Available online at www.sciencedirect.com
SciVerse ScienceDirect
Electronic Notes in Theoretical Computer Science 278 (2011) 115–128
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Expressiveness of Hybrid Temporal Logic on Data Words

Ahmet Kara[1,2] Thomas Schwentick[1,3]

TU Dortmund, Germany

Abstract

Hybrid temporal logic (HTL) on data words can be considered as an extension of the logic LTL$^{\downarrow}$ introduced by Demri and Lazic [3]. The paper compares the expressive power of HTL on data words with that of LTL$^{\downarrow}$. It is shown that there are properties of data words that can be expressed in HTL with two variables but not in LTL$^{\downarrow}$. On the other hand, every property that can be expressed in HTL with one variable can also be expressed in LTL$^{\downarrow}$ with one variable. The paper further studies the succinctness of HTL in comparison with LTL$^{\downarrow}$ and shows that the number-of-variables hierarchy of HTL is infinite.

Keywords: Hybrid logics, temporal logics, data words.

1 Introduction

In this paper, a data word is a finite sequence of positions which carry a data value and a set of propositions. Logics on data words have been investigated a lot in recent years, e.g., in [2,3] and many follow-up papers. In this paper we consider hybrid temporal logic on data words. As an example, the formula $\varphi = \mathrm{G}(p \wedge {\downarrow}x.\mathrm{F}(q \wedge {\sim}x))$ expresses that for every position carrying the proposition p there is a position in the future that has the same data value and carries proposition q. The quantifier ${\downarrow}x$ binds variable x to the current position and ${\sim}x$ compares the current data value with the value at the position bound to x. The logic further allows formulas as $@x.\chi$ (evaluate χ at position x) and x (true if x is the current position).

Hybrid temporal logic was first considered in [13], and intensively studied on linear structures, e.g., in [1,9,15]. It has been noted before that the logic LTL$^{\downarrow}$ on data words, introduced in [3] is essentially a hybrid temporal logic [4,5,19]. In

[1] We acknowledge the financial support by the German DFG under grant SCHW 678/4-1.
[2] Email: ahmet.kara@cs.uni-dortmund.de
[3] Email: thomas.schwentick@cs.tu-dortmund.de

doi:10.1016/j.entcs.2011.10.010

fact, $\mathrm{LTL}^{\downarrow}$ can be considered as the syntactical fragment of hybrid temporal logic without formulas of the forms $@x.\chi$ and x. Formally, in $\mathrm{LTL}^{\downarrow}$, variables are bound to data values instead of positions. However, the difference does not matter as long as the only way to refer to a variable x is through an atomic formula $\sim x$. Thus, formula φ above can be seen as a $\mathrm{LTL}^{\downarrow}$ formula.

We study hybrid temporal logics with future *and* past temporal operators (HTL) on data words and compare its expressive power with that of $\mathrm{LTL}^{\downarrow}$.

There is a correspondence between HTL and $\mathrm{LTL}^{\downarrow}$ on one hand and automata models for data words, on the other hand. As shown in [4], every property expressible in $\mathrm{LTL}^{\downarrow}$ can be decided by an alternating register automaton. The logic HTL, on the other hand, is captured by pebble automata. This follows directly from the fact that HTL can only express first-order properties and that even deterministic one-way pebble automata can express [4] all first-order properties on data words [12].

The relationship between $\mathrm{LTL}^{\downarrow}$ and alternating register automata together with previous results in [12] immediately yields a separation between $\mathrm{LTL}^{\downarrow}$ and HTL. Indeed, in [12] a first-order expressible property of data words was defined that cannot be decided by alternating register automata and therefore cannot be expressed by $\mathrm{LTL}^{\downarrow}$. On the other hand, it is not hard to show that HTL can express all first-order properties of data words and thus also this particular property. We strengthen this result by showing that a similar property that still cannot be expressed in $\mathrm{LTL}^{\downarrow}$ can be expressed by an HTL formula with only two variables, thus establishing $\mathrm{LTL}^{\downarrow} \not\leq \mathrm{HTL}_2$.

Interestingly, the difference between HTL and $\mathrm{LTL}^{\downarrow}$ vanishes when we restrict formulas to one variable. Our main result shows that $\mathrm{HTL}_1 = \mathrm{LTL}^{\downarrow}_1$.

We further show that

- HTL_1-formulas can be exponentially more succinct than $\mathrm{LTL}^{\downarrow}$-formulas and HTL-formulas can even be non-elementarily more succinct than $\mathrm{LTL}^{\downarrow}$-formulas; and that
- the variable hierarchy of HTL on data strings is infinite. To this end, we show that Rossman's result that the variable hierarchy for first-order logic on finite ordered graphs is strict can be carried over to data strings.

Most of our results also hold for infinite data words (cf. Section 5).

We already mentioned related work above. The paper is organized as follows. After the preliminaries (Section 2) we show in Section 3 the results on HTL with at least two variables and in Section 4 the results on HTL_1. We conclude in Section 5. Due to lack of space we do not present all proofs in full detail. In particular, some correctness proofs for translations of formulas are missing. However, we always tried to design the translated formulas so that the correctness can be verified easily by the reader.

We thank Thomas Zeume and an anonymous reviewer for many useful suggestions.

[4] Formally, this was only shown for finite data words over an empty proposition set in [12]. However, the proof simply goes through for data words.

2 Preliminaries

Data Words. Let Σ be a finite set of propositions and $\mathcal{D}$ an infinite set of data values. A *string* over Σ is a finite sequence of elements from 2^Σ. A *data word* over Σ is a finite sequence of elements from $(2^\Sigma \times \mathcal{D})$. A data word is denoted by $w = \frac{P_1}{d_1} \cdots \frac{P_n}{d_n}$, where $P_1, \ldots, P_n \in 2^\Sigma$ and $d_1, \ldots, d_n \in \mathcal{D}$. If $p \in P_i$ we say that i is *labelled* with p and that i is a *p-position.* We call d_i *the data value of position* i. If P_i is a singleton set $\{p\}$ we write $\frac{p}{d}$ instead of $\frac{\{p\}}{d}$.

Logics. *Hybrid temporal logic* (HTL) on data words over Σ is an extension of the linear temporal logic LTL (see e.g. [7]) by past operators and variables. The syntax of HTL is as follows.

$$\varphi ::= p \mid x \mid {\downarrow} x.\varphi \mid {\sim} x \mid @x.\varphi \mid \mathrm{X}\varphi \mid \varphi \mathrm{U} \varphi \mid \mathrm{X}^{-}\varphi \mid \varphi \mathrm{U}^{-}\varphi \mid \neg\varphi \mid \varphi \wedge \varphi$$

where $p \in \Sigma$ and x is a variable from an infinite set VAR of variables.

As usual, we consider the other logical operators $\vee$, $\rightarrow$ and $\leftrightarrow$ and the other temporal operators $\mathrm{F}\varphi := \top\mathrm{U}\varphi$, $\mathrm{F}^{-}\varphi := \top\mathrm{U}^{-}\varphi$, $\mathrm{G}\varphi := \neg\mathrm{F}\neg\varphi$ and $\mathrm{G}^{-}\varphi := \neg\mathrm{F}^{-}\neg\varphi$ as abbreviations. The operators $\mathrm{X}, \mathrm{U}, \mathrm{F}, \mathrm{G}, \mathrm{X}^{-}, \mathrm{U}^{-}, \mathrm{F}^{-}, \mathrm{G}^{-}$ are called *temporal operators.* We refer to the operators $\mathrm{X}, \mathrm{U}, \mathrm{F}, \mathrm{G}$ as *future operators* and to the latter four as *past operators.*

For a data word w with $|w| = n$ an *assignment* is a mapping $g : \mathrm{VAR} \to \{1, \ldots, n\}$. For an assignment g, a variable x and a position i in w, g_i^x denotes the mapping defined by $g_i^x(x) := i$ and $g_i^x(y) := g(y)$ for all $y \neq x$. The semantics of a HTL formula over Σ is defined with respect to a data word $w = \frac{P_1}{d_1} \cdots \frac{P_n}{d_n}$ over Σ, a position i in w and an assignment g for w.

- $w, g, i \models p$ if $p \in P_i$
- $w, g, i \models {\downarrow} x.\varphi$ if $w, g_i^x, i \models \varphi$
- $w, g, i \models x$ if $g(x) = i$
- $w, g, i \models {\sim} x$ if $d_i = d_{g(x)}$
- $w, g, i \models @x.\varphi$ if $w, g, g(x) \models \varphi$
- $w, g, i \models \neg\varphi$ if $w, g, i \not\models \varphi$

- $w, g, i \models \varphi \wedge \psi$ if $w, g, i \models \varphi$ and $w, g, i \models \psi$
- $w, g, i \models \mathrm{X}\varphi$ if $i < n$ and $w, g, i+1 \models \varphi$
- $w, g, i \models \varphi\mathrm{U}\psi$ if there is a j with $i \leq j \leq n$ and $w, g, j \models \psi$ and $w, g, k \models \varphi$ for all k with $i \leq k < j$
- $w, g, i \models \mathrm{X}^{-}\varphi$ if $i > 1$ and $w, g, i-1 \models \varphi$
- $w, g, i \models \varphi\mathrm{U}^{-}\psi$ if there is a j with $1 \leq j \leq i$ and $w, g, j \models \psi$ and $w, g, k \models \varphi$ for all k with $j < k \leq i$

If at most one variable x occurs in φ and $g(x) = j$, we also write $w, j, i \models \varphi$ instead of $w, g, i \models \varphi$. We say that *a data word w satisfies a formula φ* (denoted as $w \models \varphi$) if $w, g, 1 \models \varphi$ where $g(x) := 1$ for all $x \in \mathrm{VAR}$.

The notions of *bound* and *free* variable occurrences and of *closed formulas* are defined as usual. We say that two formulas φ_1 and φ_2 are *equivalent* (denoted as $\varphi_1 \equiv \varphi_2$) if $w, g, i \models \varphi_1$ if and only if $w, g, i \models \varphi_2$ for all data words w, all assignments g and all positions i of w. The formulas are *initially equivalent* if for all data words w and assignments g it holds $w, g, 1 \models \varphi_1$ if and only if $w, g, 1 \models \varphi_2$. For two logics $\mathcal{L}_1$ and $\mathcal{L}_2$ we write $\mathcal{L}_1 \leq \mathcal{L}_2$ if for every closed formula φ_1 of $\mathcal{L}_1$ there is an equivalent formula φ_2 in $\mathcal{L}_2$. Furthermore, $\mathcal{L}_1 \equiv \mathcal{L}_2$ if $\mathcal{L}_1 \leq \mathcal{L}_2$ and $\mathcal{L}_2 \leq \mathcal{L}_1$.

Freeze LTL ($\text{LTL}^{\downarrow}$) ([3]) is the fragment of HTL that does not allow (sub-) formulas of the form x and $@x.\varphi$. The fragments of HTL and $\text{LTL}^{\downarrow}$ where at most k variables are allowed, are denoted as HTL_k and $\text{LTL}^{\downarrow}_k$, respectively. For a set $\mathcal{O}$ of temporal operators and a logic $\mathcal{L}$, we denote by $\mathcal{L}(\mathcal{O})$ the fragment of $\mathcal{L}$ in which only operators from $\mathcal{O}$ are used. The extension of LTL by past operators is denoted as PLTL.

3 HTL with more than one variable

In this section, we consider HTL without a bound on the number of variables. We first show that HTL is more expressive than $\text{LTL}^{\downarrow}$ and that it actually only needs two variables to express a property that $\text{LTL}^{\downarrow}$ cannot express. Furthermore, we show that the variable hierarchy for HTL is infinite and that HTL is non-elementarily more succinct than $\text{LTL}^{\downarrow}$.

3.1 Expressiveness

The expressive power of HTL on data words coincides with the expressive power of first-order logic (FO). Here, first-order formulas can use the atomic formulas $p(x)$ (stating that proposition p holds at position x), $x < y$ (stating that position y is to the right of position x) and $x \sim y$ (stating that x and y carry the same data value).

Theorem 3.1 HTL $\equiv$ FO *on data words.*

Proof. For every $k \geq 1$ every HTL_k-formula can be translated into a first-order formula with at most $k + 3$ variables by a standard translation (similarly as in, e.g. [9]). The translation of first-order formulas into HTL-formulas is also along standard lines. A first-order formula φ can be translated into a HTL-formula φ' as follows (we omit the Boolean cases).

- $(\exists x \psi)' = \text{FF}^{-}{\downarrow}x.\psi'$
- $(x < y)' = @x.\text{XF}y$
- $(x \sim y)' = @x.{\sim}y$
- $(x = y)' = \text{FF}^{-}(x \wedge y)$
- $(p(x))' = @x.p$

$\square$

It follows from existing results that HTL is strictly more expressive than $\text{LTL}^{\downarrow}$. In [3] it was shown that every property of data words expressed by a $\text{LTL}^{\downarrow}$-formula can be decided by a 2-way alternating register automaton. In [12], for every $m \geq 1$, a set $L^{=}_m$ of data words was defined such that

(a) $L^{=}_m$ is definable in first-order logic for every m, but

(b) for every $m \geq 4$, there is no 2-way alternating register automaton for $L^{=}_m$.

Of course, (b) and [3] yield[5] that $L_m^=$ cannot be defined by a LTL$^\downarrow$-formula, for any $m \geq 4$. By Theorem 3.1 and (a), every $L_m^=$ can be expressed in HTL and thus HTL is strictly more expressive than LTL$^\downarrow$. In Theorem 3.3 we show that, for every m, there is a set L_m, similarly defined as $L_m^=$, such that L_m can be defined by a HTL$_2$(X, U)-formula, but there is no 2-way alternating register automaton for L_m if $m \geq 4$. Thus, HTL$_2$(X, U) $\not\leq$ LTL$^\downarrow$.

A *1-hyperset* over $\mathcal{D}$ is a finite subset of $\mathcal{D}$. For $m > 1$, an *m-hyperset* over $\mathcal{D}$ is a finite set of $(m-1)$-hypersets over $\mathcal{D}$. We assume for simplicity that $\mathcal{D}$ is the set of natural numbers. We associate m-hypersets $H_m(w)$ with data words w over $\{z, b_1, e_1, \ldots, b_m, e_m\}$ as follows. A data word $w = {}^{b_1}_{d}\, {}^{z}_{n_1} \cdots {}^{z}_{n_j}\, {}^{e_1}_{d'}$, where d and d' are arbitrary data values, represents the 1-hyperset $H_1(w) = \{n_1, \ldots, n_j\}$. If for some $m \geq 2$ and $\ell \geq 0$ every data word u_j, $1 \leq j \leq \ell$, represents an $(m-1)$-hyperset, then the data word $u = {}^{b_m}_{d}\, u_1 \cdots u_\ell\, {}^{e_m}_{d'}$ represents the m-hyperset $H_m(u) = \{H_{m-1}(u_1), \ldots, H_{m-1}(u_\ell)\}$, e.g.,

$$H_2\left(\begin{smallmatrix} b_2 & b_1 & z & z & e_1 & b_1 & z & z & z & e_1 & b_1 & z & z & e_1 & e_2 \\ 2 & 1 & 1 & 2 & 2 & 3 & 7 & 9 & 8 & 2 & 1 & 2 & 5 & 9 & 3 \end{smallmatrix}\right) = \{\{1,2\},\{7,8,9\},\{2,5\}\}.$$

If u does not represent any m-hyperset, we write $H_m(u) = \perp$. For every $m \geq 1$ we define the set L_m (with the additional proposition s) as

$$L_m = \{u\, {}^{s}_{d}\, v \mid H_m(u) = H_m(v) \neq \perp, d \in \mathcal{D}\}.$$

Proposition 3.2 ([12]) *For $m \geq 4$, L_m cannot be decided by a 2-way alternating register automaton.*

This proposition can be shown along the same lines as the above mentioned results. Indeed, the result of [12] easily carries over to L_m and to the model of 2-way alternating register automata as defined in [3].

Theorem 3.3 *For each $m \geq 1$ the set L_m is definable in* HTL$_2$*.*

Proof. We define, for every $m \geq 1$, a formula $\varphi_m \in$ HTL$_2$(X, U) such that $w \models \varphi_m$ if and only if $w \in L_m$ for all data words w. The formula φ_m is a conjunction of several subformulas. The following three subformulas together express that w is of the form $u\, {}^{s}_{d}\, v$ with $H_m(u) \neq \perp$ and $H_m(v) \neq \perp$.

- A straightforward formula χ_{one} expresses that "every position carries exactly one proposition from $\{z, s, b_1, \ldots, b_m, e_1, \ldots, e_m\}$."
- "w is of the form $u\, {}^{s}_{d}\, v$, u and v start with a b_m-position and end with a e_m-position and there are no other positions carrying b_m, e_m or s."

$$\chi_{main} = b_m \wedge \mathrm{X}[\neg(b_m \vee s \vee e_m)\mathrm{U}(e_m \wedge \mathrm{X}(s \wedge \mathrm{X}(b_m \wedge (\neg(b_m \vee s \vee e_m)\mathrm{U}(e_m \wedge \neg\mathrm{X}\top)))))]$$

[5] Actually, the definitions of 2-way alternating register automata in [3] and [12] do not coincide completely. However, it is not hard to see that the result of [3] also holds for the automata of [12] over data words without propositions.

- "Every b_i-position is directly followed by a b_{i-1}- or an e_i-position, every z-position is directly followed by a z- or an e_1-position and, for $i < m$, every e_i-position is directly followed by a b_i- or an e_{i+1}-position." Here, b_0 denotes z.

$$\chi_{hyp} = \mathrm{G}(z \to \mathrm{X}(z \vee e_1)) \wedge \bigwedge_{i=1}^{m}(b_i \to \mathrm{X}(b_{i-1} \vee e_i)) \wedge \bigwedge_{i=1}^{m-1}(e_i \to \mathrm{X}(b_i \vee e_{i+1})).$$

Next, we construct a formula ψ_m that actually expressess $H_m(u) = H_m(v)$.

- The formula ψ_1 checks that, if x and y are bound to b_1-positions, the two 1-hypersets whose encodings start at x and y, respectively, are equal.

$$\begin{aligned}\psi_1 =&@x.\mathrm{X}\Big(\big(\neg e_1 \wedge {\downarrow}x.@y.\mathrm{X}(\neg e_1 \mathrm{U}(\neg e_1 \wedge \sim x))\big)\mathrm{U}e_1\Big)\wedge\\ &@y.\mathrm{X}\Big(\big(\neg e_1 \wedge {\downarrow}y.@x.\mathrm{X}(\neg e_1 \mathrm{U}(\neg e_1 \wedge \sim y))\big)\mathrm{U}e_1\Big)\end{aligned}$$

- Likewise, for $2 \leq i \leq m$ the formula ψ_i expresses that, if x and y are bound to b_i-positions, the i-hypersets starting at x and y, respectively, are equal:

$$\begin{aligned}\psi_i = &@x.\Big(\big(b_{i-1} \to {\downarrow}x.@y.(\neg e_i \mathrm{U}(b_{i-1} \wedge {\downarrow}y.\psi_{i-1}))\big)\mathrm{U}e_i\Big)\wedge\\ &@y.\Big(\big(b_{i-1} \to {\downarrow}y.@x.(\neg e_i \mathrm{U}(b_{i-1} \wedge {\downarrow}x.\psi_{i-1}))\big)\mathrm{U}e_i\Big)\end{aligned}$$

Finally, the desired formula is $\varphi_m = \chi_{one} \wedge \chi_{main} \wedge \chi_{hyp} \wedge {\downarrow}x.\mathrm{F}(s \wedge \mathrm{X}{\downarrow}y.\psi_m)$.

Every word $w = u{}^{s}_{d}v \in L_m$ satisfies χ_{one}, χ_{main} and χ_{hyp}, by construction. As u and v represent the same hypersets, both parts of ψ_m are satisfied, too.

If a data word w satisfies φ_m, the formulas χ_{one}, χ_{main} and χ_{hyp} ensure that w is of the form $u{}^{s}_{d}v$ and that u and v encode m-hypersets. The two parts of ψ_m make sure that every $(m-1)$-hyperset encoded in u also occurs in v and vice versa. Thus, the completeness and correctness of φ_m follow. □

Corollary 3.4 $\mathrm{HTL}_2 \not\leq \mathrm{LTL}^{\downarrow}$.

A closer inspection of the proof of Theorem 3.3 gives further insights. First, the formulas φ_m do not use any past operators and do not use atomic formulas of the form x. On the other hand, as already observed in [15], formulas of the form $@x.\chi$ can be replaced by formulas of the form $\mathrm{F}\mathrm{F}^{-}(x \wedge \chi)$ and thus, in the presence of past operators and atomic formulas x, there is no need for an @-operator. Thus, we get the following corollary of the proof of Theorem 3.3.

Corollary 3.5 $\mathrm{LTL}^{\downarrow}$ *cannot express all properties expressible in*

(a) $\mathrm{HTL}_2(\mathrm{X}, \mathrm{U})$ *without atomic formulas of the form* x*, and in*

(b) HTL_2 *without subformulas of the form* $@x.\chi$*.*

However, without past operators, the @-operator cannot be simulated any more. Indeed, the future fragment of HTL without the @-operator is as expressive as the future fragment of $\mathrm{LTL}^{\downarrow}$ with respect to closed formulas.

Proposition 3.6 *Every closed* HTL(X, U) *formula without the @-operator can be translated into an equivalent* $\mathrm{LTL}^{\downarrow}(\mathrm{X},\mathrm{U})$*-formula.*

Proof. For $k \geq 1$ let φ be a closed $\mathrm{HTL}_k(\mathrm{X},\mathrm{U})$ formula without any occurrences of @. Without loss of generality we assume that at most the variables $x_1, \ldots, x_k$ occur in φ. The idea is that in a formula $\downarrow x.\psi$, atomic formulas x evaluate to true only until some temporal operator has "moved" the "current position". Thus, it suffices is to keep track of whether a subformula of φ is evaluated on a position bound to a variable or not. Depending on this, atomic formulas x_i can be replaced by $\top$ or $\bot$. As an example, the formula $\downarrow x_i.\mathrm{X}x_i$ is equivalent to the formula $\downarrow x_i.\mathrm{X}\bot$.

We define a mapping on_S that translates, for every subset S of $\{x_1, \ldots, x_k\}$, $\mathrm{HTL}_k(\mathrm{X},\mathrm{U})$-formulas ψ into $\mathrm{LTL}_k^{\downarrow}(\mathrm{X},\mathrm{U})$-formulas $on_S(\psi)$ such that for every data word w, every assignment g and every position i of w it holds

$$w, g, i \models \psi \ \Leftrightarrow \ w, g, i \models on_S(\psi),$$

if S is chosen as $\{x_j \mid g(x_j) = i\}$.

The transformation is defined as follows (we omit the Boolean operators).

- $on_S(p) = p$ for propositions p
- $on_S(x_i) = \begin{cases} \top, & \text{if } x_i \in S \\ \bot, & \text{otherwise} \end{cases}$
- $on_S(\sim x_i) = \sim x_i$
- $on_S(\downarrow x_i.\chi) = \downarrow x_i.on_{S\cup\{x_i\}}(\chi)$
- $on_S(\mathrm{X}\chi) = \mathrm{X}\,on_{\emptyset}(\chi)$
- $on_S(\chi\mathrm{U}\chi') = on_S(\chi') \vee \big(on_S(\chi) \wedge \mathrm{X}(on_{\emptyset}(\chi)\mathrm{U}\,on_{\emptyset}(\chi'))\big)$

Finally, formula φ is equivalent to the $\mathrm{LTL}_k^{\downarrow}(\mathrm{X},\mathrm{U})$ formula $on_{\emptyset}(\varphi)$. □

We finally note that for HTL on data words similar observations can be made as for HTL on linear frames in [9]. In particular, for $k \geq 1$, every HTL_k formula can be converted into an initially equivalent HTL_{k+2} future formula. The idea is to bind the first additional variable, say x, to the first position of the word. A similar technique was used in [19] in the context of branching time logics.

Lemma 3.7 *For* $k \geq 1$*, every* HTL_k *formula can be converted into an initially equivalent* $\mathrm{HTL}_{k+2}(\mathrm{X},\mathrm{U})$ *formula.*

Proof. Let x and y be two additional variables. We use the following translation of HTL_k formulas φ into HTL_{k+2} formulas φ'.

- $(\mathrm{X}\psi)' = \mathrm{X}\psi'$
- $(\psi\mathrm{U}\chi)' = \psi'\mathrm{U}\chi'$
- $(\mathrm{X}^-\psi)' = \downarrow y.@x.\mathrm{F}(\mathrm{X}y \wedge \psi')$
- $(\psi\mathrm{U}^-\chi)'$: $\downarrow y.@x.\mathrm{F}(\mathrm{F}y \wedge \chi' \wedge (y \vee \mathrm{XG}(\mathrm{F}y \rightarrow \psi')))$

The trivial cases are omitted, here. Then, every HTL_k formula φ is initially equivalent to the $\mathrm{HTL}_{k+2}(\mathrm{X},\mathrm{U})$ formula $\downarrow x.\varphi'$. □

3.2 Hierarchy results

In this section, we show that both the HTL- and the LTL$^{\downarrow}$-hierarchy have infinitely many levels. More precisely, there is no $k > 0$ such that for every i, $\mathrm{HTL}_i \subseteq \mathrm{HTL}_k$ or $\mathrm{LTL}^{\downarrow}_i \subseteq \mathrm{LTL}^{\downarrow}_k$. It turns out that this can be concluded from Rossman's celebrated theorem that the variable hierarchy for first-order logic on ordered graphs is strict [14]. In the following, we denote the restriction of first-order logic to formulas with at most k variables by FO^k.

Theorem 3.8 (Rossman [14]) *For every $k \geq 1$, $\mathrm{FO}^k \lneq \mathrm{FO}^{k+1}$ on finite undirected, ordered graphs.*

We define *canonical encodings* of finite undirected ordered graphs by data words and show the following two lemmas.

Lemma 3.9 *For every formula $\varphi \in \mathrm{FO}^k$ there is a formula $\varphi' \in \mathrm{LTL}^{\downarrow}_{k+1}$ such that for every undirected ordered graph G and every canonical encoding w of G it holds $G \models \varphi \Leftrightarrow w \models \varphi'$.*

Lemma 3.10 *For every formula $\varphi' \in \mathrm{HTL}_k$ there is a formula $\varphi \in \mathrm{FO}^{2k+6}$ such that for every undirected ordered graph G and every canonical encoding w of G it holds $w \models \varphi' \Leftrightarrow G \models \varphi$.*

These two lemmas and Rossman's result yield the following theorem.

Theorem 3.11 *(a) The $\mathrm{LTL}^{\downarrow}_k$-hierarchy on data words is infinite.*

(b) The HTL_k-hierarchy on data words is infinite.

Proof. For the sake of a contradiction, let us assume that there is some i such that for every $j > i$, $\mathrm{LTL}^{\downarrow}_j = \mathrm{LTL}^{\downarrow}_i$ or such that for every $j > i$, $\mathrm{HTL}_j = \mathrm{HTL}_i$. Let φ be an arbitrary formula on ordered graphs from FO^{2i+7}. By Lemma 3.9, there is a formula $\varphi' \in \mathrm{LTL}^{\downarrow}_{2i+8}$ such that $G \models \varphi \Leftrightarrow w \models \varphi'$. By assumption, there is[6] a formula $\psi' \in \mathrm{HTL}_i$ such that for every data word w it holds $w \models \psi' \Leftrightarrow w \models \varphi'$. By Lemma 3.10, there is a formula $\psi \in \mathrm{FO}^{2i+6}$ such that for every undirected ordered graph G and every canonical encoding w of G it holds $w \models \psi' \Leftrightarrow G \models \psi$.

Thus, $\varphi \equiv \psi$. As φ was arbitrarily chosen from FO^{2i+7} we can conclude that $\mathrm{FO}^{2i+7} = \mathrm{FO}^{2i+6}$, the desired contradiction. □

It only remains to define canonical encodings and to prove the two lemmas.

In the following, we only consider graphs without self-loops. This is consistent with Rossman's Theorem. We say that a data word w is a *canonical encoding* of a finite ordered undirected graph $G = (V, E, <)$ if the following conditions hold.

- w has $|V| + 2|E|$ positions,
  - · a *node position* $p(u)$, for every $u \in V$ and
  - · an *edge position* $q(u, v)$, for every[7] edge (u, v).

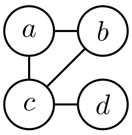
[6] If we assume that the LTL$^{\downarrow}$-hierarchy collapses then we get a formula $\psi' \in \mathrm{LTL}^{\downarrow}_i$ at this point which is, however, also in HTL_i.

[7] Every edge (u, v) gives rise to two positions, $q(u, v)$ and $q(v, u)$.

- Node positions $p(u)$ have a unique data value and carry the proposition p.
- For every u, v, $(u, v) \in E$, the edge positions $q(u, v)$ and $q(v, u)$ have the same data value (and this value does not occur otherwise) and do not carry any propositions.
- The order of the positions obeys the following rules, for every $u, u', v, v' \in V$:

· $p(u) < p(v)$ if $u < v$ in G. · $q(u, v) < p(u')$ if $u < u'$.
· $p(u) < q(u', v')$ if $u \leq u'$. · $q(u, v) < q(u, v')$ if $v < v'$.

It should be noted that these rules define a unique order on every canonical data string w.

Example 3.12 If in the undirected finite graph $G =$  the nodes are ordered by $a < b < c < d$, then $\begin{array}{cccccccccccc} p & \emptyset & \emptyset & p & \emptyset & \emptyset & p & \emptyset & \emptyset & \emptyset & p & \emptyset \\ 7 & 8 & 3 & 2 & 8 & 1 & 6 & 3 & 1 & 5 & 4 & 5 \end{array}$ is a canonical encoding of G. It should be observed that the underlying linear order of data values $(1 < 2 < 3 < \cdots)$ is not relevant for the encoding.

A maximal subword in a canonical encoding where only the first position is labeled by p is called a *variable block*. As an example, the subword from the 4th to the 6th position of the above encoding constitutes a variable block.

Proof of Lemma 3.9. We assume without loss of generality that formulas from FO^k only use variables $x_1, \ldots, x_k$. The translation $\varphi \mapsto \varphi'$ can be defined inductively as follows.

- $(x_i = x_j)' = \mathrm{FF}^-(\sim x_i \wedge \sim x_j)$
- $(x_i < x_j)' = \mathrm{FF}^-(\sim x_i \wedge \mathrm{XF}{\sim} x_j)$
- $E(x_i, x_j)' = \mathrm{FF}^-(\sim x_i \wedge \big(\neg p \mathrm{U}{\downarrow} y.\mathrm{FF}^-(\sim y \wedge \neg p \mathrm{U}^-(\sim x_j \wedge \not\sim x_i))\big)$
- $(\exists x_i \psi)' = \mathrm{FF}^- {\downarrow} x_i.(p \wedge \psi')$
- $(\neg\psi)' = \neg\psi'$
- $(\psi \wedge \chi)' = \psi' \wedge \chi'$

In φ', variables x_i are always only bound to first positions of variable blocks. Whether two nodes bound to x_i and x_j are connected by an edge can then be tested by checking whether the blocks starting at x_i and x_j, respectively, share some data value. □

Proof of Lemma 3.10. The proof consists of two steps. First, we construct, as in the proof of Theorem 3.1, an FO^{k+3}-formula φ'' that is equivalent to φ' on data words.

Then we transform φ'' into φ by means of a quantifier-free logical interpretation [6, Section 11.2] that defines, for every finite ordered undirected graph $G = (V, E, <)$, a (unique) representation of the canonical encodings of G on the set $V \times V$. However, the translation of φ'' into φ requires two variables, x_i, x_i', for every variable

x_i of φ'', thus resulting in a formula with $2k+6$ variables. More precisely, the logical interpretation $\Phi = (\varphi_U, \varphi_p, \varphi_<, \varphi_\sim)$ is defined as follows.

- $\varphi_U(x_1, x_2)$ defines the set of pairs that are actual positions of the representation of the canonical encodings. Thus, it is just $x_1 = x_2 \vee E(x_1, x_2)$.
- $\varphi_p(x_1, x_2)$ defines the positions that carry the proposition p and is thus just $x_1 = x_2$.
- $\varphi_<(x_1, x_2, y_1, y_2)$ defines the linear order on positions. It is $(x_1 = x_2 \wedge (x_1 < y_1 \vee (x_1 = y_1 \wedge x_2 \neq y_2)))\vee$ $(x_1 \neq x_2 \wedge (x_1 < y_1 \vee (x_1 = y_1 \wedge x_2 < y_2)))$.
- Finally, $\varphi_\sim(x_1, x_2, y_1, y_2)$ defines the pairs of positions that have the same data value. It is simply $x_1 = y_2 \wedge x_2 = y_1$.

It is not hard to see that Φ indeed defines, for every G, the unique representation of the canonical encodings of G. □

Remark 3.13 We conjecture that both hierarchies are even strict and that this can be shown in a similar way as the strictness of the FO^k-hierarchy can be concluded from its infinity [14]. However, we did not have time to figure out the details before the deadline for this submission.

3.3 Succinctness

We have seen before that HTL$_2$ can express properties that cannot be expressed by any LTL$^\downarrow$-formula. We show next that there are also LTL$^\downarrow$-expressible properties that can be expressed non-elementarily more succinct in HTL$_2$. This is basically a corollary from results in [17], [16] and [15] and the observation that on data words in which all positions carry the same data value, LTL$^\downarrow$ is expressively equivalent to PLTL.

In [17] it is shown that there are star-free regular expressions $(\alpha_n)_{n\geq 1}$, built from union, concatenation, and negation, such that, there is no elementary function f for which $f(n)$ bounds the length of the size of the smallest string satisfying α_n, for every $n \geq 1$. In [8] it is explained how for every star-free regular expression α one can build an equivalent FO formula. Following a similar technique, [15] gives a translation from star-free regular expressions to hybrid logic formulas. In our setting, the translation yields HTL-formulas of linear size in $|\alpha_n|$. On the other hand, [16] proves that every satisfiable LTL formula ψ can be satisfied by a string of length at most exponential in $|\psi|$. Combining these results we get:

Proposition 3.14 HTL$_2$(X, F) *is non-elementarily more succinct than* LTL$^\downarrow$.

4 One Variable

In this section, we show our main result, that HTL$_1 \equiv$ LTL$^\downarrow_1$ on data words. Further, we prove that HTL$_1$(F) is exponentially more succinct than LTL$^\downarrow$.

4.1 Expressiveness

The translation of HTL_1 to $\mathrm{LTL}_1^{\downarrow}$ relies on a kind of separation property. We show that if $\varphi = \downarrow x.\chi$ then the "top level" of χ can be rewritten into a Boolean combination of future and past formulas.

We introduce some new notation for the proof.

For a data word w, a position i of w and a set Φ of HTL_1-formulas we denote by $(w,i)^{\Phi}$ the string that is obtained from w by removing the data values and adding to each position j all propositions p_ψ, $\psi \in \Phi$, for which $w,i,j \models \psi$.

A subformula ψ of a HTL_1-formula φ is called a *top-level subformula* of φ if ψ is not in the scope of any $\downarrow x$ quantifier.

For every HTL_1-formula φ we let $\overrightarrow{T}(\varphi)$ be the set of all top-level subformulas of φ that are of one of the forms $\downarrow x.\chi$, $\sim x$ or $@x.\chi$ and of all formulas $@x.\mathrm{X}^-\chi$ and $@x.\chi\mathrm{U}^-\theta$ for which $\mathrm{X}^-\chi$ or $\chi\mathrm{U}^-\theta$, respectively, is a top-level subformula of φ. We define $\overrightarrow{T_x}(\varphi)$ analogously, but with the additional atomic formula x.

If j is a position of a data word w we write $w[j,\ldots)$ for the subword of w starting at position j.

Lemma 4.1 *For every* HTL_1*-formula* φ *there is an* LTL*-formula* $\overrightarrow{\varphi}$ *such that for every data word* w *and all positions* i,ℓ *of* w *it holds*

$$w,\ell,i \models \downarrow x.\varphi \;\Leftrightarrow\; (w,i)^{\overrightarrow{T}(\varphi)}, i \models \overrightarrow{\varphi}.$$

The size of $\overrightarrow{\varphi}$ *is at most triply exponential in* $|\varphi|$.

Proof. We inductively define, for every top-level subformula ψ of φ, a PLTL-formula $\widetilde{\psi}$ such that, for all positions i,j with $j \geq i$, it holds

$$w,i,j \models \psi \;\Leftrightarrow\; (w,i)^{\overrightarrow{T_x}(\varphi)}, j \models \widetilde{\psi}. \tag{1}$$

To this end, let $\widetilde{\psi}$ be

- p if ψ is a proposition p;
- p_ψ if ψ is of one of the forms x, $\sim x$, $\downarrow x.\chi$, $@x.\chi$;
- $\mathrm{X}\widetilde{\chi}$ if $\psi = \mathrm{X}\chi$;
- $\widetilde{\chi}\mathrm{U}\widetilde{\theta}$, if $\psi = \chi\mathrm{U}\theta$;
- $(\neg p_x \wedge \mathrm{X}^-\widetilde{\chi}) \vee (p_x \wedge p_{@x.\mathrm{X}^-\chi})$ if $\psi = \mathrm{X}^-\chi$;
- $(\neg p_x \wedge \widetilde{\chi})\mathrm{U}^-((\neg p_x \wedge \widetilde{\theta}) \vee (p_x \wedge p_{@x.\chi\mathrm{U}^-\theta}))$ if $\psi = \chi\mathrm{U}^-\theta$.

That is, the usual evaluation of past operators is restricted to positions $\geq i$. If this is insufficient then the new propositions are used. It is straightforward to show by induction that Equation 1 indeed holds.

Let now φ_1 be the formula that results from $\widetilde{\varphi}$ by replacing every occurrence of p_x with $\neg\mathrm{X}^-\top$. Clearly, for all data words w and positions $i \leq j$, $(w,i)^{\overrightarrow{T_x}(\varphi)}, j \models \widetilde{\varphi}$ if and only if $(w,i)^{\overrightarrow{T}(\varphi)}[i,\ldots),(j-i+1) \models \varphi_1$.

By [10, Theorem 2.4] the PLTL-formula φ_1 can be effectively translated into an LTL-formula $\overrightarrow{\varphi}$ that is initially equivalent to φ_1. Moreover, by [11] it follows that $\overrightarrow{\varphi}$ can be computed in triply exponential time. As the positions $< i$ are irrelevant for the validity of $\overrightarrow{\varphi}$ at position i, altogether the lemma follows. □

Similarly, we let, for every HTL$_1$-formula φ, $\overleftarrow{T}(\varphi)$ be the set of all top-level subformulas of φ that are of one of the forms $\downarrow x.\chi$, $\sim x$, $@x.\chi$ and of all formulas $@x.\mathrm{X}\chi$ and $@x.\chi\mathrm{U}\theta$ for which $\mathrm{X}\chi$ or $\chi\mathrm{U}\theta$, respectively, is a top-level subformula of φ. Analogously we get the following lemma.

Lemma 4.2 *For every* HTL$_1$*-formula* φ *there is a* PLTL*-formula* $\overleftarrow{\varphi}$ *which does not use any future operator such that for every data word* w *and all positions* i, ℓ *of* w *it holds*

$$w, \ell, i \models \downarrow x.\varphi \;\Leftrightarrow\; (w,i)^{\overleftarrow{T}(\varphi)}, i \models \overleftarrow{\varphi}.$$

The size of $\overleftarrow{\varphi}$ *is at most triply exponential in* $|\varphi|$.

Theorem 4.3 *Every closed* HTL$_1$ *formula can be translated into an equivalent* LTL$_1^\downarrow$ *formula of at most triply exponential size.*

Proof. It suffices to prove that for every HTL$_1$ formula φ there is a LTL$_1^\downarrow$ formula φ' of triply exponential size with $\downarrow x.\varphi \equiv \downarrow x.\varphi'$. We show this by induction on the size of φ. Thanks to the equivalences

- $\downarrow x.p \equiv p$,
- $\downarrow x.x \equiv \top$,
- $\downarrow x.\sim x \equiv \top$,
- $\downarrow x.@x.\psi \equiv \downarrow x.\psi$,
- $\downarrow x.\neg\psi \equiv \neg\downarrow x.\psi$,
- $\downarrow x.(\psi \wedge \chi) \equiv \downarrow x.\psi \wedge \downarrow x.\chi$

we can assume that φ is of one of the forms (1) $\psi\mathrm{U}\chi$, (2) $\mathrm{X}\psi$, (3) $\psi\mathrm{U}^-\chi$, (4) $\mathrm{X}^-\psi$.

We first consider the cases (1) and (2). Let $\overrightarrow{\varphi}$ be as guaranteed by Lemma 4.1. Thus, for every data word w and all positions i, ℓ of w, it holds

$$w, \ell, i \models \downarrow x.\varphi \;\Leftrightarrow\; (w,i)^{\overrightarrow{T}(\varphi)}, i \models \overrightarrow{\varphi}.$$

Clearly, replacing every atomic formula $p_{\sim x}$ by $\sim x$ in $\overrightarrow{\varphi}$ and evaluating the formula in the data word resulting from $(w,i)^{\overrightarrow{T}(\varphi)}[i,\ldots)$ by putting back the data values from w does not change the validity of $\overrightarrow{\varphi}$. Likewise, replacing every atomic formula $p_{\downarrow x.\psi}$ by the formula $\downarrow x.\psi'$, where ψ' is a LTL$_1^\downarrow$-formula equivalent to ψ obtained by induction, does not change the validity either. We denote the resulting formula by $\widehat{\varphi}$.

It only remains to eliminate atomic formulas of the kind $p_{@x.\psi}$ from $\widehat{\varphi}$. For every assignment $\alpha : \overrightarrow{T}(\varphi) \to \{\top, \bot\}$ let φ_α be the formula resulting from $\widehat{\varphi}$ by replacing every occurrence of an atomic formula $p_{@x.\psi}$ with $\alpha(@x.\psi)$. We finally let

$$\varphi' = \bigvee_{\alpha:\overrightarrow{T}(\varphi)\to\{\top,\bot\}} \varphi_\alpha \wedge \Big(\bigwedge_{\substack{@x.\psi\in\overrightarrow{T}(\varphi)\\ \alpha(@x.\psi)=\top}} \psi' \wedge \bigwedge_{\substack{@x.\psi\in\overrightarrow{T}(\varphi)\\ \alpha(@x.\psi)=\bot}} \neg\psi' \Big),$$

where ψ' is again a $\mathrm{LTL}_1^{\downarrow}$-formula equivalent to ψ obtained by induction that does not use any additional propositions anymore.

The cases (3) and (4) are completely analogous.

The size of the resulting formula is dominated by the (at most) triply exponential size of $\overrightarrow{\varphi}$. The elimination of @ only contributes an exponential factor and we end up with a formula of at most triply exponential size. □

4.2 Succinctness

Even though $\mathrm{HTL}_1 \equiv \mathrm{LTL}_1^{\downarrow}$, HTL_1 can express some properties exponentially more succinct than $\mathrm{LTL}_1^{\downarrow}$ and, actually, even $\mathrm{LTL}^{\downarrow}$.

Proposition 4.4 $\mathrm{HTL}_1(\mathrm{F})$ *is exponentially more succinct than* $\mathrm{LTL}^{\downarrow}$.

Proof. The proof essentially follows the proof of [8, Theorem 3 (1)] that FO^2 is exponentially more succinct than *unary* LTL. Let E_n be the property *"Any two positions of the word that agree on propositions $p_1, p_2, \ldots, p_n$ also agree on proposition p_0."* and let L_n be the set of data words fulfilling E_n in which all positions have the same data value. For every $n \geq 1$, L_n is expressed by the following $\mathrm{HTL}_1(\mathrm{F})$ formula of length $\mathcal{O}(n)$:

$$\mathrm{G}{\downarrow}x.\mathrm{G} \sim x \wedge \mathrm{G}[{\downarrow}x.\mathrm{G}(\textstyle\bigwedge_{i=1}^{n}(p_i \leftrightarrow @x.p_i) \rightarrow (p_0 \leftrightarrow @x.p_0))].$$

Let us assume now that, for every n, there is a $\mathrm{LTL}^{\downarrow}$-formula ψ_n expressing L_n and $|\psi_n| = 2^{o(n)}$. This formula can be translated into a formula χ_n of roughly the same size that expresses E_n on words without data. Similarly as in [18], there is, for every n, a non-deterministic automaton for χ_n of size $2^{|\chi_n|} = 2^{2^{o(n)}}$. However, it can be shown as in [8] that every automaton for E_n requires at least 2^{2^n} states, the desired contradiction. □

5 Discussion

In this paper, we compared the expressive power of hybrid temporal logic on data words with $\mathrm{LTL}^{\downarrow}$. Although HTL is more powerful in general, the two logics coincide if only one variable is allowed.

The main results of this paper carry over to data words of infinite length. For Corollary 3.4 and Theorem 3.11 this simply holds because the separating languages can be easily turned into ω-languages by padding with an infinite number of positions. The generalization of Theorem 4.3 to infinite data words is also straightforward. Here, it is important that the result of [11] was already shown for ω-strings.

Clearly, all considered logics have an undecidable satisfiability problem (for $\mathrm{LTL}_1^{\downarrow}$ this was shown in [3]). However, the model checking remains largely unexplored, especially for the case of infinite data words.

As already mentioned in Section 3, we conjecture that the variable-hierarchy for HTL and $\mathrm{LTL}^{\downarrow}$ are both strict.

References

[1] Carlos Areces, Patrick Blackburn, and Maarten Marx. The computational complexity of hybrid temporal logics. *Logic Journal of the IGPL*, 8(5):653–679, 2000.

[2] Mikolaj Bojanczyk, Anca Muscholl, Thomas Schwentick, Luc Segoufin, and Claire David. Two-variable logic on words with data. In *LICS*, pages 7–16. IEEE Computer Society, 2006.

[3] Stephane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. In *LICS '06: Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science*, pages 17–26, Washington, DC, USA, 2006. IEEE Computer Society.

[4] Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3), 2009. Full version of [3].

[5] Stéphane Demri, Ranko Lazić, and David Nowak. On the freeze quantifier in constraint LTL: Decidability and complexity. *Inf. Comput.*, 205(1):2–24, 2007.

[6] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory.* Springer, Heidelberg, 2005.

[7] E. Allen Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 995–1072. 1990.

[8] Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Information and Computation*, 179(2):279 – 295, 2002.

[9] Massimo Franceschet, Maarten de Rijke, and Bernd-Holger Schlingloff. Hybrid logics on linear structures: Expressivity and complexity. In *TIME*, pages 166–173, 2003.

[10] Dov M. Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In *Temporal Logic in Specification*, pages 409–448, 1987.

[11] Nicolas Markey. Temporal logic with past is exponentially more succinct, concurrency column. *Bulletin of the EATCS*, 79:122–128, 2003.

[12] Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004.

[13] Arthur Prior. *Past, Present, and Future.* Oxford University Press, 1967.

[14] Benjamin Rossman. On the constant-depth complexity of k-clique. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 721–730, New York, NY, USA, 2008. ACM.

[15] Thomas Schwentick and Volker Weber. Bounded-variable fragments of hybrid logics. In *STACS*, pages 561–572, 2007.

[16] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985.

[17] L. Stockmeyer. The complexity of decision problems in automata and logic, 1974. Ph.D. Thesis, MIT, 1974.

[18] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *LICS*, pages 332–344. IEEE Computer Society, 1986.

[19] Volker Weber. Branching-time logics repeatedly referring to states. *Journal of Logic, Language and Information*, 18(4):593–624, 2009.