of *LNCS*, pages 411–425. Springer, 2006.

[65] Joël Ouaknine and James Worrell. On the decidability and complexity of metric temporal logic over finite words. *Logical Methods in Computer Science*, 3(1), 2007.

[66] Paritosh K. Pandya and Simoni S. Shah. On expressive powers of timed logics: Comparing boundedness, non-punctuality, and deterministic freezing. In *Concurrency Theory - 22nd International Conference, CONCUR 2011, Aachen, Germany, September 6-9, 2011*, volume 6901 of *LNCS*, pages 60–75. Springer, 2011.

[67] Pawel Parys and Igor Walukiewicz. Weak alternating timed automata. *Logical Methods in Computer Science*, 8(3), 2012.

[68] Pavithra Prabhakar and Deepak D'Souza. On the expressiveness of MTL with past operators. In *Formal Modeling and Analysis of Timed Systems, 4th International Conference, FORMATS 2006, Paris, France, September 25-27, 2006*, volume 4202 of *LNCS*, pages 322–336. Springer, 2006.

[69] Karin Quaas. Model checking metric temporal logic over automata with one counter. In *Language and Automata Theory and Applications - 7th International Conference, LATA 2013, Bilbao, Spain, April 2-5, 2013*, volume 7810 of *LNCS*, pages 468–479. Springer, 2013.

[70] Alexander Rabinovich. Complexity of metric temporal logics with counting and the pnueli modalities. *Theor. Comput. Sci.*, 411(22-24):2331–2342, May 2010.

[71] Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4):17:1–17:24, September 2008.

[72] M. Reynolds. The complexity of the temporal logic with "until" over general linear time. *J. Comput. Syst. Sci.*, 66(2):393–426, March 2003.

[73] Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. MIT Press, Cambridge, MA, USA, 1987.

[74] Kristin Y. Rozier. Survey: Linear temporal logic symbolic model checking. *Comput. Sci. Rev.*, 5(2):163–203, May 2011.

[75] Özlem Salehi, Abuzer Yakaryilmaz, and A. C. Cem Say. Real-time vector automata. In *Fundamentals of Computation Theory - 19th International Symposium, FCT 2013, Liverpool, UK, August 19-21, 2013*, volume 8070 of *LNCS*, pages 293–304. Springer, 2013.

[76] Luc Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006*, volume 4207 of *LNCS*, pages 41–57. Springer, 2006.

[77] Michael Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1st edition, 1996.

[78] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, July 1985.

[79] Stephen Travers. The complexity of membership problems for circuits over sets of integers. *Theor. Comput. Sci.*, 369(1):211–229, December 2006.

[80] Moshe Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*, volume 1043 of *LNCS*, pages 238–266. Springer, 1996.

[81] Thomas Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In *Formal Techniques in Real-Time and Fault-Tolerant Systems, Third International Symposium Organized Jointly with the Working Group Provably Correct Systems - ProCoS, Lübeck, Germany, September 19-23*, volume 863 of *LNCS*, pages 694–715. Springer, 1994.

[82] Pierre Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1-2):72–99, 1983.

Selbstständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, 06.01.2016

Shiguang Feng

Scientific Career Experience

Aug. 1999 – Jul. 2002 Qingzhou No.1 senior middle school, P.R. China

Sep. 2002 – Jul. 2006 Shandong Agricultural University

- Major: Pharmaceutical Engineering
- Minor: Computer Science and Technology
- received Bachelor's Degree of Engineering in July 2006

Sep. 2006 – Jul. 2012 Sun Yat-sen University

- Master-doctoral student of Institute of Logic and Cognition
- Major: Mathematical Logic, Supervisor: Prof. Dr. Xishun Zhao
- Thesis "The Complexity and Expressive Power of Second-Order HORN Logic"
- received Doctor's Degree of Philosophy in July 2012

Since Oct. 2012 Universität Leipzig

- Doctoral student of Institut für Informatik,
- Research direction: Quantitative temporal logic
- Supervisor: Prof. Dr. Markus Lohrey & Prof. Dr. Manfred Droste

List of Publications

1. C. Carapelle, S. Feng, A. Kartzow, and M. Lohrey: Satisfiability of ECTL* with tree constraints. In 10th International Computer Science Symposium in Russia (CSR 2015), volume 9139 of LNCS, pages 94-108. Springer, 2015.

2. S. Feng, M. Lohrey, and K. Quaas: Path checking for MTL and TPTL over data words. In 19th International Conference on Developments in Language Theory (DLT 2015), volume 9168 of LNCS, pages 326-339. Springer, 2015.

3. C. Carapelle, S. Feng, O. F. Gil, and K. Quaas: Satisfiability for MTL and TPTL over Non-monotonic Data Words. In 8th International Conference on Language and Automata Theory and Applications(LATA 2014), volume 8370 of LNCS, pages 248-259. Springer, 2014.

4. C. Carapelle and S. Feng, O. F. Gil, and K. Quaas: On the Expressiveness of TPTL and MTL over ω-Data Words. In 14th International Conference on Automata and Formal Languages(AFL 2014), Szeged, Hungary, May 27-29, 2014, volume 151 of EPTCS, pages 174-187, 2014.

5. S. Feng and X. Zhao. Complexity and Expressive Power of Second-Order Extended Horn Logic, Mathematical Logic Quarterly, 59. 1-2(2013):4-11.

6. S. Feng and X. Zhao. The Complexity and Expressive Power of Second-Order Extended Logic, Studies in Logic, 2012, 5(1):11-34.

List of Talks

1. Ehrenfeucht-Fraïssé Games for Metric Temporal Logic on Data Words. Highlights of Logics, Games and Automata 2013, Paris, 21.09.2013.
2. On the Expressiveness of TPTL and MTL over ω-Data Words. AFL 2014, Szeged, 28.05.2014.
3. Satisfiability of ECTL* with tree constraints. CSR 2015, Listvyanka, 13.07.2015.
4. The Complexity of Path Checking for MTL and TPTL over Data Words. DLT 2015, Liverpool, 29.07.2015.
5. The Complexity of Path Checking for MTL and TPTL over Data Words. Highlights of Logics, Games and Automata 2015, Prague, 18.09.2015.