Temporal Logics on Strings with Prefix Relation

Stéphane Demri, Morgan Deters

Temporal Logics on Strings with Prefix Relation

Stéphane Demri[1] and Morgan Deters[2]

[1] LSV, CNRS, France [2] New York University, USA

Abstract. We show that linear-time temporal logic over concrete domains made of finite strings and the prefix relation admits a PSpace-complete satisfiability problem. Actually, we extend a known result with the concrete domain made of the set of natural numbers and the greater than relation (corresponding to the singleton alphabet case) and we solve an open problem mentioned in several publications. Since the prefix relation is not a total ordering, it is not possible to take advantage of existing techniques dedicated to temporal logics with concrete domains that are essentially linearly ordered structures. Instead, we introduce an adequate encoding of string constraints into length constraints that allows us to reduce the problem on strings to the problem on natural numbers. To do so, we also propose an extended version of the logic on strings that is able to compare lengths of longest common prefixes and for which the satisfiability problem is shown in PSpace. Finally, we show how to lift the result for the branching-time case in order to get decidability when the underlying temporal logic is CTL*.

1 Introduction

String theories in a temporal setting. Reasoning about strings is increasingly required in program verification or in analysis of web applications and recently much effort has been dedicated toward designing solvers that handle string theories (possibly in combination with other theories), see e.g. [GKA$^+$11,ZZG13,LRT$^+$14,AAC$^+$14,HL14,Lia14]. This also includes works related to first-order theory of strings with order, see e.g. [Kus06]. The decidability status of expressive string theories is not always known, see e.g. [LRT$^+$14, Section 2.1] or [AAC$^+$14], but fortunately, decidability of word equations is known to be decidable thanks to Makanin's result [Mak77] and a PSpace algorithm has been designed by Plandowski in [Pla04] (see also [DJP14]). By contrast, first-order theory on strings is undecidable by Quine's undecidability result [Qui46].

At the same time, many works have been dedicated to reasoning about temporal logics on concrete domains, see e.g. [BC02,Lut04,BG06,BGL12,DHV14], so that temporal reasoning is done about the evolution of typed variables (for instance interpreted by integers or by strings to cite a few examples). Hence, fibring a temporal logic with a concrete domain happens to be a very natural approach, see e.g. [Gab98], but this may easily lead to undecidability, see e.g. [DD07, Section 10]. Even when decidability is preserved, computational complexity can be high. For instance, while ALC-LTL satisfiability with rigid names is 2ExpTime-complete [BGL12], LTL over the concrete domain $(\mathbb{N}, \leq)$ is only PSpace-complete [DD07,DG08]. In this paper, we are interested in temporal logics when the concrete domain contains finite strings.

Our motivation. Linear-time temporal logics on concrete domains have been shown decidable mainly when the underlying concrete domain has the completion property, see e.g. [Dec92,BC02,LM05,DD07,Gas09] or when it is a linearly ordered structure [DD07,ST11].

Branching-time extensions have also been considered in [Čer94,BG06,Gas07,Gas09,CKL13] and a remarkable breakthrough has been made recently in [CKL13] by showing that CTL* over the domain $(\mathbb{N}, \leq)$ is decidable by using the decidability of Boolean combinations of formulae from MSO and from WMSO+U [BT12] where U is the unbounding second-order quantifier (see also the follow-up work [CKL14] involving ECTL*). Unfortunately, none of the known techniques has been able to handle LTL over concrete domain of the form $(\Sigma^*, \preceq)$ where $\preceq$ is not a total ordering on Σ^* such as the prefix relation $\preceq_p$ or the subword relation $\sqsubseteq$. This is surprising since the prefix relation $\preceq_p$ seems quite harmless whereas the subword relation $\sqsubseteq$ is a well-quasi-ordering and therefore one could expect decidability by Higman's Lemma. Actually, the fact that the decidability status of the satisfiability problem for LTL over $(\{0,1\}^*, \preceq_p)$ is open first appeared in [DG05] and it has been reformulated for the branching-time version based on CTL* in [CKL13], see also [ST11]. Even worse, in [CKL14, Section 9], it is stated that a new strategy is needed to solve the satisfiability problem for LTL over $(\{0,1\}^*, \preceq_p)$. In this work, we aim to better understand how to solve the satisfiability problem for temporal logics over concrete domains that are of the form $(\Sigma^*, \preceq)$ when $\preceq$ is not necessarily linearly ordered. More specifically, we focus our attention on the prefix relation.

Our contribution. In this paper, we write LTL$(\Sigma^*, \preceq_p)$ to denote LTL over the concrete domain $(\Sigma^*, \preceq_p)$ (see a formal definition in Section 2). Our main result is that for every (finite or infinite and countable) non-empty alphabet Σ, the satisfiability problem for LTL$(\Sigma^*, \preceq_p)$ is PSPACE-complete. It is worth noting that the decidability status of these logics was open until now for $\mathrm{card}(\Sigma) \geq 2$ (the case $k = 1$ naturally corresponds to LTL over the concrete domain $(\mathbb{N}, \leq)$, written LTL$(\mathbb{N}, \leq)$). We adopt an original approach since prefix constraints are first generalized and then translated into numerical constraints so that we are able to design a satisfiability-preserving reduction into LTL$(\mathbb{N}, \leq)$, and then use the PSPACE upper bound from [DD07,DG08] (see also [ST11]). The PSPACE upper bound also holds for many extensions of LTL as soon as they behave as LTL for the translation of formulae into Büchi automata (see Section 4.1). Solving string constraints by formulae in Presburger arithmetic is not a new idea (see e.g. [Pla06]) but in the paper we provide a straightforward method to deal with temporal logics on finite strings with the prefix relation. Moreover, in the technical developments, we have considered a substantial extension of LTL$(\Sigma^*, \preceq_p)$, namely LTL$(\Sigma^*, \mathrm{clen})$, for which our technique can be applied naturally.

As by-products of our method, we are also able to establish a small alphabet property when $\Sigma = \mathbb{N}$ (see Corollary 1) and more importantly, the satisfiability problem for CTL$^\star(\Sigma^*, \mathrm{clen})$, the branching-time extension of LTL$(\Sigma^*, \mathrm{clen})$ based on CTL*, is decidable by taking advantage of the decidability of CTL$^\star(\mathbb{N}, \leq)$ recently shown in [CKL13]. As for the linear-time case, we provide a reduction from CTL$^\star(\Sigma^*, \mathrm{clen})$ to CTL$^\star(\mathbb{N}, \leq)$ by encoding length constraints about longest common prefixes into counters. It is worth noting that the decidability results in [CKL13] are based on the decidability of Boolean combinations of formulae from MSO and from WMSO+U [BT12] where U is the unbounding second-order quantifier, see e.g. [Boj04,BC06] (note that full MSO+U has been shown recently undecidable in [BPT15]). Nevertheless, to get advantage of this result on monadic second-order

logics, a property for $(\mathbb{Z}, \leq)$ needs to be established, see e.g. [CKL13, Lemma 15] or an earlier analogous formulation in [DD07, Lemma 6.1] or in [Čer94, Lemma 5.5] and therefore, our result for $\mathrm{CTL}^\star(\Sigma^*, \mathrm{clen})$ requires the decidability results from [BT12,CKL13]. We wonder whether the (complexity/decidability) results established in the present paper can be simply explained by an approach using the definability of existence of homomorphisms in a monadic second-order logic, as done in [CKL13,CKL14] (this would require to provide a variant to this approach). Our translations have some flavor of the notion of interpretability from [CKL13], but this does not fit completely, as far as we can judge.

At the time of completing this paper, we have been aware of the work [KW15] that has been done independently and that answers similar questions by using techniques for constraint automata.

2 Preliminaries

2.1 LTL on finite strings with the prefix relation

Let $\mathrm{SVAR} = \{\mathtt{x}_1, \mathtt{x}_2, \ldots\}$ be a countably infinite set of **string variables**. Given a non-empty alphabet Σ, we write $\mathrm{LTL}(\Sigma^*, \preceq_p)$ to denote the corresponding version of LTL over the concrete domain $(\Sigma^*, \preceq_p)$ where $\preceq_p$ is the prefix relation. Given two words w, w' in Σ^*, we write $w \preceq_p w'$ whenever w is a prefix of w', i.e., there is $w'' \in \Sigma^*$ such that $w \cdot w'' = w'$.

Terms in the logic $\mathrm{LTL}(\Sigma^*, \preceq_p)$ are defined as follows:

$$\mathtt{t} \quad ::= \quad w \mid \mathtt{x} \mid \mathtt{X}^i\mathtt{x}$$

where $w \in \Sigma^*$, $\mathtt{x} \in \mathrm{SVAR}$ and $\mathtt{X}^i\mathtt{x}$ is the sequence of $i \geq 1$ occurrences of the symbol $\mathtt{X}$ followed by the string variable $\mathtt{x}$. We write ε to indicate the empty string. The term $\mathtt{X}^i\mathtt{x}$ is interpreted as the value of $\mathtt{x}$, ith position ahead of the current position.

Formulae in the logic $\mathrm{LTL}(\Sigma^*, \preceq_p)$ are defined as follows:

$$\phi \quad ::= \quad \mathtt{t} \preceq_p \mathtt{t}' \mid \neg\phi \mid \phi \wedge \phi \mid \mathtt{X}\phi \mid \phi \,\mathtt{U}\, \phi$$

Hence, $\mathrm{LTL}(\Sigma^*, \preceq_p)$ contains the usual temporal connectives from LTL. The symbol $\mathtt{X}$ is overloaded since it is used in terms of the form $\mathtt{X} \cdots \mathtt{X}\mathtt{x}$ and in formulae of the form $\mathtt{X}\phi$. This will not cause any confusion but we keep the current usage, since in both cases, the presence of '$\mathtt{X}$' refers to future positions of bounded distance. Standard abbreviations related to Boolean or temporal connectives are also used in the rest of this paper (for $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\mathtt{F}$, $\mathtt{G}$, etc.). Here is an example of a formula in $\mathrm{LTL}(\{0,1\}^*, \preceq_p)$:

$$\mathtt{GF}((001 \preceq_p \mathtt{x}) \vee (\mathtt{x} \preceq_p 1001)) \wedge \mathtt{G}(\neg(\mathtt{x} \preceq_p \mathtt{Xx})).$$

A **string valuation** $\mathfrak{s}$ is a map of the form $\mathfrak{s} : \mathrm{SVAR} \to \Sigma^*$ (or a restriction to a subset of SVAR), and a **model** for $\mathrm{LTL}(\Sigma^*, \preceq_p)$ is an ω-sequence σ of string valuations. For every model σ and $i \in \mathbb{N}$, the **satisfaction relation** $\models$ is defined as follows (standard Boolean clauses are omitted):

– $\sigma, i \models \mathtt{t} \preceq_p \mathtt{t}' \stackrel{\text{def}}{\Leftrightarrow} [\![\mathtt{t}]\!]_i \preceq_p [\![\mathtt{t}']\!]_i$ where

$$\begin{array}{lcll} [\![w]\!]_i & \stackrel{\text{def}}{=} & w & \text{for all } w \in \Sigma^* \\ [\![\mathtt{x}]\!]_i & \stackrel{\text{def}}{=} & \sigma(i)(\mathtt{x}) & \text{for all } \mathtt{x} \in \text{SVAR} \\ [\![\mathtt{X}^j\mathtt{x}]\!]_i & \stackrel{\text{def}}{=} & \sigma(i+j)(\mathtt{x}) & \text{for all } \mathtt{x} \in \text{SVAR and } j \geq 1 \end{array}$$

– $\sigma, i \models \mathtt{X}\phi \stackrel{\text{def}}{\Leftrightarrow} \sigma, i+1 \models \phi$,
– $\sigma, i \models \phi \ \mathtt{U} \ \psi \stackrel{\text{def}}{\Leftrightarrow}$ there exists $j \geq i$ such that $\sigma, j \models \psi$ and for all $i \leq k < j$, we have $\sigma, k \models \phi$.

Note that the temporal operators have their usual semantics and for a given formula ϕ built over the finite set of string variables $\{\mathtt{x}_1, \ldots, \mathtt{x}_q\}$, we can restrict ourselves to $\{\mathtt{x}_1, \ldots, \mathtt{x}_q\}$ to evaluate the formula on a model. Similarly, in the sequel, we use the abbreviation $\mathtt{t} = \mathtt{t}'$ instead of $(\mathtt{t} \preceq_p \mathtt{t}') \wedge (\mathtt{t}' \preceq_p \mathtt{t})$ and the abbreviation $\mathtt{t} \prec_p \mathtt{t}'$ instead of $\neg(\mathtt{t} = \mathtt{t}') \wedge \mathtt{t} \preceq_p \mathtt{t}'$. Unsurprisingly, we have that $\sigma, i \models \mathtt{t} = \mathtt{t}'$ iff $[\![\mathtt{t}]\!]_i = [\![\mathtt{t}']\!]_i$.

An important feature of the semantics for LTL$(\Sigma^*, \preceq_p)$ is the ability to state properties for strings at the current state but also for strings at future positions, which may propagate constraints all over the model with the help of the temporal operator $\mathtt{G}$.

As usual, an instance of the **satisfiability problem** for LTL$(\Sigma^*, \preceq_p)$ consists in checking whether a formula ϕ in LTL$(\Sigma^*, \preceq_p)$ admits a model σ such that $\sigma, 0 \models \phi$. The **validity problem** is defined dually, as usual. By way of example, the formula below is valid in LTL$(\{0,1\}^*, \preceq_p)$:

$$\mathtt{G}(\mathtt{x} \preceq_p 1001 \Leftrightarrow (\mathtt{x} = \varepsilon \vee \mathtt{x} = 1 \vee \mathtt{x} = 10 \vee \mathtt{x} = 100 \vee \mathtt{x} = 1001)).$$

We write LTL$(\mathbb{N}, \leq)$ to denote LTL$(\Sigma^*, \preceq_p)$ when Σ is a singleton alphabet. Terms in the logic LTL$(\mathbb{N}, \leq)$ have one of the forms below: $n \in \mathbb{N}$, $\mathtt{x}$ or $\mathtt{X}^i\mathtt{x}$ with $i \geq 1$ and $\mathtt{x} \in$ SVAR. For example, $\mathtt{GF}(\mathtt{Xx} < \mathtt{x})$ is satisfiable in LTL$(\mathbb{N}, \leq)$ whereas $\mathtt{FG}(\mathtt{Xx} < \mathtt{x})$ is not satisfiable. By contrast, both $\mathtt{GF}p$ and $\mathtt{FG}p$ are satisfiable in standard LTL.

Unlike $(\mathbb{R}, \leq)$, the concrete domain $(\mathbb{N}, \leq)$ does not satisfy the completion property from [DD07] (see also similar properties in [Dec92,BC02,LM05]), which does not help to decide LTL$(\mathbb{N}, \leq)$. Indeed, LTL augmented with a concrete domain satisfying the completion property can be decided with Büchi automata, since the sets of satisfiable symbolic models are ω-regular [DD07, Lemma 4.3]. However, the set of satisfiable symbolic models in LTL$(\mathbb{N}, \leq)$ is not ω-regular [DD07, Corollary 6.5] and therefore the automata-based approach to solve LTL$(\mathbb{N}, \leq)$ is certainly not a routine exercise. Still, it is possible to decide LTL$(\mathbb{N}, \leq)$ with the help of Büchi automata, see details in [DD07] even though the proper extension of MSO that can express satisfiable symbolic models is presented in [CKL13].

Proposition 1. *[DD07,DG08] The satisfiability problem for* LTL$(\mathbb{N}, \leq)$ *is* PSPACE*-complete.*

The way we define LTL$(\mathbb{N}, \leq)$ from LTL$(\Sigma^*, \preceq_p)$ implies a unary encoding of natural numbers in the formulae but the PSPACE upper bound also holds with a binary encoding, as considered in [DG08].

So far, we have not assumed much about the alphabet Σ. When $\mathrm{card}(\Sigma) = k$ for some $k \geq 1$, we can assume that $\Sigma = [0, k-1]$. In the sequel, we assume that $\mathrm{card}(\Sigma) \geq 2$ since the case $k = 1$ corresponds to the logic $\mathrm{LTL}(\mathbb{N}, \leq)$. When Σ is a countably infinite alphabet, without any loss of generality, we can assume that $\mathbb{N} \subseteq \Sigma$. However, it is sufficient to assume that $\Sigma = \mathbb{N}$ thanks to the simple property below.

Lemma 1. *Let Σ be a countably infinite alphabet. There is a logarithmic-space reduction from the satisfiability problem for $\mathrm{LTL}(\Sigma^*, \preceq_p)$ into the satisfiability problem for LTL($\mathbb{N}^*, \preceq_p$).*

Proof. Since Σ is countably infinite, there is a bijection $\mathfrak{f}^\star : \Sigma \to \mathbb{N}$. Let ϕ be a formula in $\mathrm{LTL}(\Sigma^*, \preceq_p)$ built over words in Σ^* such that the letters $a_1, \ldots, a_k$ occur in ϕ. Let ϕ' be the formula obtained from ϕ by replacing every occurrence of the letter a_i by the natural number i. This provides a reduction that can be easily implemented in logarithmic space. It remains to show that ϕ is satisfiable in $\mathrm{LTL}(\Sigma^*, \preceq_p)$ iff ϕ' is satisfiable in $\mathrm{LTL}(\mathbb{N}^*, \preceq_p)$.

Let $\mathfrak{f}$ be a bijection $\mathfrak{f} : \Sigma \to \mathbb{N}$. We write $\mathfrak{f}(\phi)$ to denote the formula obtained from ϕ by substituting every letter a by $\mathfrak{f}(a)$. Similarly, we write $\mathfrak{f}(\sigma)$ to denote the model in $\mathrm{LTL}(\mathbb{N}^*, \preceq_p)$ such that for all $i, j \geq 0$, $\mathfrak{f}(\sigma)(i)(\mathsf{x}_j)$ is obtained from $\sigma(i)(\mathsf{x}_j)$ by substituting every letter a by $\mathfrak{f}(a)$

Given two bijections $\mathfrak{f} : \Sigma \to \mathbb{N}$ and $\mathfrak{g} : \mathbb{N} \to \Sigma$, the following properties can be easily shown:

1. for all $\mathrm{LTL}(\Sigma^*, \preceq_p)$ models σ, we have $\sigma, 0 \models \phi$ iff $\mathfrak{f}(\sigma), 0 \models \mathfrak{f}(\phi)$.
2. for all models $\mathrm{LTL}(\mathbb{N}^*, \preceq_p)$ models σ, we have $\sigma, 0 \models \phi$ iff $\mathfrak{g}(\sigma), 0 \models \mathfrak{g}(\phi)$.

($\Rightarrow$) Let σ be an $\mathrm{LTL}(\Sigma^*, \preceq_p)$ model σ such that $\sigma, 0 \models \phi$. Let us build a bijection $\mathfrak{f} : \Sigma \to \mathbb{N}$ such that for all $i \in [1, k]$, we have $\mathfrak{f}(a_i) = i$ and $\mathfrak{f}(\sigma), 0 \models \mathfrak{f}(\phi)$. Let $b_{i_1}, \ldots, b_{i_{k'}}$ be all the letters in $(\Sigma \setminus \{a_1, \ldots, a_k\})$ such that $\mathfrak{f}^\star(b_{i_j}) \in \{1, \ldots, k\}$. So, there are k' letters $a_{j_1}, \ldots, a_{j_{k'}}$ whose image by $\mathfrak{f}^\star$ is not in $[1, k]$ since $\mathfrak{f}^\star$ is a bijection.

$$\begin{array}{lcll} \mathfrak{f}(a_i) & \stackrel{\text{def}}{=} & i & \text{for all } i \in [1, k] \\ \mathfrak{f}(b_{i_\alpha}) & \stackrel{\text{def}}{=} & \mathfrak{f}^\star(a_{j_\alpha}) & \text{for all } \alpha \in [1, k'] \\ \mathfrak{f}(a) & \stackrel{\text{def}}{=} & \mathfrak{f}^\star(a) & \text{otherwise} \end{array}$$

One can check that $\mathfrak{f}$ is a bijection, $\phi' = \mathfrak{f}(\phi)$ and therefore $\mathfrak{f}(\sigma), 0 \models \phi'$.

($\Leftarrow$) Let σ be a $\mathrm{LTL}(\mathbb{N}^*, \preceq_p)$ model σ such that $\sigma, 0 \models \phi'$. Similarly, one can build a bijection $\mathfrak{g} : \mathbb{N} \to \Sigma$ such that for all $i \in [1, k]$, we have $\mathfrak{g}(i) = a_i$ and $\mathfrak{g}(\sigma), 0 \models \phi$. Let $i_1, \ldots, i_{k'}$ be all the indices in $\mathbb{N} \setminus [1, k]$ such that $(\mathfrak{f}^\star)^{-1}(i_j) \in \{a_1, \ldots, a_k\}$. So, there are k' natural numbers $j_1, \ldots, j_{k'}$ in $[1, k]$ whose image by $(\mathfrak{f}^\star)^{-1}$ is not in $\{a_1, \ldots, a_k\}$.

$$\begin{array}{lcll} \mathfrak{g}(i) & \stackrel{\text{def}}{=} & a_i & \text{for all } i \in [1, k] \\ \mathfrak{g}(i_\alpha) & \stackrel{\text{def}}{=} & (\mathfrak{f}^\star)^{-1}(j_\alpha) & \text{for all } \alpha \in [1, k'] \\ \mathfrak{g}(n) & \stackrel{\text{def}}{=} & (\mathfrak{f}^\star)^{-1}(n) & \text{otherwise} \end{array}$$

One can check that $\mathfrak{g}$ is a bijection, $\phi = \mathfrak{g}(\phi')$ and therefore $\mathfrak{g}(\sigma), 0 \models \phi$. □

As a consequence, in the sequel, either we consider that Σ is finite and non-empty, or we consider that $\Sigma = \mathbb{N}$. Similarly, we have the following property.

Lemma 2. *Let Σ be a finite alphabet with cardinality $k \geq 2$. There is a logarithmic-space satisfiability-preserving reduction from $\mathrm{LTL}(\Sigma^*, \preceq_p)$ restricted to formulae without words in Σ^* to $\mathrm{LTL}(\{0,1\}^*, \preceq_p)$.*

Note that Lemma 2 does not cover the case $\Sigma = \mathbb{N}$ but at the end of our investigation (see Corollary 1), we will be able to deal with the infinite case too, by reduction to the case with a finite alphabet.

Proof. Let Σ be a finite alphabet of cardinality $k \geq 2$, say $\Sigma = [0, k-1]$, and ϕ be a formula in $\mathrm{LTL}(\Sigma^*, \preceq_p)$ without words in Σ^*. For every $n \in [0, k-1]$, we write $enc(n)$ to denote the binary writing of n using $\lceil log_2(k) \rceil$ bits and we extend enc homomorphically to any finite word in Σ^*. One can show that ϕ is satisfiable in $\mathrm{LTL}(\Sigma^*, \preceq_p)$ iff ϕ is satisfiable in $\mathrm{LTL}(\{0,1\}^*, \preceq_p)$. This is based on the property that for all $w, w' \in \Sigma^*$, we have $w \preceq_p w'$ iff $enc(w) \preceq_p enc(w')$. The other direction is by an easy verification since an $\mathrm{LTL}(\{0,1\}^*, \preceq_p)$ model is an $\mathrm{LTL}(\Sigma^*, \preceq_p)$ model too. □

In the paper, we shall characterize the computational complexity of the satisfiability problem for any logic $\mathrm{LTL}(\Sigma^*, \preceq_p)$. The problem is already PSPACE-hard by the PSPACE-hardness of the LTL satisfiability problem. Branching-time extension shall be considered also.

2.2 LTL on finite strings with length constraints

In order to characterize the computational complexity for $\mathrm{LTL}(\Sigma^*, \preceq_p)$, we make a slight detour by introducing an enriched version in which length constraints are allowed, instead of the prefix relation only.

Given $w \in \Sigma^*$, we write $\mathrm{len}(w)$ to denote its **length** in $\mathbb{N}$. When w is a word $a_1 \cdots a_n$ of length $n \geq 1$, for every $i \in [0, n-1]$, we write $w(i)$ to denote the letter a_{i+1} (so, as usual, the first index is zero). Similarly, given $w, w' \in \Sigma^*$, we write $\mathrm{clen}(w, w')$ to denote the length of the longest common prefix between w and w'. More precisely, there are w_0, w_1, and w_1' such that $w = w_0 \cdot w_1$, $w' = w_0 \cdot w_1'$ and, w_1 and w_1' cannot start by the same first letter, if any. We set $\mathrm{clen}(w, w') \stackrel{\text{def}}{=} \mathrm{len}(w_0)$. For instance $\mathrm{clen}(00100, 0011100) = 3$. So, $\mathrm{clen}(w, w) = \mathrm{len}(w)$, and w is a prefix of w' iff $\mathrm{clen}(w, w) = \mathrm{clen}(w, w')$.

The logic $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$ is defined as $\mathrm{LTL}(\Sigma^*, \preceq_p)$ except that atomic formulae of the form $\mathtt{t} \preceq_p \mathtt{t}'$ are replaced by atomic formulae of the form

$$\mathrm{clen}(\mathtt{t}_0, \mathtt{t}_0') \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}_1'),$$

where $\mathtt{t}_0$, $\mathtt{t}_0'$, $\mathtt{t}_1$, $\mathtt{t}_1'$ are terms, defined exactly as in $\mathrm{LTL}(\Sigma^*, \preceq_p)$. The argument of $\mathrm{clen}(\cdot)$ should be viewed as a set and therefore the expressions $\mathrm{clen}(\mathtt{t}, \mathtt{t}')$ and $\mathrm{clen}(\mathtt{t}', \mathtt{t})$ are considered as identical. Expressions of the form $\mathrm{clen}(\mathtt{t}_0, \mathtt{t}_0')$ are called **counters** since they are interpreted by natural numbers. Sometimes, we also write $\mathrm{len}(\mathtt{t})$ instead of $\mathrm{clen}(\mathtt{t}, \mathtt{t})$. The satisfaction relation is then adapted as follows (only the clause for atomic formulae needs to be specified since the other clauses are identical):

$$\sigma, i \models \mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1) \stackrel{\text{def}}{\Leftrightarrow} \mathrm{clen}([\![\mathtt{t}_0]\!]_i, [\![\mathtt{t}'_0]\!]_i) \leq \mathrm{clen}([\![\mathtt{t}_1]\!]_i, [\![\mathtt{t}'_1]\!]_i).$$

So, the logic $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$ allows us to express richer constraints than those from $\mathrm{LTL}(\Sigma^*, \preceq_p)$. Roughly speaking, while atomic formulae in $\mathrm{LTL}(\Sigma^*, \preceq_p)$ can only speak about the prefix relation, atomic formulae in $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$ are more precise and can state relationships between the lengths of longest common prefixes.

Lemma 3. *Let Σ be either a finite non-empty alphabet or $\mathbb{N}$. There is a logarithmic-space reduction from the satisfiability problem for $\mathrm{LTL}(\Sigma^*, \preceq_p)$ to the satisfiability problem for $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$.*

It is clear that the reduction is homomorphic for any Boolean or temporal connective, whereas $\mathtt{t} \preceq_p \mathtt{t}'$ is translated into $\mathrm{clen}(\mathtt{t}, \mathtt{t}) \leq \mathrm{clen}(\mathtt{t}, \mathtt{t}')$. Not only is the reduction satisfiability-preserving (which is sufficient to conclude upper bounds for $\mathrm{LTL}(\Sigma^*, \preceq_p)$ from upper bounds from $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$ but it preserves also the semantics, which is after all not so surprising at this point.

The definition of $\mathrm{clen}(\cdot)$ can be easily generalized to any (finite) number of strings. Given $w_1, \ldots, w_n \in \Sigma^*$, we write $\mathrm{clen}(w_1, \ldots, w_n)$ to denote the length of the longest common prefix for all w_i's. The logic $\mathrm{LTL}'(\Sigma^*, \mathrm{clen})$ is defined as $\mathrm{LTL}(\Sigma^*, \preceq_p)$ except that atomic formulae of the form $\mathtt{t} \preceq_p \mathtt{t}'$ are replaced by atomic formulae of the form

$$\mathrm{clen}(\mathtt{t}_1, \ldots, \mathtt{t}_n) \leq \mathrm{clen}(\mathtt{t}'_1, \ldots, \mathtt{t}'_{n'}),$$

where the terms are defined exactly as in $\mathrm{LTL}(\Sigma^*, \preceq_p)$ and $n, n' \geq 1$. The satisfaction relation is then adapted as follows:

$$\sigma, i \models \mathrm{clen}(\mathtt{t}_1, \ldots, \mathtt{t}_n) \leq \mathrm{clen}(\mathtt{t}'_1, \ldots, \mathtt{t}'_{n'}) \stackrel{\text{def}}{\Leftrightarrow} \mathrm{clen}([\![\mathtt{t}_1]\!]_i, \ldots, [\![\mathtt{t}_n]\!]_i) \leq \mathrm{clen}([\![\mathtt{t}'_1]\!]_i, \ldots, [\![\mathtt{t}'_{n'}]\!]_i).$$

So, apparently, the logic $\mathrm{LTL}'(\Sigma^*, \mathrm{clen})$ allows us to express even richer constraints than those from $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$. Nevertheless, by noting that

$$\mathrm{clen}(w_1, \ldots, w_n) = min\{\mathrm{clen}(w_i, w_j) : i \neq j \in [1, n]\},$$

one can show that there is a logarithmic-space reduction from the satisfiability problem for $\mathrm{LTL}'(\Sigma^*, \mathrm{clen})$ and the satisfiability problem for $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$.

That is why, in the rest of the paper, we mainly focus our investigations on linear-time temporal logics of the form $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$ where Σ is either a finite alphabet or $\mathbb{N}$. In the finite case, we cannot restrict ourselves to a binary alphabet since the proof for Lemma 2 cannot be used to show that $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$ with finite alphabet Σ can be reduced to $\mathrm{LTL}(\{0,1\}^*, \mathrm{clen})$ because the encoding used in the proof of Lemma 2 does not preserve the length of longest common prefixes.

Here are three simple properties about $\mathrm{clen}(\cdot)$ that play a special role in the sequel, namely this is all that one needs to know about $\mathrm{clen}(\cdot)$ (see Lemma 5 and Lemma 6 to justify that claim).

Proposition 2. *Let Σ be a finite alphabet of cardinality $k \geq 2$.*

(I) *For all* $w, w' \in \Sigma^*$, $\mathrm{len}(w) \geq \mathrm{clen}(w, w')$.

(II) *For all* $w_0, w_1, \ldots, w_k \in \Sigma^*$ *such that* $\mathrm{clen}(w_0, w_1) = \cdots = \mathrm{clen}(w_0, w_k)$ *and for all* $i \in [0, k]$, $\mathrm{clen}(w_0, w_1) < \mathrm{len}(w_i)$, *we have that there are* $i \neq j \in [1, k]$ *such that* $\mathrm{clen}(w_0, w_1) < \mathrm{clen}(w_i, w_j)$.

(III) *For all* $w_0, w_1, w_2 \in \Sigma^*$, *if* $\mathrm{clen}(w_0, w_1) < \mathrm{clen}(w_1, w_2)$, *then* $\mathrm{clen}(w_0, w_1) = \mathrm{clen}(w_0, w_2)$.

Similarly, if $\Sigma = \mathbb{N}$, then the properties Proposition 2(I) and Proposition 2(III) hold true. In Figure 1, we present the strings w_0, ..., w_k so that assumptions of Proposition 2(II) are satisfied. In particular, $\mathrm{clen}(w_0, w_1) = \cdots = \mathrm{clen}(w_0, w_k) = 3$ and for all $i \in [1, k]$, $w_i(3) \neq w_0(3)$ (here $w_0(3) = 1$). Since for every $i \in [1, k]$, the word w_i has at least four letters, by the Pigeonhole Principle, there are $i \neq j \in [1, k]$ such that $w_i(3) = w_j(3)$.

$w_0 :$ **0 0 0** 1 0 2
$w_1 :$ **0 0 0** 0 1 3 5 6
$w_2 :$ **0 0 0** 2 1 4
...
$w_k :$ **0 0 0** 3 1 3

Fig. 1. Strings satisfying assumptions of Proposition 2(II).

Proof. (I) Since the longest common prefix between w and w' is clearly a prefix of w, its length is necessarily bounded by the length of w, which is precisely the statement for (I).
(II) The proof is essentially based on the famous Pigeonhole Principle. Let w_0, w_1, ..., w_k be words in Σ^* satisfying the assumptions in statement (II). So, for every $i \in [0, k]$, w_i can be written in the form $w \cdot w'_i$ such that w is the longest common prefix between the words w_0, w_1, ..., w_k and $w'_i \in \Sigma^+$. By the Pigeonhole Principle, since $\mathrm{card}(\Sigma \setminus w'_0(0)) = k - 1$, there are $j \neq j' \in [1, k]$ such that $w'_j(0) = w'_{j'}(0) \in (\Sigma \setminus w'_0(0))$. Consequently, $\mathrm{clen}(w_0, w_1) < \mathrm{clen}(w_j, w_{j'})$.
(III) Let w_0, w_1 and w_2 be words such that $\mathrm{clen}(w_0, w_1) < \mathrm{clen}(w_1, w_2)$. So, the words w_0, w_1 and w_2 can be decomposed in the following way

$$w_0 = w_0^1 \cdot w_0^2 \qquad w_1 = w_0^1 \cdot w_1^1 \cdot w_1^2 \qquad w_2 = w_0^1 \cdot w_1^1 \cdot w_2^1$$

for some $w_1^1 \in \Sigma^+$ and, w_0^2 and w_1^1 do not start by the same letter if any. It is immediate that $\mathrm{clen}(w_0, w_1) = \mathrm{clen}(w_0, w_2) = \mathrm{len}(w_0^1)$. □

As a corollary, the formulae below are valid in $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$ with $\mathrm{card}(\Sigma) = k$:

$$\phi_{\mathrm{I}} \stackrel{\mathrm{def}}{=} \mathrm{len}(\mathsf{x}) \geq \mathrm{clen}(\mathsf{x}, \mathsf{x}')$$

$$\phi_{\mathrm{II}}^k \stackrel{\mathrm{def}}{=} ((\bigwedge_{i \in [0,k]} (\mathrm{clen}(\mathsf{x}_0, \mathsf{x}_1) < \mathrm{len}(\mathsf{x}_i))) \wedge \mathrm{clen}(\mathsf{x}_0, \mathsf{x}_1) = \cdots = \mathrm{clen}(\mathsf{x}_0, \mathsf{x}_k)) \Rightarrow$$

$$\Big(\bigvee_{i\neq j\in[1,k]} (\mathrm{clen}(\mathtt{x}_0,\mathtt{x}_1) < \mathrm{clen}(\mathtt{x}_i,\mathtt{x}_j))\Big)$$

$$\phi_{\mathrm{III}} \stackrel{\mathrm{def}}{=} (\mathrm{clen}(\mathtt{x},\mathtt{x}') < \mathrm{clen}(\mathtt{x}',\mathtt{x}'')) \Rightarrow (\mathrm{clen}(\mathtt{x},\mathtt{x}') = \mathrm{clen}(\mathtt{x},\mathtt{x}''))$$

Note that $\phi_{\mathrm{II}}^{k'}$ is valid in $\mathrm{LTL}(\Sigma^*,\mathrm{clen})$ if $\mathrm{card}(\Sigma) \leq k'$.

Below, we introduce another syntactic restriction that is useful in the sequel without sacrificing expressive power.

Lemma 4. *Let Σ be a non-empty countable alphabet. There is a logarithmic-space reduction from the* $\mathrm{LTL}(\Sigma^*,\mathrm{clen})$ *satisfiability problem into its restriction in which terms are restricted to $w \in \Sigma^*$, $\mathtt{x}$, and $\mathtt{X}\mathtt{x}$, with $\mathtt{x} \in \mathrm{SVAR}$.*

The proof is standard, and it is based on the renaming technique.

Proof. Given terms $\mathtt{t}$ and $\mathtt{t}'$, we write $\mathtt{t} = \mathtt{t}'$ to denote the formula below:

$$(\mathtt{t} = \mathtt{t}') \stackrel{\mathrm{def}}{=} (\mathrm{clen}(\mathtt{t},\mathtt{t}) \leq \mathrm{clen}(\mathtt{t},\mathtt{t}')) \wedge (\mathrm{clen}(\mathtt{t}',\mathtt{t}') \leq \mathrm{clen}(\mathtt{t},\mathtt{t}')).$$

Obviously, $(\mathtt{t} = \mathtt{t}')$ holds true whenever $\mathtt{t}$ and $\mathtt{t}'$ are interpreted by the same string. Now, let ϕ be a formula in $\mathrm{LTL}(\Sigma^*,\mathrm{clen})$ with occurrences of the term $\mathtt{X}^j\mathtt{x}$ for some $j \geq 2$. Let $\mathtt{x}_1^{new}, \ldots, \mathtt{x}_j^{new}$ be new variables. One can show that ϕ is satisfiable iff the formula below is satisfiable:

$$\mathtt{G}(\mathtt{x}_1^{new} = \mathtt{X}\mathtt{x}) \wedge \bigwedge_{\alpha=2}^{j} \mathtt{G}(\mathtt{x}_\alpha^{new} = \mathtt{X}\mathtt{x}_{\alpha-1}^{new}) \wedge \phi[\mathtt{X}^j\mathtt{x} \leftarrow \mathtt{x}_j^{new}],$$

where $\phi[\mathtt{X}^j\mathtt{x} \leftarrow \mathtt{x}_j^{new}]$ is the formula obtained from ϕ by replacing every occurrence of $\mathtt{X}^j\mathtt{x}$ by $\mathtt{x}_j^{new}$. Note that the first two conjuncts do not introduce terms of the form $\mathtt{X}^{j'}\mathtt{x}'$ with $j' > 1$. Therefore by applying such a transformation a number of times bounded by the size of ϕ, we obtain a logarithmic-space reduction from the $\mathrm{LTL}(\Sigma^*,\mathrm{clen})$ satisfiability problem to its adequate restriction. We use the facts that logarithmic-space reductions are closed under composition and enumerating a polynomial amount of conjuncts, as done above, requires only logarithmic-space thanks to a binary encoding for indices (the size of the outcome formula is not part of the used space, as usual in complexity theory). □

3 String-compatible counter valuations

This section contains the key result to transform string constraints with the prefix order into counter constraints (Lemma 6). Before presenting such a result, first we explain why solving word equations (see e.g. [Mak77]) allows to solve Boolean combinations of prefix constraints in polynomial space thanks to the PSPACE upper bound established in [Pla04]. Actually, at the end of the section, we show that the problem can be solved in NP by taking advantage of Lemma 6 and the NP upper bound for solving quantifier-free Presburger formulae.

3.1 A remark on word equations

Given a finite alphabet Σ, a **word equation** $\mathcal{E}$ is an expression of the form $w = w'$ where $w, w' \in (\Sigma \uplus \text{SVAR})^*$. The equation $\mathcal{E}$ is **satisfiable** iff there is a valuation $\mathfrak{s} : (\Sigma \uplus \text{SVAR}) \to \Sigma^*$ extended homomorphically to $(\Sigma \uplus \text{SVAR})^*$ and equal to the identity on Σ^* such that $\mathfrak{s}(w) = \mathfrak{s}(w')$. The satisfiability problem for word equations can be solved in PSPACE [Pla04] (see also Makanin's contribution for decidability in [Mak77]). Thanks to this result, deciding the Boolean combination of prefix constraints of the form $\mathtt{t} \preceq_p \mathtt{t}'$ (the terms are either words in Σ^* or string variables) can be decided in PSPACE too, as briefly explained below. However, prefix constraints are simple enough so that it is possible to design an ad-hoc but much simpler decision procedure, as shown below. This happens to be very useful in Section 4.

Before introducing string-compatible counter valuations (thanks to constraints on the lengths of the longest common prefixes that lead to genuine string valuations), let us briefly explain first how to solve Boolean combinations of prefix constraints by solving word equations. This is indeed based on the following observations.

1. Simultaneous satisfiability for a finite set $\{w_1 = w'_1, \ldots, w_n = w'_n\}$ of word equations can be done by checking the satisfiability of the single word equation below

$$w_1 \sharp_1 w_2 \sharp_2 \cdots \sharp_{n-1} w_n \;=\; w'_1 \sharp_1 w'_2 \sharp_2 \cdots \sharp_{n-1} w'_n,$$

   where $\sharp_1, \ldots, \sharp_{n-1}$ are new letters (separators) dedicated to synchronization.
2. $\mathtt{t} \preceq_p \mathtt{t}'$ can be reduced to the word equation $\mathtt{t} \cdot \mathtt{x}_{new} = \mathtt{t}'$ where $\mathtt{x}_{new}$ is a new string variable.
3. $\neg(\mathtt{t} \preceq_p \mathtt{t}')$ can be reduced to the positive Boolean combination of word equations below:

$$\Big(\bigvee_{a \neq b \in \Sigma} (\mathtt{t} \;=\; \mathtt{x}^1_{new}\; a\; \mathtt{x}^2_{new} \;\wedge\; \mathtt{t}' \;=\; \mathtt{x}^1_{new}\; b\; \mathtt{x}^3_{new})\Big) \;\vee\; \Big(\bigvee_{a \in \Sigma} (\mathtt{t} \;=\; \mathtt{t}'\; a\; \mathtt{x}^4_{new})\Big).$$

So, given a Boolean combination ϕ of prefix constraints, the reduction works as follows. Without any loss of generality, the formula ϕ is in negation normal form. First, replace every constraint $\mathtt{t} \preceq_p \mathtt{t}'$ or $\neg(\mathtt{t} \preceq_p \mathtt{t}')$ by its translation as a positive Boolean combination of word equations; call this result ϕ'. Then, guess a finite set of word equations that makes ϕ' true propositionally and, finally, apply the PSPACE algorithm to the set of word equations using the reduction in (1) above. A finite set of word equations makes true a propositional formula in NNF that is built from word equations when replacing every word equation from the set by $\top$ (truth constant) makes the formula true whatever the truth value of the other word equations. By way of example, $\{\mathtt{x} = a, \mathtt{z} = c\}$ makes true the formula $((\mathtt{x} = a) \vee (\mathtt{y} = b)) \wedge (\mathtt{x} = a) \wedge (\mathtt{z} = c)$. This leads to an NPSPACE algorithm, which can be solved in PSPACE thanks to Savitch's Theorem [Sav70]. At this point, it is worth noting that the above reduction does not work when $\Sigma = \mathbb{N}$ but adding suffix constraints to the prefix constraints can be dealt with similarly. As a by-product of the developments below, solving Boolean combinations of prefix constraints (whatever the cardinality of the fixed alphabet) can be done in NP.

3.2 The characterisation

Let Σ be either a finite alphabet of the form $[0, k-1]$ for some $k \geq 2$ or $\mathbb{N}$. Let SVAR' be a finite subset of SVAR. As usual, a string valuation $\mathfrak{s}$ with respect to SVAR' is a map $\mathfrak{s} : \mathrm{SVAR}' \to \Sigma^*$. A **counter valuation** $\mathfrak{c}$ with respect to SVAR' is defined as a map

$$\mathfrak{c} : \{\mathrm{clen}(\mathsf{x}, \mathsf{x}') : \mathsf{x}, \mathsf{x}' \in \mathrm{SVAR}'\} \to \mathbb{N}.$$

When a counter valuation $\mathfrak{c}$ satisfies the conjunction of the formulae below, we say that $\mathfrak{c}$ is **string-compatible** (with respect to SVAR'):

- Formula ψ_{I} related to condition (I) from Proposition 2:

$$\bigwedge_{\mathsf{x},\mathsf{x}' \in \mathrm{SVAR}'} (\mathrm{clen}(\mathsf{x}, \mathsf{x}) \geq \mathrm{clen}(\mathsf{x}, \mathsf{x}')).$$

- Formula ψ_{II} related to condition (II) from Proposition 2: either $\top$ if $\Sigma = \mathbb{N}$ or the formula below if $\Sigma = [0, k-1]$

$$\bigwedge_{\mathsf{x}_0,\ldots,\mathsf{x}_k \in \mathrm{SVAR}'} ((\bigwedge_{i \in [0,k]} (\mathrm{clen}(\mathsf{x}_0, \mathsf{x}_1) < \mathrm{clen}(\mathsf{x}_i, \mathsf{x}_i))) \ \wedge \ \mathrm{clen}(\mathsf{x}_0, \mathsf{x}_1) = \cdots = \mathrm{clen}(\mathsf{x}_0, \mathsf{x}_k)) \ \Rightarrow$$

$$(\bigvee_{i \neq j \in [1,k]} (\mathrm{clen}(\mathsf{x}_0, \mathsf{x}_1) < \mathrm{clen}(\mathsf{x}_i, \mathsf{x}_j))).$$

- Formula ψ_{III} related to condition (III) from Proposition 2:

$$\bigwedge_{\mathsf{x},\mathsf{x}',\mathsf{x}'' \in \mathrm{SVAR}'} (\mathrm{clen}(\mathsf{x}, \mathsf{x}') < \mathrm{clen}(\mathsf{x}', \mathsf{x}'')) \Rightarrow (\mathrm{clen}(\mathsf{x}, \mathsf{x}') = \mathrm{clen}(\mathsf{x}, \mathsf{x}'')).$$

The size of the above conjunction is in $\mathcal{O}(\mathrm{card}(\mathrm{SVAR}')^{\mathrm{k}+2})$, i.e. polynomial in $\mathrm{card}(\mathrm{SVAR}')$ when the alphabet is fixed and finite. If $\Sigma = \mathbb{N}$, then the size of the above conjunction is in $\mathcal{O}(\mathrm{card}(\mathrm{SVAR}')^3)$. When $X \subseteq \mathrm{SVAR}'$ and $\mathfrak{c}$ is string-compatible with respect to SVAR', the restriction of $\mathfrak{c}$ to X is also string-compatible with respect to X.

Let X be a non-empty subset of SVAR', $\mathfrak{s}$ be a string valuation and $\mathfrak{c}$ be a counter valuation, both with respect to SVAR'. We say that $\mathfrak{c}$ agrees with $\mathfrak{s}$ on X (written $\mathfrak{c} \approx_X \mathfrak{s}$) $\overset{\text{def}}{\Leftrightarrow}$ $\mathfrak{c}(\mathrm{clen}(\mathsf{x}, \mathsf{x}')) = \mathrm{clen}(\mathfrak{s}(\mathsf{x}), \mathfrak{s}(\mathsf{x}'))$ for all $\mathsf{x}, \mathsf{x}' \in X$. So, if $X' \subseteq X$ and $\mathfrak{c} \approx_X \mathfrak{s}$, then $\mathfrak{c} \approx_{X'} \mathfrak{s}$ too. Moreover, note that the interpretation of $\mathfrak{s}$ on $\mathrm{SVAR}' \setminus \mathrm{X}$ is useless for the binary relation $\approx_X$ and therefore the restriction of $\mathfrak{s}$ to $\mathrm{SVAR}' \setminus \mathrm{X}$ may be sometime omitted. Given a string valuation $\mathfrak{s}$, there is a counter valuation $\mathfrak{c}$ such that $\mathfrak{c} \approx_X \mathfrak{s}$ and $\mathfrak{c}$ can be defined obviously by: $\mathfrak{c}(\mathrm{clen}(\mathsf{x}, \mathsf{x}')) \overset{\text{def}}{=} \mathrm{clen}(\mathfrak{s}(\mathsf{x}), \mathfrak{s}(\mathsf{x}'))$ for all $\mathsf{x}, \mathsf{x}' \in X$.

Lemma 5. *Let $\mathfrak{s}$ be a string valuation and $\mathfrak{c}$ be the counter valuation that agree on* SVAR'. *Then, $\mathfrak{c}$ is string-compatible with respect* SVAR'.

This is an immediate consequence of Proposition 2. Whereas Lemma 5 corresponds to a correctness lemma (string valuations induce string-compatible counter valuations), Lemma 6 below states a completeness result: every string-compatible counter valuation can be indeed obtained from a string valuation. The statement in Lemma 6 is just a bit more complex than one might think at first because of the distinction between two sets of string variables, but this is due to its forthcoming use in relationship with the logics LTL(Σ^*, clen) and CTL*(Σ^*, clen).

Lemma 6. *Let $X \neq \emptyset$ and Y be finite and disjoint sets of string variables, $\mathfrak{c}$ be a string-compatible counter valuation with respect to $X \uplus Y$ and $\mathfrak{s} : Y \to \Sigma^*$ be such that $\mathfrak{c} \approx_Y \mathfrak{s}$. Then, there is a string valuation $\mathfrak{s}'$ that is a conservative extension of $\mathfrak{s}$, such that $\mathfrak{c} \approx_{X \uplus Y} \mathfrak{s}'$.*

Assuming that $Y = \{\mathtt{y}_1, \mathtt{y}_2\}$, the string valuations $\mathfrak{s}$ and $\mathfrak{s}'$ below are therefore equivalent as far as the existence of a conservative extension is concerned since the lengths of longest common prefixes are identical.

$$\begin{array}{ll} \mathfrak{s}(\mathtt{y}_1) = 0101 & \mathfrak{s}'(\mathtt{y}_1) = 3456 \\ \mathfrak{s}(\mathtt{y}_2) = 01112 & \mathfrak{s}'(\mathtt{y}_2) = 34777 \end{array}$$

Proof. Below, we prove the result when $\mathrm{card}(X) = 1$ (unary case) since this is sufficient to conclude the proof of the lemma when $\mathrm{card}(X) > 1$ by applying the construction below exactly $\mathrm{card}(X)$ times. Indeed, let $X = \{\mathtt{x}_1, \ldots, \mathtt{x}_r\}$ with $r \geq 1$. Here is how the general case can be solved from the unary case.

— Compute $\mathfrak{s}_1 : Y \uplus \{\mathtt{x}_1\} \to \Sigma^*$ such that $\mathfrak{c} \approx_{Y \uplus \{\mathtt{x}_1\}} \mathfrak{s}_1$ by application of the lemma in the unary case. ($\mathfrak{c}$ is also string-compatible with respect to $Y \uplus \{\mathtt{x}_1\}$)
— Let $Y := Y \uplus \{\mathtt{x}_1\}$. Compute $\mathfrak{s}_2 : Y \uplus \{\mathtt{x}_2\} \to \Sigma^*$ extending $\mathfrak{s}_1$ such that $\mathfrak{c} \approx_{Y \uplus \{\mathtt{x}_2\}} \mathfrak{s}_2$ by application of the lemma in the unary case. ($\mathfrak{c}$ is also string-compatible with respect to $Y \uplus \{\mathtt{x}_1, \mathtt{x}_2\}$)
...
— Let $Y := Y \uplus \{\mathtt{x}_{r-1}\}$. Compute $\mathfrak{s}_r : Y \uplus \{\mathtt{x}_r\} \to \Sigma^*$ extending $\mathfrak{s}_{r-1}$ such that $\mathfrak{c} \approx_{Y \uplus \{\mathtt{x}_r\}} \mathfrak{s}_r$ by application of the lemma in the unary case.

Let us therefore consider the unary case below with $X = \{\mathtt{x}\}$. The construction of $\mathfrak{s}'(\mathtt{x})$ is done in at most four steps.

(Step 1) Let $\mathtt{t}^1, \ldots, \mathtt{t}^\alpha$ be all the distinct variables in Y such that

$$\mathfrak{c}(\mathrm{clen}(\mathtt{t}^1, \mathtt{x})) = \cdots = \mathfrak{c}(\mathrm{clen}(\mathtt{t}^\alpha, \mathtt{x})) \;=\; max\{\mathfrak{c}(\mathrm{clen}(\mathtt{t}, \mathtt{x})) : \mathtt{t} \in Y\} \stackrel{\text{def}}{=} \mathtt{M}.$$

In a sense, the first $\mathfrak{c}(\mathrm{clen}(\mathtt{t}^1, \mathtt{x}))$ values of the string $\mathfrak{s}'(\mathtt{x})$ are uniquely inherited from $\mathfrak{s}$. The only situation for which $\mathtt{t}^1, \ldots, \mathtt{t}^\alpha$ is void is when Y is empty. In that case, we omit the steps (2), (3) and (4) below and for every $j \in [0, \mathfrak{c}(\mathrm{clen}(\mathtt{x}, \mathtt{x})) - 1]$, $\mathfrak{s}'(\mathtt{x})(j) \stackrel{\text{def}}{=} 0$.

(Step 2) For every $j \in [0, \mathtt{M} - 1]$, $\mathfrak{s}'(\mathtt{x})(j) \stackrel{\text{def}}{=} \mathfrak{s}(\mathtt{t}^1)(j)$. If $\mathfrak{c}(\mathrm{clen}(\mathtt{x}, \mathtt{x})) = \mathtt{M}$, then we are done, otherwise go to Step (3).

(Step 3) Let β be $min(\mathbb{N} \setminus \{\mathfrak{s}(\mathtt{t}^j)(\mathtt{M}) : j \in [1, \alpha], \mathfrak{c}(\mathrm{clen}(\mathtt{t}^j, \mathtt{t}^j)) > \mathtt{M}\})$. We pose $\mathfrak{s}'(\mathtt{x})(\mathtt{M}) \stackrel{\text{def}}{=} \beta$.

(Step 4) For every $j \in [\mathtt{M}+1, \mathfrak{c}(\mathrm{clen}(\mathsf{x},\mathsf{x}))-1]$, $\mathfrak{s}'(\mathsf{x})(j) \stackrel{\text{def}}{=} 0$.

Above, it is remarkable that $\mathfrak{s}'$ can be defined deterministically once $\mathfrak{c}$ and $\mathfrak{s}$ are provided. An alternative definition of $\mathfrak{s}'$ that allows nondeterminism consists of allowing in Step (4) any value in Σ to define $\mathfrak{s}'(\mathsf{x})(j)$. The same reasoning applies to the degenerate case in Step (1). Furthermore, nondeterminism in Step (3) would also be possible (and the construction would satisfy the same essential properties) by allowing any value in

$$(\Sigma \setminus \{\mathfrak{s}(\mathtt{t}^j)(\mathtt{M}) : j \in [1,\alpha], \mathfrak{c}(\mathrm{clen}(\mathtt{t}^j,\mathtt{t}^j)) > \mathtt{M}\})$$

for defining $\mathfrak{s}'(\mathsf{x})(\mathtt{M})$. By contrast, the situation would be quite different if we had constraints on suffixes or regularity constraints too. Note also that in (Step 2), $\beta \leq \mathrm{card}(Y)$ since there are at most $\mathrm{card}(Y)$ values that are removed from $\mathbb{N}$.

It remains to show that $\mathfrak{s}'$ satisfies the required properties, which amounts to checking the following three properties.

(a) $\mathrm{len}(\mathfrak{s}'(\mathsf{x})) = \mathfrak{c}(\mathrm{clen}(\mathsf{x},\mathsf{x}))$.
(b) $\mathfrak{s}'(\mathsf{x}) \in \Sigma^*$.
(c) For all $\mathsf{x}', \mathsf{x}'' \in Y \cup \{\mathsf{x}\}$, $\mathrm{clen}(\mathfrak{s}'(\mathsf{x}'), \mathfrak{s}'(\mathsf{x}'')) = \mathfrak{c}(\mathrm{clen}(\mathsf{x}',\mathsf{x}''))$.

(a) The only reason for assigning a letter to the string $\mathfrak{s}'(\mathsf{x})$ strictly after the index $\mathfrak{c}(\mathrm{clen}(\mathsf{x},\mathsf{x}))-1$ would be if $\mathtt{M} > \mathfrak{c}(\mathrm{clen}(\mathsf{x},\mathsf{x}))$. This cannot occur since $\mathfrak{c}$ is string-compatible and more specifically it satisfies the formula

$$\bigwedge_{\mathsf{z},\mathsf{z}' \in Y \cup \{\mathsf{x}\}} (\mathrm{clen}(\mathsf{z},\mathsf{z}) \geq \mathrm{clen}(\mathsf{z},\mathsf{z}'))$$

from condition (I) in Proposition 2. Moreover, the abort condition in (Step 2) rules out the case $\mathtt{M} = \mathfrak{c}(\mathrm{clen}(\mathsf{x},\mathsf{x}))$.

(b) The proof is *ad absurdum*. Let $J \in [0, \mathfrak{c}(\mathrm{clen}(\mathsf{x},\mathsf{x}))-1]$, $\mathfrak{s}'(\mathsf{x})(J) \notin \Sigma$. When $\Sigma = \mathbb{N}$ this leads to an immediate contradiction, so let us assume that $\Sigma = [0, k-1]$ for some $k \geq 2$. Since $\mathfrak{s}$ has the profile $Y \to \Sigma^*$, $\mathfrak{s}'(\mathsf{x})(J)$ cannot be defined in (Step 1). Similarly, $\mathfrak{s}'(\mathsf{x})(J)$ cannot be defined in (Step 4) since (always) $0 \in \Sigma$. So, $\mathfrak{s}'(\mathsf{x})(J)$ can only be defined in (Step 3), $\mathtt{M} < \mathfrak{c}(\mathrm{clen}(\mathsf{x},\mathsf{x}))$ and $J = \mathtt{M}$. Since $\mathfrak{s}'(\mathsf{x})(J) \notin \Sigma$, this means that $\mathfrak{s}'(\mathsf{x})(J) \geq k$ and therefore

$$[0, k-1] \subseteq \left\{\mathfrak{s}(\mathtt{t}^j)(\mathtt{M}) \ : \ j \in [1,\alpha], \mathfrak{c}(\mathrm{clen}(\mathtt{t}^j,\mathtt{t}^j)) > \mathtt{M}\right\}.$$

So, there are terms $\mathtt{t}'_0, \ldots, \mathtt{t}'_{k-1}$ in $\{\mathtt{t}^1, \ldots, \mathtt{t}^\alpha\}$ such that for every $j \in [0, k-1]$, $\mathfrak{s}(\mathtt{t}'_j)(\mathtt{M}) = j$. The strings $\mathfrak{s}'(\mathsf{x})$, $\mathfrak{s}(\mathtt{t}'_0)$, $\ldots$, $\mathfrak{s}(\mathtt{t}'_{k-1})$ satisfy the assumption in Proposition 2(II). Since $\mathfrak{c}$ is string-compatible, it satisfies the formula

$$\bigwedge_{\mathsf{z}_0,\ldots,\mathsf{z}_k \in Y \cup \{\mathsf{x}\}} ((\bigwedge_{i \in [0,k]} (\mathrm{clen}(\mathsf{z}_0,\mathsf{z}_1) < \mathrm{clen}(\mathsf{z}_i,\mathsf{z}_i))) \ \wedge \ \mathrm{clen}(\mathsf{z}_0,\mathsf{z}_1) = \cdots = \mathrm{clen}(\mathsf{z}_0,\mathsf{z}_k)) \ \Rightarrow$$

$$(\bigvee_{j \neq j' \in [1,k]} (\mathrm{clen}(\mathsf{z}_0,\mathsf{z}_1) < \mathrm{clen}(\mathsf{z}_j,\mathsf{z}_{j'})))$$

So, there are $j \neq j' \in [1,k]$ such that $\mathfrak{s}(\mathtt{t}'_j)(\mathtt{M}) = \mathfrak{s}(\mathtt{t}'_{j'})(\mathtt{M})$, which leads to a contradiction (since $j \neq j'$ implies that $\mathfrak{s}(\mathtt{t}'_j)(\mathtt{M})$ is different from $\mathfrak{s}(\mathtt{t}'_{j'})(\mathtt{M})$).

(c) The proof is *ad absurdum*. Let $\mathtt{t}$ in $Y \uplus \{\mathtt{x}\}$ with $\mathrm{clen}(\mathfrak{s}'(\mathtt{x}), \mathfrak{s}'(\mathtt{t})) \neq \mathfrak{c}(\mathrm{clen}(\mathtt{x}, \mathtt{t}))$. Note that by assumptions in the lemma, such a mismatch cannot be due to two terms from Y. By Step (2), we have $\mathrm{clen}(\mathfrak{s}'(\mathtt{x}), \mathfrak{s}(\mathtt{t}^1)) \geq \mathtt{M}$. Now suppose that $\mathrm{clen}(\mathfrak{s}'(\mathtt{x}), \mathfrak{s}(\mathtt{t}^j)) < \mathtt{M}$ for some $j \in [2, \alpha]$. By Proposition 2(III), we get that $\mathrm{clen}(\mathfrak{s}'(\mathtt{x}), \mathfrak{s}(\mathtt{t}^j)) = \mathrm{clen}(\mathfrak{s}(\mathtt{t}^1), \mathfrak{s}(\mathtt{t}^j))$. Furthermore, we have $\mathrm{clen}(\mathfrak{s}(\mathtt{t}^1), \mathfrak{s}(\mathtt{t}^j)) = \mathfrak{c}(\mathrm{clen}(\mathtt{t}^1, \mathtt{t}^j))$ since $\mathfrak{c} \approx_Y \mathfrak{s}$. So, $\mathfrak{c}(\mathrm{clen}(\mathtt{t}^1, \mathtt{t}^j)) < \mathtt{M}$ and therefore $\mathfrak{c}(\mathrm{clen}(\mathtt{t}^1, \mathtt{t}^j)) < \mathfrak{c}(\mathrm{clen}(\mathtt{x}, \mathtt{t}^1))$. Since $\mathfrak{c}$ is string-compatible, this implies that $\mathfrak{c}(\mathrm{clen}(\mathtt{t}^1, \mathtt{t}^j)) = \mathfrak{c}(\mathrm{clen}(\mathtt{x}, \mathtt{t}^j))$, which leads to a contradiction since $\mathfrak{c}(\mathrm{clen}(\mathtt{t}^1, \mathtt{t}^j)) < \mathtt{M}$ and $\mathfrak{c}(\mathrm{clen}(\mathtt{x}, \mathtt{t}^j)) = \mathtt{M}$. So, by definition of $\mathtt{t}^1, \ldots, \mathtt{t}^\alpha$ and by Step (2), for every $j \in [1, \alpha]$, we have $\mathrm{clen}(\mathfrak{s}'(\mathtt{x}), \mathfrak{s}(\mathtt{t}^j)) \geq \mathtt{M}$.

If $\mathfrak{c}(\mathrm{clen}(\mathtt{x}, \mathtt{x})) = \mathtt{M}$, then we are done with the definition of $\mathfrak{s}'(\mathtt{x})$ and therefore we get $\mathrm{clen}(\mathfrak{s}'(\mathtt{x}), \mathfrak{s}(\mathtt{t}')) = \mathfrak{c}(\mathrm{clen}(\mathtt{x}, \mathtt{t}'))$ for all $\mathtt{t}' \in \{\mathtt{t}^1, \ldots, \mathtt{t}^\alpha\}$. Otherwise by definition of $\mathfrak{s}'(\mathtt{x})(\mathtt{M})$,

$$\mathfrak{s}'(\mathtt{x})(\mathtt{M}) \ \notin \ \big\{\mathfrak{s}(\mathtt{t}^j)(\mathtt{M}) \ : \ j \in [1, \alpha], \mathfrak{c}(\mathrm{clen}(\mathtt{t}^j, \mathtt{t}^j)) > \mathtt{M}\big\}$$

and therefore not $\mathrm{clen}(\mathfrak{s}'(\mathtt{x}), \mathfrak{s}(\mathtt{t}^j)) > \mathtt{M}$, whence $\mathrm{clen}(\mathfrak{s}'(\mathtt{x}), \mathfrak{s}(\mathtt{t}^j)) = \mathtt{M}$.

Consequently, $\mathtt{t}$ does not belong to $\mathtt{t}^1, \ldots, \mathtt{t}^\alpha$ and therefore $\mathfrak{c}(\mathrm{clen}(\mathtt{t}, \mathtt{x})) < \mathtt{M}$. Indeed $\mathtt{M} = max\{\mathfrak{c}(\mathrm{clen}(\mathtt{t}, \mathtt{x})) : \mathtt{t} \in Y\}$. Remember that $\mathfrak{c}$ is string-compatible and more specifically it satisfies the formula

$$\bigwedge_{\mathtt{z},\mathtt{z}',\mathtt{z}'' \in Y \uplus \{\mathtt{x}\}} (\mathrm{clen}(\mathtt{z}, \mathtt{z}') < \mathrm{clen}(\mathtt{z}', \mathtt{z}'')) \Rightarrow (\mathrm{clen}(\mathtt{z}, \mathtt{z}') = \mathrm{clen}(\mathtt{z}, \mathtt{z}'')),$$

from condition (III) in Proposition 2. So, $\mathfrak{c}(\mathrm{clen}(\mathtt{t}, \mathtt{x})) = \mathfrak{c}(\mathrm{clen}(\mathtt{t}, \mathtt{t}^1))$ and we obtain that $\mathrm{clen}(\mathfrak{s}(\mathtt{t}), \mathfrak{s}(\mathtt{t}^1)) = \mathfrak{c}(\mathrm{clen}(\mathtt{t}, \mathtt{t}^1))$. So, for every $j \in [0, L-1]$ with $L = \mathrm{clen}(\mathfrak{s}(\mathtt{t}), \mathfrak{s}(\mathtt{t}^1))$, $\mathfrak{s}(\mathtt{t})(j) = \mathfrak{s}(\mathtt{t}^1)(j) = \mathfrak{s}'(\mathtt{x})(j)$.

If $\mathfrak{c}(\mathrm{len}(\mathtt{t})) = L$, then obviously, $\mathrm{clen}(\mathfrak{s}(\mathtt{t}), \mathfrak{s}'(\mathtt{x})) = \mathfrak{c}(\mathrm{clen}(\mathtt{t}, \mathtt{x}))$, which leads to a contradiction. Otherwise, we know that $\mathfrak{s}(\mathtt{t})(L) \neq \mathfrak{s}(\mathtt{t}^1)(L)$ and $\mathfrak{s}'(\mathtt{x})(L) = \mathfrak{s}(\mathtt{t}^1)(L)$ (by Step (2) and the fact that $L = \mathfrak{c}(\mathrm{clen}(\mathtt{t}, \mathtt{x})) < \mathfrak{c}(\mathrm{clen}(\mathtt{t}^1, \mathtt{x}))$); whence $\mathfrak{s}(\mathtt{t})(L) \neq \mathfrak{s}'(\mathtt{x})(L)$. So, we get $\mathrm{clen}(\mathfrak{s}(\mathtt{t}), \mathfrak{s}'(\mathtt{x})) = L$ and therefore $\mathrm{clen}(\mathfrak{s}(\mathtt{t}), \mathfrak{s}'(\mathtt{x})) = \mathfrak{c}(\mathrm{clen}(\mathtt{t}, \mathtt{x}))$, which leads to a contradiction. □

Lemma 7 below states a simple property about testing the satisfiability status of a Boolean combination of atomic formulae expressing prefix relationships. The NP upper bound can be obtained by reduction to the satisfiability problem for quantifier-free Presburger formulae [BT76,Pap81,Pot91]. Its proof technique involving Lemma 6 is reused for the full logic $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$ and for the logic $\mathrm{CTL}^\star(\Sigma^*, \mathrm{clen})$.

Lemma 7. *Let Σ be a non-empty alphabet equal to either $[0, k-1]$ for some $k \geq 2$ or $\mathbb{N}$. The satisfiability problem for Boolean combinations of atomic formulae from $\mathrm{LTL}(\Sigma^*, \preceq_p)$ [resp. from $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$] is* NP*-complete.*

Proof. NP-hardness is proved by an easy logarithmic-space reduction from SAT where each propositional variable p_i is encoded by the atomic formula $0 \preceq_p \mathtt{x}_i^{new}$ and $\mathtt{x}_i^{new}$ is a string variable dedicated to p_i.

Below, we establish the NP complexity upper bound for LTL$(\Sigma^*, \mathrm{clen})$, the proof for LTL$(\Sigma^*, \preceq_p)$ being then a direct corollary. Let ϕ be a Boolean combination of atomic formulae from LTL$(\Sigma^*, \mathrm{clen})$. Without any loss of generality, we can assume that the terms are of the form $\mathtt{x} \in \mathrm{SVAR}$ or $w \in \Sigma^*$. Indeed, if a term $\mathtt{X}^i\mathtt{x}$ occurs in ϕ, we can substitute it by a new variable $\mathtt{y}^{new}$, leading to an equi-satisfiable formula in LTL$(\Sigma^*, \mathrm{clen})$.

So, assume that ϕ contains the following terms only: the words $w_1, \ldots, w_q$ and the string variables $\mathtt{x}_1, \ldots, \mathtt{x}_r$. Let ϕ' be the formula obtained from ϕ by replacing every occurrence of w_i by the new string variable $\mathtt{y}_i$. Let us show that ϕ is satisfiable iff the formula below is satisfiable in quantifier-free Presburger arithmetic

$$\phi'' \quad \stackrel{\text{def}}{=} \quad \phi' \ \wedge\ \psi_{\mathrm{I}} \ \wedge\ \psi_{\mathrm{II}} \ \wedge\ \psi_{\mathrm{III}} \ \wedge \bigwedge_{i,j \in [1,q]} \mathrm{clen}(\mathtt{y}_i, \mathtt{y}_j) = \mathrm{clen}(w_i, w_j),$$

where the formulae ψ_{I}, ψ_{II} and ψ_{III} are those obtained from the conditions in Proposition 2 with the two disjoint sets of variables $Y = \{\mathtt{y}_1, \ldots, \mathtt{y}_q\}$ and $X = \{\mathtt{x}_1, \ldots, \mathtt{x}_r\}$. We have seen that $\psi_{\mathrm{I}} \ \wedge\ \psi_{\mathrm{II}} \ \wedge\ \psi_{\mathrm{III}}$ is of size $\mathcal{O}((q+r)^{k+2})$ if $\Sigma = [0, k-1]$ or of size $\mathcal{O}((q+r)^3)$ if $\Sigma = \mathbb{N}$. Similarly, size(ϕ') is linear in size(ϕ) and the last generalized conjunction is of quadratic size in size(ϕ). Hence, ϕ'' is of polynomial size in size(ϕ). Since the satisfiability problem for quantifier-free Presburger formulae is in NP, see e.g. [Pap81,BT76], we get the upper bound NP for solving our initial problem.

It remains to show that ϕ is satisfiable in LTL$(\Sigma^*, \mathrm{clen})$ iff ϕ'' is satisfiable in Presburger arithmetic.

First, let us suppose that there is a string valuation $\mathfrak{s}$ such that $\mathfrak{s} \models \phi'$ verifying for every $j \in [1, q]$, the equality $\mathfrak{s}(\mathtt{y}_j) = w_j$ (which is equivalent to state that ϕ is satisfiable). Let $\mathfrak{c}$ be the counter valuation such that for all $\mathtt{x}, \mathtt{x}' \in X \uplus Y$, we have $\mathfrak{c}(\mathrm{clen}(\mathtt{x}, \mathtt{x}')) \stackrel{\text{def}}{=} \mathrm{clen}(\mathfrak{s}(\mathtt{x}), \mathfrak{s}(\mathtt{x}'))$. Obviously, $\mathfrak{c} \models \phi' \wedge (\bigwedge_{i,j \in [1,q]} \mathrm{clen}(\mathtt{y}_i, \mathtt{y}_j) = \mathrm{clen}(w_i, w_j))$ where $\models$ is the satisfaction relation in Presburger arithmetic. Furthermore, by Lemma 5, we have that $\mathfrak{c} \models \psi_{\mathrm{I}} \ \wedge\ \psi_{\mathrm{II}} \ \wedge\ \psi_{\mathrm{III}}$. Consequently, ϕ'' is satisfiable in Presburger arithmetic.

Now suppose that there is a counter valuation $\mathfrak{c}$ such that $\mathfrak{c} \models \phi''$ and let $\mathfrak{s}$ be the string valuation such that for every $j \in [1, q]$, we have $\mathfrak{s}(\mathtt{y}_j) = w_j$. In particular, $\mathfrak{c}$ is string-compatible, $\mathfrak{c} \approx_Y \mathfrak{s}$, and by application of Lemma 6, there is a string valuation $\mathfrak{s}'$ that is a conservative extension of $\mathfrak{s}$, such that $\mathrm{clen}(\mathfrak{s}'(\mathtt{x}), \mathfrak{s}'(\mathtt{x}')) = \mathfrak{c}(\mathrm{clen}(\mathtt{x}, \mathtt{x}'))$ for all $\mathtt{x}, \mathtt{x}' \in X \cup Y$, i.e. $\mathfrak{c} \approx_{X \cup Y} \mathfrak{s}'$. Consequently, for all $\mathrm{clen}(\mathtt{t}_0, \mathtt{t}_0') \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}_1')$ occurring in ϕ' we have $\mathfrak{c} \models \mathrm{clen}(\mathtt{t}_0, \mathtt{t}_0') \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}_1')$ (in the sense of Presburger arithmetic) iff $\mathfrak{s}' \models \mathrm{clen}(\mathtt{t}_0, \mathtt{t}_0') \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}_1')$ (in the sense of LTL$(\Sigma^*, \mathrm{clen})$). Therefore, we get that $\mathfrak{s}' \models \phi'$ and ϕ is satisfiable since $\mathfrak{s}'(\mathtt{y}_j) = w_j$ for all $j \in [1, q]$. $\square$

4 Temporal logics on strings with prefix constraints

4.1 Linear-time temporal logics

Let Σ be a fixed non-empty (either finite, or infinite and countable) alphabet and ϕ be a formula in LTL(Σ^*, clen). Without any loss of generality, we can assume that the words in Σ^* occurring in ϕ are $w_1, \ldots, w_q$, the string variables occurring in ϕ are $\mathtt{x}_1, \ldots, \mathtt{x}_r$ and the other terms are among $\mathtt{Xx}_1, \ldots, \mathtt{Xx}_r$ (see Lemma 4).

We build a formula ϕ' in LTL($\mathbb{N}, \leq$) such that ϕ is satisfiable in LTL(Σ^*, clen) iff ϕ' is satisfiable in LTL($\mathbb{N}, \leq$) and size(ϕ') is polynomial in size(ϕ). Before defining ϕ', let us make preliminary remarks and constructions.

1. For all $i, j \in [1, q]$, we write $c_{i,j}$ to denote the constant value clen(w_i, w_j).
2. By convention, the counters in the formula ϕ' from LTL($\mathbb{N}, \leq$) are of the form clen($\mathtt{t}, \mathtt{t}'$) with $\mathtt{t}, \mathtt{t}' \in \mathtt{T}$ and $\mathtt{T} \stackrel{\text{def}}{=} \{\mathtt{y}_1, \ldots, \mathtt{y}_q\} \cup \{\mathtt{x}_i, \mathtt{Xx}_i : i \in [1, r]\}$. So, the expression clen($\mathtt{t}, \mathtt{t}'$) is overloaded: it may occur in the formula ϕ in LTL(Σ^*, clen) or as a (symbolic) counter in ϕ', but no confusion will be possible. Moreover, note that we have introduced a new set of variables $\{\mathtt{y}_1, \ldots, \mathtt{y}_q\}$ but each variable $\mathtt{y}_i$ shall be interpreted rigidly by taking the value w_i.
3. Let ϕ^{subst} be the formula in LTL($\mathbb{N}, \leq$) obtained from ϕ by replacing every occurrence of the word w_i by $\mathtt{y}_i$. This formula can be also viewed as a formula in LTL(Σ^*, clen). More generally, for every subformula ψ of ϕ, we write ψ^{subst} to denote the formula obtained from ψ by performing similar substitutions. Obviously, ψ^{subst} is a subformula of ϕ^{subst} too.
4. Let ϕ^{rig} be the formula in LTL($\mathbb{N}, \leq$) defined below:
$$\mathtt{G}\ (\bigwedge_{i,j \in [1,q]} (\text{clen}(\mathtt{y}_i, \mathtt{y}_j) = c_{i,j})).$$
5. Let ϕ^{next} be the formula in LTL($\mathbb{N}, \leq$) defined below:
$$\mathtt{G}\ (\bigwedge_{\mathtt{t},\mathtt{t}' \in \{\mathtt{y}_1,\ldots,\mathtt{y}_q\} \cup \{\mathtt{Xx}_1,\ldots,\mathtt{Xx}_r\}} \text{clen}(\mathtt{t}, \mathtt{t}') = \mathtt{X}\ \text{clen}(\mathtt{t} \setminus \mathtt{X}, \mathtt{t}' \setminus \mathtt{X})),$$
where $(\mathtt{Xx}_i)\backslash\mathtt{X} \stackrel{\text{def}}{=} \mathtt{x}_i$ and $\mathtt{y}_j\backslash\mathtt{X} \stackrel{\text{def}}{=} \mathtt{y}_j$. For example, an instance of a conjunct is $\text{clen}(\mathtt{Xx}_1, \mathtt{Xx}_2) = \mathtt{X}\ \text{clen}(\mathtt{x}_1, \mathtt{x}_2)$ where clen($\mathtt{Xx}_1, \mathtt{Xx}_2$) and clen($\mathtt{x}_1, \mathtt{x}_2$) are understood as counters in LTL($\mathbb{N}, \leq$).
6. The formula ψ_{I} is defined as $\psi_{\text{I}} \stackrel{\text{def}}{=} \mathtt{G}\ \psi'_{\text{I}}$ with
$$\psi'_{\text{I}} \stackrel{\text{def}}{=} \bigwedge_{\mathtt{t},\mathtt{t}' \in \mathtt{T}} (\text{clen}(\mathtt{t}, \mathtt{t}) \geq \text{clen}(\mathtt{t}, \mathtt{t}')).$$
7. The formula ψ_{II} is defined as $\psi_{\text{II}} \stackrel{\text{def}}{=} \mathtt{G}\ \psi'_{\text{II}}$ with either $\psi'_{\text{II}} \stackrel{\text{def}}{=} \top$ if $\Sigma = \mathbb{N}$ or the formula below if $\Sigma = [0, k-1]$
$$\psi'_{\text{II}} \stackrel{\text{def}}{=} \bigwedge_{\mathtt{t}_0,\ldots,\mathtt{t}_k \in \mathtt{T}} ((\bigwedge_{i \in [0,k]} (\text{clen}(\mathtt{t}_0, \mathtt{t}_1) < \text{clen}(\mathtt{t}_i, \mathtt{t}_i))) \ \wedge\ \text{clen}(\mathtt{t}_0, \mathtt{t}_1) = \cdots = \text{clen}(\mathtt{t}_0, \mathtt{t}_k)) \ \Rightarrow$$

$$(\bigvee_{i\neq j\in[1,k]} (\mathrm{clen}(\mathtt{t}_0,\mathtt{t}_1) < \mathrm{clen}(\mathtt{t}_i,\mathtt{t}_j))).$$

8. The formula ψ_{III} is defined as $\psi_{\mathrm{III}} \stackrel{\mathrm{def}}{=} \mathtt{G}\ \psi'_{\mathrm{III}}$ with

$$\psi'_{\mathrm{III}} \stackrel{\mathrm{def}}{=} \bigwedge_{\mathtt{t},\mathtt{t}',\mathtt{t}''\in\mathtt{T}} (\mathrm{clen}(\mathtt{t},\mathtt{t}') < \mathrm{clen}(\mathtt{t}',\mathtt{t}'')) \Rightarrow (\mathrm{clen}(\mathtt{t},\mathtt{t}') = \mathrm{clen}(\mathtt{t},\mathtt{t}'')).$$

The formula ϕ' is defined as the following conjunction:

$$\phi' \quad \stackrel{\mathrm{def}}{=} \quad \phi^{subst} \ \wedge\ \phi^{rig} \ \wedge\ \phi^{next} \ \wedge\ \psi_{\mathrm{I}} \ \wedge\ \psi_{\mathrm{II}} \ \wedge\ \psi_{\mathrm{III}}.$$

When $\Sigma = \mathbb{N}$, $\mathrm{size}(\phi')$ is in $\mathcal{O}(\mathrm{size}(\phi)^3)$ whereas $\mathrm{size}(\phi')$ is in $\mathcal{O}(\mathrm{size}(\phi)^{k+2})$ when $\Sigma = [0, k-1]$ for some $k \geq 2$.

Let us start by establishing the correctness of the reduction in one direction.

Lemma 8. *If ϕ is satisfiable then ϕ' is satisfiable.*

Proof. Let $\sigma : \mathbb{N} \to (\{\mathtt{x}_1,\ldots,\mathtt{x}_r\} \to \Sigma^*)$ be a model for $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$—actually restricted to the relevant string variables in $\{\mathtt{x}_1,\ldots,\mathtt{x}_r\}$—such that $\sigma, 0 \models \phi$. Let $\sigma' : \mathbb{N} \to ((\{\mathtt{x}_1,\ldots,\mathtt{x}_r\} \uplus \{\mathtt{y}_1,\ldots,\mathtt{y}_q\}) \to \Sigma^*)$ be the conservative extension of σ such that each string variable $\mathtt{y}_i$ is always interpreted by the string w_i. So, the formula ϕ^{subst} understood as a formula in $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$ verifies that $\sigma', 0 \models \phi^{subst}$.

Let $\sigma_{\mathbb{N}}$ be the model of $\mathrm{LTL}(\mathbb{N}, \leq)$ such that for all $i \in \mathbb{N}$, for all counters $\mathrm{clen}(\mathtt{t},\mathtt{t}')$ (built from terms in $\mathtt{T}$), we have $\sigma_{\mathbb{N}}(i)(\mathrm{clen}(\mathtt{t},\mathtt{t}')) \stackrel{\mathrm{def}}{=} \mathrm{clen}([\![\mathtt{t}]\!]_i, [\![\mathtt{t}']\!]_i)$ where $[\![\mathtt{y}_j]\!]_i = \sigma'(i)(\mathtt{y}_j) = w_j$, $[\![\mathtt{x}_j]\!]_i = \sigma'(i)(\mathtt{x}_j)$ and $[\![\mathtt{X}\mathtt{x}_j]\!]_i = \sigma'(i+1)(\mathtt{x}_j)$. It is quite immediate that $\sigma_{\mathbb{N}}, 0 \models \phi^{subst} \wedge \phi^{rig}$. In order to show that $\sigma_{\mathbb{N}}, 0 \models \phi^{next}$, by way of example let us show that $\sigma_{\mathbb{N}}, 0 \models \mathtt{G}\ (\mathrm{clen}(\mathtt{y}_j, \mathtt{X}\mathtt{x}_{j'}) = \mathtt{X}\ \mathrm{clen}(\mathtt{y}_j, \mathtt{x}_{j'}))$. Given $i \in \mathbb{N}$, we have

$$\sigma_{\mathbb{N}}(i)(\mathrm{clen}(\mathtt{y}_j, \mathtt{X}\mathtt{x}_{j'})) = \mathrm{clen}([\![\mathtt{y}_j]\!]_i, [\![\mathtt{X}\mathtt{x}_{j'}]\!]_i) = \mathrm{clen}(w_j, \sigma'(i+1)(\mathtt{x}_{j'}))$$

whereas

$$\sigma_{\mathbb{N}}(i+1)(\mathrm{clen}(\mathtt{y}_j, \mathtt{x}_{j'})) = \mathrm{clen}([\![\mathtt{y}_j]\!]_{i+1}, [\![\mathtt{x}_{j'}]\!]_{i+1}) = \mathrm{clen}(w_j, \sigma'(i+1)(\mathtt{x}_{j'})).$$

This means precisely that $\sigma_{\mathbb{N}}, i \models \mathrm{clen}(\mathtt{y}_j, \mathtt{X}\mathtt{x}_{j'}) = \mathtt{X}\ \mathrm{clen}(\mathtt{y}_j, \mathtt{x}_{j'})$. A similar simple reasoning can be performed for the other cases, and we conclude that $\sigma_{\mathbb{N}}, 0 \models \phi^{next}$.

Let $\mathfrak{s}_i$ be the string valuation with respect to the set $\mathtt{T}$ such that:

$$\begin{array}{lcll} \mathfrak{s}_i(\mathtt{x}_j) & \stackrel{\mathrm{def}}{=} & \sigma'(i)(\mathtt{x}_j) & \text{for all } j \in [1,r] \\ \mathfrak{s}_i(\mathtt{X}\mathtt{x}_j) & \stackrel{\mathrm{def}}{=} & \sigma'(i+1)(\mathtt{x}_j) & \text{for all } j \in [1,r] \\ \mathfrak{s}_i(\mathtt{y}_j) & \stackrel{\mathrm{def}}{=} & w_j & \text{for all } j \in [1,q] \end{array}$$

Note that $\mathfrak{s}_i$ and $\sigma_{\mathbb{N}}(i)$ satisfy the hypotheses from Lemma 5 since $\sigma_{\mathbb{N}}(i) \approx_{\mathtt{T}} \mathfrak{s}_i$, and therefore $\sigma_{\mathbb{N}}(i)$ is string-compatible. By definition of string-compatibility, this means that $\sigma_{\mathbb{N}}, i \models \psi'_{\mathrm{I}} \wedge \psi'_{\mathrm{II}} \wedge \psi'_{\mathrm{III}}$. Since the position i above is arbitrary and $\mathtt{G}(\psi'_{\mathrm{I}} \wedge \psi'_{\mathrm{II}} \wedge \psi'_{\mathrm{III}})$ is logically equivalent to $\psi_{\mathrm{I}} \wedge \psi_{\mathrm{II}} \wedge \psi_{\mathrm{III}}$, we get that $\sigma_{\mathbb{N}}, 0 \models \psi_{\mathrm{I}} \wedge \psi_{\mathrm{II}} \wedge \psi_{\mathrm{III}}$. As a conclusion, $\sigma_{\mathbb{N}}, 0 \models \phi'$ and therefore ϕ' is satisfiable. □

Let us now establish the correctness of the reduction in the other direction.

Lemma 9. *If ϕ' is satisfiable then ϕ is satisfiable.*

Proof. Let $\sigma : \mathbb{N} \to (\{\mathrm{clen}(\mathtt{t},\mathtt{t}') : \mathtt{t},\mathtt{t}' \in \mathtt{T}\} \to \mathbb{N})$ be a model for the formula ϕ', i.e. $\sigma, 0 \models \phi'$. For every $i \in \mathbb{N}$, we can consider that $\sigma(i)$ is a counter valuation with respect to the set $\mathtt{T}$. Let us define a model σ_{Σ^*} for $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$ such that $\sigma_{\Sigma^*}, 0 \models \phi$ inductively on the position i.
Base case ($i = 0$): We have seen that $\sigma(0)$ is a counter valuation with respect to $\mathtt{T}$. Let $\mathfrak{s}$ be a string valuation such that for every $j \in [1,q]$, we have $\mathfrak{s}(\mathtt{y}_j) \stackrel{\text{def}}{=} w_j$. Since $\sigma, 0 \models \phi'$, this implies that $\sigma, 0 \models \psi'_{\mathrm{I}} \wedge \psi'_{\mathrm{II}} \wedge \psi'_{\mathrm{III}}$. So, by definition, $\sigma(0)$ is string-compatible. Moreover, $\mathrm{clen}(\mathfrak{s}(\mathtt{y}_i), \mathfrak{s}(\mathtt{y}_j)) = \sigma(0)(\mathrm{clen}(\mathtt{y}_i, \mathtt{y}_j)) = c_{i,j}$ (by satisfaction of the formula ϕ^{rig}) for all $i, j \in [1,q]$, i.e. $\sigma(0) \approx_{Y_0} \mathfrak{s}$ with $Y_0 = \{\mathtt{y}_1, \ldots, \mathtt{y}_q\}$. By application of Lemma 6, there is a string valuation $\mathfrak{s}'$ that is a conservative extension of $\mathfrak{s}$, such that $\mathrm{clen}(\mathfrak{s}'(\mathtt{t}), \mathfrak{s}'(\mathtt{t}')) = \sigma(0)(\mathrm{clen}(\mathtt{t},\mathtt{t}'))$ for all $\mathtt{t},\mathtt{t}' \in \mathtt{T}$, i.e. $\sigma(0) \approx_{\mathtt{T}} \mathfrak{s}'$. Consequently, for all counters $\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0)$, $\mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1)$, we have $\sigma, 0 \models \mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1)$ iff $\mathfrak{s}' \models \mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1)$. We define $\sigma_{\Sigma^*}(0)$ and $\sigma_{\Sigma^*}(1)$ as follows:

$$\begin{array}{lcll} \sigma_{\Sigma^*}(0)(\mathtt{y}_j) & \stackrel{\text{def}}{=} & \sigma_{\Sigma^*}(1)(\mathtt{y}_j) \stackrel{\text{def}}{=} w_j & \text{for all } j \in [1,q] \\ \sigma_{\Sigma^*}(0)(\mathtt{x}_j) & \stackrel{\text{def}}{=} & \mathfrak{s}'(\mathtt{x}_j) & \text{for all } j \in [1,r] \\ \sigma_{\Sigma^*}(1)(\mathtt{x}_j) & \stackrel{\text{def}}{=} & \mathfrak{s}'(\mathtt{X}\mathtt{x}_j) & \text{for all } j \in [1,r] \end{array}$$

For all $\mathtt{t}$, $\mathtt{t}'$ in $\mathtt{T}$, we have $\mathrm{clen}([\![\mathtt{t}]\!]_0, [\![\mathtt{t}']\!]_0)$ in σ_{Σ^*} is equal to $\sigma(0)(\mathrm{clen}(\mathtt{t},\mathtt{t}'))$.

Induction step: Suppose that σ_{Σ^*} is defined until the position $i \geq 1$, and let us define $\sigma_{\Sigma^*}(i{+}1)$. The induction hypothesis assumes that for all $j \in [0, i-1]$, for all terms $\mathtt{t}$ and $\mathtt{t}'$ in $\mathtt{T}$ we have that $\mathrm{clen}([\![\mathtt{t}]\!]_j, [\![\mathtt{t}']\!]_j) = \sigma(j)(\mathrm{clen}(\mathtt{t},\mathtt{t}'))$. So, for all $j < i$, for all counters $\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0)$, $\mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1)$, we also have

$$\sigma, j \models \mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1) \;\text{ iff }\; \sigma_{\Sigma^*}, j \models \mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1) \qquad (1)$$

Let $\mathfrak{s}$ be a string valuation such that for every $j \in [1,q]$, we have $\mathfrak{s}(\mathtt{y}_j) = w_j$ and for every $j \in [1,r]$, $\mathfrak{s}(\mathtt{x}_j) = \sigma_{\Sigma^*}(i)(\mathtt{x}_j)$. Since $\sigma, 0 \models \phi'$, this implies that $\sigma, i \models \psi'_{\mathrm{I}} \wedge \psi'_{\mathrm{II}} \wedge \psi'_{\mathrm{III}}$ and therefore $\sigma(i)$ is string-compatible with respect to $X \cup Y$ where $Y = Y_0 \cup \{\mathtt{x}_1, \ldots, \mathtt{x}_r\}$ and $X = \{\mathtt{X}\mathtt{x}_1, \ldots, \mathtt{X}\mathtt{x}_r\}$. Moreover, for all $\mathtt{t},\mathtt{t}' \in Y$, $\mathrm{clen}(\mathfrak{s}(\mathtt{t}), \mathfrak{s}(\mathtt{t}')) = \sigma(i)(\mathrm{clen}(\mathtt{t},\mathtt{t}'))$ by the induction hypothesis, i.e. $\sigma(i) \approx_Y \mathfrak{s}$. Indeed, by way of example, we have the following equalities:

$$\begin{array}{lcll} \mathrm{clen}(\mathfrak{s}(\mathtt{x}_j), \mathfrak{s}(\mathtt{x}_{j'})) & = & \mathrm{clen}(\sigma_{\Sigma^*}(i)(\mathtt{x}_j), \sigma_{\Sigma^*}(i)(\mathtt{x}_{j'})) & \text{(by definition of } \mathfrak{s}) \\ & = & \mathrm{clen}([\![\mathtt{x}_j]\!]_i, [\![\mathtt{x}_{j'}]\!]_i) & \text{(by definition of } [\![\cdot]\!]) \\ & = & \mathrm{clen}([\![\mathtt{X}\mathtt{x}_j]\!]_{i-1}, [\![\mathtt{X}\mathtt{x}_{j'}]\!]_{i-1}) & \text{(by definition of } [\![\cdot]\!]) \\ & = & \sigma(i-1)(\mathrm{clen}(\mathtt{X}\mathtt{x}_j, \mathtt{X}\mathtt{x}_{j'})) & \text{(by IH)} \\ & = & \sigma(i)(\mathrm{clen}(\mathtt{x}_j, \mathtt{x}_{j'})) & \text{(since } \sigma, 0 \models \phi^{next}) \end{array}$$

By application of Lemma 6, there is a string valuation $\mathfrak{s}'$ that is a conservative extension of $\mathfrak{s}$, such that $\sigma(i) \approx_{X \cup Y} \mathfrak{s}'$. Let us define $\sigma_{\Sigma^*}(i+1)$ as follows:

$$\begin{array}{lcll} \sigma_{\Sigma^*}(i+1)(\mathtt{y}_j) & \stackrel{\text{def}}{=} & w_j & \text{for all } j \in [1,q] \\ \sigma_{\Sigma^*}(i+1)(\mathtt{x}_j) & \stackrel{\text{def}}{=} & \mathfrak{s}'(\mathtt{X}\mathtt{x}_j) & \text{for all } j \in [1,r] \end{array}$$

Observe that for all $j \in [0,i]$, for all terms $\mathtt{t}$, $\mathtt{t}'$, we have that $\mathrm{clen}([\![\mathtt{t}]\!]_j, [\![\mathtt{t}']\!]_j) = \sigma(j)(\mathrm{clen}(\mathtt{t},\mathtt{t}'))$, whence for all counters $\mathrm{clen}(\mathtt{t}_0,\mathtt{t}'_0)$, $\mathrm{clen}(\mathtt{t}_1,\mathtt{t}'_1)$, $\sigma, i \models \mathrm{clen}(\mathtt{t}_0,\mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1,\mathtt{t}'_1)$ iff $\sigma_{\Sigma^*}, i \models \mathrm{clen}(\mathtt{t}_0,\mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1,\mathtt{t}'_1)$, establishing (1).

Now that the construction of σ_{Σ^*} is done, for all $i \in \mathbb{N}$, we have $\sigma, i \models \mathrm{clen}(\mathtt{t}_0,\mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1,\mathtt{t}'_1)$ iff $\sigma_{\Sigma^*}, i \models \mathrm{clen}(\mathtt{t}_0,\mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1,\mathtt{t}'_1)$ for all expressions $\mathrm{clen}(\mathtt{t}_0,\mathtt{t}'_0)$, $\mathrm{clen}(\mathtt{t}_1,\mathtt{t}'_1)$ occurring in ϕ^{subst}. By structural induction, one can show that for all $i \in \mathbb{N}$, we get $\sigma, i \models \psi^{subst}$ iff $\sigma_{\Sigma^*}, i \models \psi^{subst}$ for all subformulae ψ of ϕ. The base case with atomic formulae of the form $\mathrm{clen}(\mathtt{t}_0,\mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1,\mathtt{t}'_1)$ is obtained by construction of σ_{Σ^*} (see above) whereas the cases for the Boolean connectives in the induction step are by an easy verification. By way of example, below, we consider the case $\psi = \varphi_1 \mathtt{U} \varphi_2$.

$$\begin{array}{lll} \sigma, i \models (\varphi_1 \mathtt{U} \varphi_2)^{subst} & \text{iff} & \sigma, i \models (\varphi_1)^{subst}\ \mathtt{U}\ (\varphi_2)^{subst} \qquad \text{(by definition of } (\cdot)^{subst}) \\ & \text{iff} & \text{there is } j \geq i \text{ such that } \sigma, j \models (\varphi_2)^{subst} \text{ and for all } i \leq k < j, \\ & & \text{we have } \sigma, k \models (\varphi_1)^{subst} \qquad \text{(by definition of } \models) \\ & \text{iff} & \text{there is } j \geq i \text{ such that } \sigma_{\Sigma^*}, j \models (\varphi_2)^{subst} \text{ and for all } i \leq k < j, \\ & & \text{we have } \sigma_{\Sigma^*}, k \models (\varphi_1)^{subst} \qquad \text{(by induction hypothesis)} \\ & \text{iff} & \sigma_{\Sigma^*}, i \models (\varphi_1)^{subst}\ \mathtt{U}\ (\varphi_2)^{subst} \text{ (by definition of } \models \text{ in LTL}(\Sigma^*, \mathrm{clen})) \\ & \text{iff} & \sigma_{\Sigma^*}, i \models (\varphi_1 \mathtt{U} \varphi_2)^{subst} \qquad \text{(by definition of } (\cdot)^{subst}) \end{array}$$

So, since $\sigma, 0 \models \phi^{subst}$, we conclude that $\sigma_{\Sigma^*}, 0 \models \phi^{subst}$ (when ϕ^{subst} is viewed as a formula in $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$). Moreover, for all $j \in [1,q]$ and for all $i \in \mathbb{N}$, we have $\sigma_{\Sigma^*}(i)(\mathtt{y}_j) = w_j$, whence $\sigma_{\Sigma^*}, 0 \models \phi$ too. □

Theorem 1. *For every non-empty countable alphabet Σ, the respective satisfiability problems for* $\mathrm{LTL}(\Sigma^*, \preceq_p)$ *and* $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$ *are* PSPACE*-complete.*

By Lemmas 8 and 9, there is a logarithmic-space reduction from the satisfiability problem for $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$ to the satisfiability problem for $\mathrm{LTL}(\mathbb{N}, \leq)$ and we know that the satisfiability problem for $\mathrm{LTL}(\mathbb{N}, \leq)$ is PSPACE-complete.

As a nice corollary, we state below a small alphabet property for checking satisfiability of formulae in $\mathrm{LTL}(\mathbb{N}^*, \preceq_p)$—i.e., the alphabet is countably infinite. This is due to the fact that the construction of the value β in the proof of Lemma 6 is bounded by $\mathrm{card}(X \cup Y)$ which corresponds to a bound $3 \times \mathrm{size}(\phi)$ in the context of the proof of Lemma 9, where it is invoked in an essential way. This assumes that no term of the form $\mathtt{X}^i\mathtt{x}$ with $i > 1$ occurs in ϕ (see Lemma 4).

Corollary 1. *Let ϕ be a satisfiable formula in LTL($\mathbb{N}^*, \preceq_p$) with maximal letter N and with no term of the form $\mathtt{X}^i\mathtt{x}$ with $i > 1$. Then, ϕ has a model in which all the strings are interpreted in the subalphabet $[0, max(N, 3 \times \mathrm{size}(\phi))]$.*

As a consequence, there is no formula ϕ in $\mathrm{LTL}(\mathbb{N}^*, \preceq_p)$ that characterizes exactly the set of models with an infinite number of letters from $\mathbb{N}$ to occur in it. Note that if terms of the form

$\mathtt{X}^i\mathtt{x}$ with $i > 1$ occur in ϕ, a renaming is performed first, as shown in the proof of Lemma 4 (which, in the worst case, may cause a quadratic blow-up to the size of the subalphabet). We also get the following consequence by noting that the satisfiability problem for $\mathrm{LTL}(\mathbb{N}^*, \preceq_p)$ is equivalent to the problem stated in the corollary below.

Corollary 2. *The problem below is* PSPACE*-complete:*

input: *a formula ϕ built over a finite set X of letters,*
question: *is there an alphabet $\Sigma \supseteq X$ such that ϕ is satisfiable in* $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$*?*

Indeed, given ϕ with maximal letter N, we first reduce it to an equi-satisfiable formula ϕ' with maximal letter bounded by size(ϕ) (by renaming of the letters) and then we take advantage of the polynomial-space bound from Corollary 1.

The proof method from [DD07,DG08] for establishing that the satisfiability problem for $\mathrm{LTL}(\mathbb{N}, \leq)$ is in PSPACE can be applied to any extension $\mathfrak{L}$ of LTL that uses Büchi automata similarly to LTL. Nevertheless, the automata built in [DD07] to solve $\mathrm{LTL}(\mathbb{N}, \leq)$ satisfiability problem are obtained as the product of a standard automaton (roughly speaking, following the construction from [VW94]) and an automaton to deal with satisfiable symbolic models. Any extension of LTL behaving as LTL for the first part of the product would provide a PSPACE upper bound too. Let us be a bit more precise. Let $\mathfrak{L}$ be an extension of LTL such that for every formula ϕ of $\mathfrak{L}$, we can construct a Büchi automaton $\mathcal{B}_\phi$ of size at most exponential in size(ϕ) such that:

1. $\mathcal{B}_\phi$ accepts exactly the models of ϕ,
2. the size of each state of $\mathcal{B}_\phi$ is polynomial in size(ϕ),
3. it can be checked if a state is initial [resp. accepting] in polynomial space in size(ϕ),
4. each transition of $\mathcal{B}_\phi$ can be checked in polynomial space in size(ϕ).

Past LTL [Mar04], linear μ-calculus μLTL [Var88] or Extended Temporal Logic ETL [Wol83] are well-known examples of such logics. It is then easy to design the extensions of the form $\mathfrak{L}(\Sigma^*, \preceq_p)$.

Corollary 3. *Let $\mathfrak{L}$ be an extension of LTL satisfying the above properties and Σ be a non-empty and countable alphabet. Then, the satisfiability problem for $\mathfrak{L}(\Sigma^*, \preceq_p)$ is* PSPACE*-complete.*

Such results can be pushed a bit further either by noting that any extension of $\mathrm{LTL}(\mathbb{N}, \leq)$ obtained by adding a finite amount of MSO-definable temporal operators remains in the complexity class PSPACE [DG08, Section 5] (see also [GK03]) or by considering a richer set of relations on natural numbers, as those considered in the constrained language IPC* [DG08] (see also the survey on LTL with Presburger constraints in [Dem06]). Typically, one could add the past-time temporal operators $\mathtt{X}^{-1}$ and $\mathtt{S}$ while preserving the PSPACE upper bounds stated so far.

4.2 Branching-time temporal logics

Developments for LTL(Σ^*, clen) using Lemma 6 can be lifted to the branching-time case, even though we can *only* establish decidability without providing an optimal complexity upper bound as we did for LTL(Σ^*, clen).

First, let us define the branching-time temporal logic CTL*(Σ^*, clen) extending the linear-time logic LTL(Σ^*, clen). Formulae in the logic CTL*(Σ^*, clen) are defined as follows:

$$\phi \quad ::= \quad \mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1) \;\mid\; \neg(\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1)) \;\mid$$

$$\phi \wedge \phi \;\mid\; \phi \vee \phi \;\mid\; \mathtt{X}\,\phi \;\mid\; \phi\,\mathtt{U}\,\phi \;\mid\; \phi\,\mathtt{R}\,\phi \;\mid\; \mathtt{A}\,\phi \;\mid\; \mathtt{E}\,\phi$$

Negation appears only in front of atomic formulae, and each temporal operator occurs with its dual operator. Nevertheless, we still have the full power of CTL* [EH86] in terms of path quantifiers and linear-time temporal operators. This restriction on negation will simplify a few developments in proofs. Below, we use the standard abbreviation $\mathtt{AG}\,\phi \stackrel{\text{def}}{=} \mathtt{A}(\bot\,\mathtt{R}\,\phi)$. Formulae in CTL*($\Sigma^*$, clen) contain the constraints on the longest common prefixes while the version with prefix constraints only, is easy to design and it can be translated to CTL*(Σ^*, clen) similarly to the linear-time case.

A model for CTL*(Σ^*, clen) is a structure of the form $\mathfrak{M} = (S, R, L)$ such that S is a non-empty set of states, $R \subseteq S \times S$ is a total binary relation (totality means that every state has at least one successor state) and L is a labelling function $L : S \to (\mathrm{SVAR} \to \Sigma^*)$. For all models $\mathfrak{M}$, infinite paths π in $\mathfrak{M}$ and positions $i \in \mathbb{N}$, the satisfaction relation $\models$ is defined as follows:

– $\pi, i \models \mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1) \;\stackrel{\text{def}}{\Leftrightarrow}\; \mathrm{clen}([\![\mathtt{t}_0]\!]_{\pi,i}, [\![\mathtt{t}'_0]\!]_{\pi,i}) \leq \mathrm{clen}([\![\mathtt{t}_1]\!]_{\pi,i}, [\![\mathtt{t}'_1]\!]_{\pi,i})$ where

$$\begin{array}{lcll} [\![w]\!]_{\pi,i} & \stackrel{\text{def}}{=} & w & \text{for all } w \in \Sigma^* \\ [\![\mathtt{x}]\!]_{\pi,i} & \stackrel{\text{def}}{=} & L(\pi(i))(\mathtt{x}) & \text{for every } \mathtt{x} \in \mathrm{SVAR} \\ [\![\mathtt{X}^j\mathtt{x}]\!]_{\pi,i} & \stackrel{\text{def}}{=} & L(\pi(i+j))(\mathtt{x}) & \text{for all } \mathtt{x} \in \mathrm{SVAR} \text{ and } j \geq 1 \end{array}$$

– $\pi, i \models \neg\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1) \;\stackrel{\text{def}}{\Leftrightarrow}\; \mathrm{clen}([\![\mathtt{t}_0]\!]_{\pi,i}, [\![\mathtt{t}'_0]\!]_{\pi,i}) > \mathrm{clen}([\![\mathtt{t}_1]\!]_{\pi,i}, [\![\mathtt{t}'_1]\!]_{\pi,i})$.

$\pi, i \models \mathtt{X}\phi$	$\stackrel{\text{def}}{\Leftrightarrow}$	$\pi, i+1 \models \phi$
$\pi, i \models \phi\ \mathtt{U}\ \psi$	$\stackrel{\text{def}}{\Leftrightarrow}$	there is $j \geq i$ such that $\pi, j \models \psi$ and for all $i \leq k < j$, we have $\pi, k \models \phi$.
$\pi, i \models \phi\ \mathtt{R}\ \psi$	$\stackrel{\text{def}}{\Leftrightarrow}$	either $\pi, j \models \psi$ for all $j \geq i$ or, there is $j \geq i$ such that $\pi, j \models \phi$ and for all $i \leq k \leq j$, we have $\pi, k \models \psi$
$\pi, i \models \mathtt{E}\phi$	$\stackrel{\text{def}}{\Leftrightarrow}$	there is a path π' that agrees with π on the positions $0, \ldots, i$ such that $\pi', i \models \phi$
$\pi, i \models \mathtt{A}\phi$	$\stackrel{\text{def}}{\Leftrightarrow}$	for all paths π' that agree with π on the positions $0, \ldots, i$, we have $\pi', i \models \phi$

An instance of the satisfiability problem for $\mathrm{CTL}^\star(\Sigma^*, \mathrm{clen})$ consists in checking whether a formula ϕ in $\mathrm{CTL}^\star(\Sigma^*, \mathrm{clen})$ has a model $\mathfrak{M}$ and a path π such that $\pi, 0 \models \phi$.

We write $\mathrm{CTL}^\star(\mathbb{N}, \leq)$ to denote $\mathrm{CTL}^\star(\Sigma^*, \mathrm{clen})$ when Σ is a singleton alphabet and the atomic formulae are restricted to inequalities between terms. In the sequel, we take advantage of the recent result below, partly based on the decidability of Boolean combinations of formulae from MSO and WMSO+U [BT12] where U is the unbounding second-order quantifier.

Proposition 3. *[CKL13] The satisfiability problem for* $\mathrm{CTL}^\star(\mathbb{N}, \leq)$ *is decidable.*

As for $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$, we state below a few properties about $\mathrm{CTL}^\star(\Sigma^*, \mathrm{clen})$ that will simplify a bit the developments. The properties can either be proved as for $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$ or are easily inherited from $\mathrm{CTL}^\star$ [ES84,EH86].

(PROP1) There is a logarithmic-space reduction from $\mathrm{CTL}^\star(\Sigma^*, \mathrm{clen})$ satisfiability problem into its restriction in which terms are restricted to $w \in \Sigma^*$, $\mathtt{x}$ or $\mathtt{X}\mathtt{x}$ with $\mathtt{x} \in \mathrm{SVAR}$.

(PROP2) A formula ϕ is satisfiable in $\mathrm{CTL}^\star(\Sigma^*, \mathrm{clen})$ iff ϕ has a model that is a $(k_\phi + 1)$-branching tree where k_ϕ is the number of existential path quantifiers occurring in ϕ (see also [Gas09,CKL13] or [ES84, Theorem 3.2] for the classical argument on $\mathrm{CTL}^\star$).

Let Σ be a fixed non-empty (either finite or infinite and countable) alphabet and ϕ be a formula in $\mathrm{CTL}^\star(\Sigma^*, \mathrm{clen})$. Without any loss of generality, we can assume that the words in Σ^* occurring in ϕ are $w_1, \ldots, w_q$, the string variables occurring in ϕ are $\mathtt{x}_1, \ldots, \mathtt{x}_r$ and the other terms are among $\mathtt{X}\mathtt{x}_1, \ldots, \mathtt{X}\mathtt{x}_r$ (see (PROP1) above). Let $d = k_\phi + 1$. By (PROP2), we can restrict ourselves to consider models of ϕ which are d-branching trees.

Below, we build a formula ϕ' in $\mathrm{CTL}^\star(\mathbb{N}, \leq)$ such that ϕ is satisfiable in $\mathrm{CTL}^\star(\Sigma^*, \mathrm{clen})$ iff ϕ' is satisfiable in $\mathrm{CTL}^\star(\mathbb{N}, \leq)$ and $\mathrm{size}(\phi')$ is polynomial in $\mathrm{size}(\phi)$. Before defining ϕ', let us make preliminary remarks and constructions.

1. For all $i, j \in [1, q]$, we write $c_{i,j}$ to denote the constant value $\mathrm{clen}(w_i, w_j)$.
2. For every $\gamma \in [1, d]$, we introduce a counter $\mathtt{z}_\gamma$ and we write p_γ (understood as a propositional variable) to denote the formula $(\mathtt{z}_\gamma = 0)$.
   Let ϕ^{dtree} be the formula stating that each reachable state satisfies a unique formula p_j:
   $$\mathtt{A}\mathtt{G}\Big(\bigvee_{\gamma \in [1,d]} \big(p_\gamma \wedge \bigwedge_{\gamma' \neq \gamma \in [1,d]} \neg p_{\gamma'}\big)\Big).$$
   So, along a path, to identify the direction $\gamma \in [1, d]$ for the next position, it is sufficient to test $\mathtt{X}p_\gamma$.
3. By convention, the counters in the formula ϕ' from $\mathrm{CTL}^\star(\mathbb{N}, \leq)$ are of the form $\mathrm{clen}(\mathtt{t}, \mathtt{t}')^\gamma$ with $\gamma \in [1, d]$, $\mathtt{t}, \mathtt{t}' \in \mathtt{T}$ and $\mathtt{T} \stackrel{\text{def}}{=} \{\mathtt{y}_1, \ldots, \mathtt{y}_q\} \cup \{\mathtt{x}_i, \mathtt{X}\mathtt{x}_i : i \in [1, r]\}$. There are also the counters of the form $\mathtt{z}_\gamma$. This is the most substantial difference with the linear case since each direction γ comes with its set of counters. We do not claim here a new idea since this is just a relevant adaptation of arguments from [Gas09,CKL13] but at the logical level. For instance, similar arguments are used at the level of tree automata in [Gas09], which can be done even more explicitly for automata accepting d-branching trees only.

4. Let ϕ^{subst} be the formula in $\mathrm{CTL}^\star(\mathbb{N},\leq)$ obtained from ϕ by replacing every occurrence of the word w_i by $\mathtt{y}_i$ and then,
   - by replacing every (positive) occurrence of $\mathrm{clen}(\mathtt{t}_0,\mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1,\mathtt{t}'_1)$ by the formula below:
   $$\bigwedge_{\gamma\in[1,d]} \big((\mathtt{X}p_\gamma) \Rightarrow \mathrm{clen}(\mathtt{t}_0,\mathtt{t}'_0)^\gamma \leq \mathrm{clen}(\mathtt{t}_1,\mathtt{t}'_1)^\gamma\big).$$
   - by replacing every occurrence of $\neg(\mathrm{clen}(\mathtt{t}_0,\mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1,\mathtt{t}'_1))$ by the formula below:
   $$\bigwedge_{\gamma\in[1,d]} \big((\mathtt{X}p_\gamma) \Rightarrow \neg(\mathrm{clen}(\mathtt{t}_0,\mathtt{t}'_0)^\gamma \leq \mathrm{clen}(\mathtt{t}_1,\mathtt{t}'_1)^\gamma)\big).$$

   This is one of the places where it is convenient to have formulae in negation normal form (at the minor cost of introducing dual operators). Again, for every subformula ψ of ϕ, we write ψ^{subst} to denote the formula obtained from ψ by performing similar substitutions.
5. Let ϕ^{rig} be the formula in $\mathrm{CTL}^\star(\mathbb{N},\leq)$ defined below:
   $$\mathtt{AG}\ (\bigwedge_{\gamma\in[1,d],i,j\in[1,q]} (\mathrm{clen}(\mathtt{y}_i,\mathtt{y}_j)^\gamma = c_{i,j})).$$
6. Let ϕ^{next} be the formula in $\mathrm{CTL}^\star(\mathbb{N},\leq)$ defined below:
   $$\mathtt{AG}\ (\bigwedge_{\gamma\in[1,d]} ((\mathtt{X}p_\gamma) \Rightarrow (\bigwedge_{\mathtt{t},\mathtt{t}'\in\{\mathtt{y}_1,\ldots,\mathtt{y}_q\}\cup\{\mathtt{X}\mathtt{x}_1,\ldots,\mathtt{X}\mathtt{x}_r\}} \mathrm{clen}(\mathtt{t},\mathtt{t}')^\gamma = \mathtt{X}\ \mathrm{clen}(\mathtt{t}\setminus\mathtt{X},\mathtt{t}'\setminus\mathtt{X})^\gamma))).$$
7. ϕ^{agree} be the formula in $\mathrm{CTL}^\star(\mathbb{N},\leq)$ defined below:
   $$\mathtt{AG}\ (\bigwedge_{\gamma,\gamma'\in[1,d]} \bigwedge_{\mathtt{t},\mathtt{t}'\in\{\mathtt{y}_1,\ldots,\mathtt{y}_q\}\cup\{\mathtt{x}_1,\ldots,\mathtt{x}_r\}} \mathrm{clen}(\mathtt{t},\mathtt{t}')^\gamma = \mathrm{clen}(\mathtt{t},\mathtt{t}')^{\gamma'}).$$

   This a new formula by comparison to the linear case, that states constraints related to the lengths of the longest common prefixes for string variables interpreted at the current position. The values do not depend on the directions of child nodes. The linear case roughly corresponds to $d = 1$ and therefore the formula ϕ^{agree} with $d = 1$ is valid but useless in the linear case.
8. The formula ψ_{I} is defined as $\psi_{\mathrm{I}} \stackrel{\mathrm{def}}{=} \mathtt{AG}\ \psi'_{\mathrm{I}}$ with
   $$\psi'_{\mathrm{I}} \stackrel{\mathrm{def}}{=} \bigwedge_{\gamma\in[1,d]} (\bigwedge_{\mathtt{t},\mathtt{t}'\in\mathtt{T}} (\mathrm{clen}(\mathtt{t},\mathtt{t})^\gamma \geq \mathrm{clen}(\mathtt{t},\mathtt{t}')^\gamma)).$$
9. The formula ψ_{II} is defined as $\psi_{\mathrm{II}} \stackrel{\mathrm{def}}{=} \mathtt{AG}\ \psi'_{\mathrm{II}}$ with either $\psi'_{\mathrm{II}} \stackrel{\mathrm{def}}{=} \top$ if $\Sigma = \mathbb{N}$ or the formula below if $\Sigma = [0,k-1]$
   $$\psi'_{\mathrm{II}} \stackrel{\mathrm{def}}{=} \bigwedge_{\gamma\in[1,d]} (\bigwedge_{\mathtt{t}_0,\ldots,\mathtt{t}_k\in\mathtt{T}} ((\bigwedge_{i\in[0,k]} (\mathrm{clen}(\mathtt{t}_0,\mathtt{t}_1)^\gamma < \mathrm{clen}(\mathtt{t}_i,\mathtt{t}_i)^\gamma)) \wedge$$
   $$\mathrm{clen}(\mathtt{t}_0,\mathtt{t}_1)^\gamma = \cdots = \mathrm{clen}(\mathtt{t}_0,\mathtt{t}_k)^\gamma \Rightarrow (\bigvee_{i\neq j\in[1,k]} (\mathrm{clen}(\mathtt{t}_0,\mathtt{t}_1)^\gamma < \mathrm{clen}(\mathtt{t}_i,\mathtt{t}_j)^\gamma)))).$$

10. The formula ψ_{III} is defined as $\psi_{\text{III}} \stackrel{\text{def}}{=} \mathtt{AG}\ \psi'_{\text{III}}$ with

$$\psi'_{\text{III}} \stackrel{\text{def}}{=} \bigwedge_{\gamma\in[1,d]} \Big(\bigwedge_{\mathtt{t},\mathtt{t}',\mathtt{t}''\in\mathtt{T}} (\text{clen}(\mathtt{t},\mathtt{t}')^\gamma < \text{clen}(\mathtt{t}',\mathtt{t}'')^\gamma) \Rightarrow (\text{clen}(\mathtt{t},\mathtt{t}')^\gamma = \text{clen}(\mathtt{t},\mathtt{t}'')^\gamma)\Big).$$

The formulae ψ_{I}, ψ_{II} and ψ_{III} are defined as in the linear case except that d distinct directions have to be considered. The formula ϕ' is equal to the following conjunction:

$$\phi^{dtree} \ \wedge\ \phi^{subst} \ \wedge\ \phi^{rig} \ \wedge\ \phi^{next} \ \wedge\ \phi^{agree} \ \wedge\ \psi_{\text{I}} \ \wedge\ \psi_{\text{II}} \ \wedge\ \psi_{\text{III}}.$$

Observe that some of subformulae in the conjunction above are not defined in negation normal form but it is easy to get such a form (mainly by removing the occurrences of material implication).

Correctness of the reduction is stated below.

Lemma 10. *ϕ is satisfiable iff ϕ' is satisfiable.*

The proof is quite similar to the proofs for Lemma 8 and for Lemma 9 except that for each direction γ in $[1,d]$ for the next state in a path, we consider a distinct family of counters of the form $\text{clen}(\mathtt{t},\mathtt{t}')^\gamma$. Actually, these two lemmas can be viewed as Lemma 10 when $d=1$ (a few adjustments would be necessary to have a formal correspondence). But when the terms $\mathtt{t}$ and $\mathtt{t}'$ only refer to values at the current position, $\text{clen}(\mathtt{t},\mathtt{t}')^\gamma$ and $\text{clen}(\mathtt{t},\mathtt{t}')^{\gamma'}$ have to agree even if $\gamma \neq \gamma'$. Moreover, whether we consider a model for ϕ or for ϕ', it is sufficient to consider models that are infinite branching trees. Given $X \subseteq \mathtt{T}$ and $\gamma \in [1,d]$, we write $\mathfrak{c} \approx^{\gamma}_{X} \mathfrak{s}$ to denote the agreement between $\mathfrak{c}$ and $\mathfrak{s}$ with respect to X but restricted further to the counters of the form $\text{clen}(\cdot,\cdot)^\gamma$.

Proof. ($\Rightarrow$) First, let us show that if ϕ is satisfiable, then ϕ' is satisfiable. Let $\mathfrak{M} = (S,R,L)$ be a model and π_0 be an infinite path in $\mathfrak{M}$ such that $\pi_0, 0 \models \phi$. Without any loss of generality, we can assume that

- $\mathfrak{M}$ is a d-branching tree with $d = k_\phi{+}1$ where k_ϕ is the number of occurrences of existential path quantifiers in ϕ, i.e. $S = [1,d]^*$ and wRw' iff $w' = w\cdot\gamma$ for some $\gamma \in [1,d]$.
- L is of the form $[1,d]^* \to (\{\mathtt{x}_1,\ldots,\mathtt{x}_r\} \to \Sigma^*)$, assuming that the string variables in ϕ are among $\{\mathtt{x}_1,\ldots,\mathtt{x}_r\}$.
- π_0 is an infinite path starting at the root $\varepsilon \in [1,d]^*$. Equivalently, π_0 can be represented by an ω-word in $[1,d]^\omega$.

Let $\mathfrak{M}' = (S,R,L')$ be the conservative extension of $\mathfrak{M}$ such that each string variable $\mathtt{y}_i$ is everywhere interpreted as the string w_i. Let $\mathfrak{M}_{\mathbb{N}} = (S,R,L_{\mathbb{N}})$ be the model in $\text{CTL}^\star(\mathbb{N},\leq)$ defined as follows:

- For all $w \in S$ and $\gamma,\gamma' \in [1,d]$, $L_{\mathbb{N}}(w\cdot\gamma)(\mathtt{z}_{\gamma'}) \stackrel{\text{def}}{=} 0$ if $\gamma = \gamma'$, otherwise $L_{\mathbb{N}}(w\cdot\gamma)(\mathtt{z}_{\gamma'}) \stackrel{\text{def}}{=} 1$. Moreover $L_{\mathbb{N}}(\varepsilon)(\mathtt{z}_1) \stackrel{\text{def}}{=} 0$ and $L_{\mathbb{N}}(\varepsilon)(\mathtt{z}_\gamma) \stackrel{\text{def}}{=} 1$ for all $\gamma \in [2,d]$ (this is a convention since ε has no parent). It is easy to show that $\pi_0, 0 \models \phi^{dtree}$.
- For all $w \in S$ and for all $\gamma \in [1,d]$, $L_{\mathbb{N}}(w)(\text{clen}(\mathtt{t},\mathtt{t}')^\gamma) \stackrel{\text{def}}{=} \text{clen}([\![\mathtt{t}]\!]_{w,w\gamma}, [\![\mathtt{t}']\!]_{w,w\gamma})$ where

$$\begin{array}{lcl}
[\![\mathtt{y}_j]\!]_{w,w\gamma} & \stackrel{\text{def}}{=} & w_j \\
[\![\mathtt{x}_j]\!]_{w,w\gamma} & \stackrel{\text{def}}{=} & L'(w)(\mathtt{x}_j) \\
[\![\mathtt{X}\mathtt{x}_j]\!]_{w,w\gamma} & \stackrel{\text{def}}{=} & L'(w\gamma)(\mathtt{x}_j)
\end{array}$$

Note that for all infinite paths π' of the form $\pi'_0(w)(w\gamma)\pi'_1$, the value $[\![\mathtt{t}]\!]_{w,w\gamma}$ defined above is equal to $[\![\mathtt{t}]\!]_{\pi',\mathrm{len}(w)}$ in $\mathfrak{M}'$. So, for all $\gamma \in [1,d]$, for all $w \in [1,d]^*$, $L_{\mathbb{N}}(w) \approx^{\gamma}_{\mathtt{T}} [\![\cdot]\!]_{w,w\gamma}$ where $[\![\cdot]\!]_{w,w\gamma}$ is understood in the obvious way as a string valuation with respect to $\mathtt{T}$.

It is now quite easy to show that $\pi_0, 0 \models \phi^{subst} \wedge \phi^{rig} \wedge \phi^{agree}$, in particular $\pi_0, 0 \models \phi^{subst}$ is proved by structural induction and by considering all paths in $\mathfrak{M}_{\mathbb{N}}$. Indeed, one can show that for all paths π' starting at ε, for all positions i and for all subformulae ψ of ϕ, we have $\pi', i \models \psi$ in $\mathfrak{M}'$ implies $\pi', i \models \psi^{subst}$ in $\mathfrak{M}_{\mathbb{N}}$.

Let us start by showing the base case. Suppose that $\pi', i \models (\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1))$ with $\pi'(i+1) = \gamma \in [1,d]$ and $w = \pi'(0)\cdots\pi(i)'$ ($\pi'(0) = \varepsilon$). For each $j \in \{0,1\}$, we have

$$\begin{array}{lcll}
L_{\mathbb{N}}(w)(\mathrm{clen}(\mathtt{t}_j, \mathtt{t}'_j)^{\gamma}) & = & \mathrm{clen}([\![\mathtt{t}_j]\!]_{w,w\gamma}, [\![\mathtt{t}'_j]\!]_{w,w\gamma}) & \text{(by definition of } L_{\mathbb{N}}(w)) \\
& = & \mathrm{clen}([\![\mathtt{t}_j]\!]_{\pi',i}, [\![\mathtt{t}'_j]\!]_{\pi',i}) & \text{(in } \mathfrak{M}')
\end{array}$$

So, we get that $\pi', i \models (\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0)^{\gamma} \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1)^{\gamma})$. In $\mathfrak{M}_{\mathbb{N}}$, we have $\pi', i \models \mathtt{X}p_{\gamma}$ since $L_{\mathbb{N}}(w\gamma)(\mathtt{z}_{\gamma}) = 0$ and by definition, p_{γ} is equal to $\mathtt{z}_{\gamma} = 0$. So, in $\mathfrak{M}_{\mathbb{N}}$, we get $\pi', i \models \mathtt{X}p_{\gamma} \Rightarrow (\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0)^{\gamma} \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1)^{\gamma})$. By contrast, for all $\gamma' \neq \gamma$, we have $L_{\mathbb{N}}(w\gamma)(\mathtt{z}_{\gamma'}) = 1$ and therefore in $\mathfrak{M}_{\mathbb{N}}$, we obtain $\pi', i \not\models \mathtt{X}p_{\gamma'}$, which "vacuously" guarantees that $\pi', i \models \mathtt{X}p_{\gamma'} \Rightarrow (\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0)^{\gamma'} \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1)^{\gamma'})$. So, $\pi', i \models (\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1))^{subst}$. The proof for the base case with $\neg(\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1))$ is very similar. By way of example, below we deal with the induction step for the cases $\psi = \mathtt{E}\ \psi'$ and $\psi = \psi_1 \mathtt{U} \psi_2$.

Case $\psi = \mathtt{E}\ \psi'$*:*

$$\begin{array}{lll}
\pi', i \models \mathtt{E}\ \psi' & \text{implies} & \text{there is } \pi'' \text{ such that } \pi'', i \models \psi' \text{ and,} \\
& & \pi' \text{ and } \pi'' \text{ agree on the position } 0, \ldots, i \text{ (by definition of } \models) \\
& \text{implies} & \text{there is } \pi'' \text{ such that } \pi'', i \models (\psi')^{subst} \text{ in } \mathfrak{M}_{\mathbb{N}} \text{ and,} \\
& & \pi' \text{ and } \pi'' \text{ agree on the position } 0, \ldots, i \qquad \text{(by IH)} \\
& \text{implies} & \pi', i \models \mathtt{E}\ (\psi')^{subst} \text{ in } \mathfrak{M}_{\mathbb{N}} \qquad \text{(by definition of } \models) \\
& \text{implies} & \pi', i \models (\mathtt{E}\ \psi')^{subst} \qquad \text{(by definition of } (\cdot)^{subst})
\end{array}$$

Case $\psi = \psi_1 \mathtt{U} \psi_2$*:*

$$\begin{array}{lll}
\pi', i \models \psi_1 \mathtt{U} \psi_2 & \text{implies} & \text{there is } j \geq i \text{ such that } \pi', j \models \psi_2 \text{ and} \\
& & \text{for all } k \in [i, j-1], \text{ we have } \pi', k \models \psi_1 \text{ (by definition of } \models) \\
& \text{implies} & \text{there is } j \geq i \text{ such that } \pi', j \models (\psi_2)^{subst} \text{ in } \mathfrak{M}_{\mathbb{N}} \text{ and} \\
& & \text{for all } k \in [i, j-1], \text{ we have } \pi', k \models (\psi_1)^{subst} \text{ in } \mathfrak{M}_{\mathbb{N}} \text{ (by IH)} \\
& \text{implies} & \pi', i \models (\psi_1)^{subst}\ \mathtt{U}\ (\psi_2)^{subst} \text{ in } \mathfrak{M}_{\mathbb{N}} \qquad \text{(by definition of } \models) \\
& \text{implies} & \pi', i \models (\psi_1 \mathtt{U} \psi_2)^{subst} \qquad \text{(by definition of } (\cdot)^{subst})
\end{array}$$

For the satisfaction of ϕ^{agree}, it is sufficient to observe that the values $[\![\mathtt{y}_j]\!]_{w,w\gamma}$ and $[\![\mathtt{x}_j]\!]_{w,w\gamma}$ do not depend on the direction γ. Moreover, the satisfaction of the conjunctive formula $\phi^{next} \wedge \psi_{\mathrm{I}} \wedge \psi_{\mathrm{II}} \wedge \psi_{\mathrm{III}}$ can be proved exactly as in the linear case (see the proof of Lemma 8); in particular the property $L_{\mathbb{N}}(w) \approx^{\gamma}_{\mathtt{T}} [\![\cdot]\!]_{w,w\gamma}$ for all w and γ, is crucial for the

satisfaction of $\psi_{\mathrm{I}} \wedge \psi_{\mathrm{II}} \wedge \psi_{\mathrm{III}}$.

($\Leftarrow$) Now, let us prove the other direction, namely, if ϕ' is satisfiable, then ϕ is satisfiable. Let $\mathfrak{M} = (S, R, L)$ be a model and π_0 be an infinite path in $\mathfrak{M}$ such that $\pi_0, 0 \models \phi'$. Without any loss of generality, we can assume that

- $\mathfrak{M}$ is D-branching tree with $D = k_{\phi'} + 1$, i.e. $S = [1, D]^*$ and wRw' iff $w' = w \cdot \gamma$ for some $\gamma \in [1, D]$.
- L is of the form $[1, D]^* \to (\{\mathrm{clen}(\mathtt{t}, \mathtt{t}')^\gamma : \mathtt{t}, \mathtt{t}' \in \mathtt{T},\ \gamma \in [1, d]\} \cup \{\mathtt{z}_1, \ldots, \mathtt{z}_\gamma\} \to \mathbb{N})$.
- π_0 is an infinite path starting at the root $\varepsilon \in [1, D]^*$.

Observe that size(ϕ') is clearly strictly greater than size(ϕ) but no new existential path quantifier is introduced in the reduction, so $k_{\phi'} = k_\phi$. However, the proof below does not depend on the value D since what really matters is the existence of a D-branching tree model, for some D.

So, for all $w \in S$ and for all $\gamma \in [1, d]$, $L(w)$ restricted to γ-counters (i.e., counters indexed by γ), written below $L(w)_{|\gamma}$, can be viewed as a counter valuation with respect to $\mathtt{T}$. Let us define a model $\mathfrak{M}_{\Sigma^*} = (S, R, L_{\Sigma^*})$ in CTL$^\star(\Sigma^*, \mathrm{clen})$ such that $\pi_0, 0 \models \phi$, where L_{Σ^*} is inductively defined on the nodes w as follows. Below we use the following abbreviations for the sake of clarity.

$$\begin{array}{rcl} p_{\leq} & \stackrel{\text{def}}{=} & \mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1) \\ p^s_{\leq} & \stackrel{\text{def}}{=} & (\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1))^{subst} \\ \neg p_{\leq} & \stackrel{\text{def}}{=} & \neg(\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1)) \\ \neg p^s_{\leq} & \stackrel{\text{def}}{=} & (\neg\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0) \leq \mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1))^{subst} \end{array}$$

Base case ($w = \varepsilon$): Let $\gamma \in [1, d]$. We have seen that $L(\varepsilon)_{|\gamma}$ can be viewed as a counter valuation over the γ-counters defined from the terms in $\mathtt{T}$. Let $\mathfrak{s}_\gamma$ be a string valuation such that for every $j \in [1, q]$, we have $\mathfrak{s}_\gamma(\mathtt{y}_j) \stackrel{\text{def}}{=} w_j$. Since $\pi_0, 0 \models \phi'$, this implies that $\pi_0, 0 \models \psi'_{\mathrm{I}} \wedge \psi'_{\mathrm{II}} \wedge \psi'_{\mathrm{III}}$. So, by definition, the restriction of $L(\varepsilon)_{|\gamma}$ is string-compatible. Moreover, $\mathrm{clen}(\mathfrak{s}_\gamma(\mathtt{y}_i), \mathfrak{s}_\gamma(\mathtt{y}_j)) = L(\varepsilon)(\mathrm{clen}(\mathtt{y}_i, \mathtt{y}_j)^\gamma) = c_{i,j}$ for all $i, j \in [1, q]$ (by satisfaction of the formula ϕ^{rig}). Consequently, $L(\varepsilon) \approx^\gamma_{Y_0} \mathfrak{s}_\gamma$ with $Y_0 = \{\mathtt{y}_1, \ldots, \mathtt{y}_q\}$. By application of Lemma 6, there is a string valuation $\mathfrak{s}'_\gamma$ that is a conservative extension of $\mathfrak{s}_\gamma$, such that $\mathrm{clen}(\mathfrak{s}'_\gamma(\mathtt{t}), \mathfrak{s}'_\gamma(\mathtt{t}')) = \mathfrak{c}(\mathrm{clen}(\mathtt{t}, \mathtt{t}')^\gamma)$ for all $\mathtt{t}, \mathtt{t}' \in \mathtt{T}$, i.e. $L(\varepsilon) \approx^\gamma_{\mathtt{T}} \mathfrak{s}'_\gamma$. Note that for all $\mathtt{t} \in \{\mathtt{x}_1, \ldots, \mathtt{x}_r\} \uplus \{\mathtt{y}_1, \ldots, \mathtt{y}_q\}$, for all $\gamma, \gamma' \in [1, d]$, $\mathfrak{s}'_\gamma(\mathtt{t}) = \mathfrak{s}'_{\gamma'}(\mathtt{t})$ since $\pi_0, 0 \models \phi^{agree}$ and the algorithm in the proof of Lemma 6 is deterministic (assuming that the first string variables for which we assign a string are ordered as $\mathtt{x}_1, \ldots, \mathtt{x}_r$).

We define $L_{\Sigma^*}(\varepsilon)$ as the string valuation $\mathfrak{s}'_1$ (equivalently, any $\mathfrak{s}'_\gamma$ could be used here) restricted to variables in $\{\mathtt{x}_1, \ldots, \mathtt{x}_r\} \uplus \{\mathtt{y}_1, \ldots, \mathtt{y}_q\}$. Moreover, for every $\gamma \in [1, D]$ such that $L_{\Sigma^*}(\gamma)(\mathtt{z}_{\gamma'}) = 0$ for some $\gamma' \in [1, d]$, $L_{\Sigma^*}(\gamma)$ is defined as the restriction of the string valuation $\mathfrak{s}'_{\gamma'}$ such that for all $j \in [1, r]$, we have $L_{\Sigma^*}(\gamma)(\mathtt{x}_j) \stackrel{\text{def}}{=} \mathfrak{s}'_{\gamma'}(\mathtt{X}\mathtt{x}_j)$. Of course, we also require that for all $j \in [1, q]$, we have $L_{\Sigma^*}(\gamma)(\mathtt{y}_j) \stackrel{\text{def}}{=} w_j$. So, for all counters $\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0)$, $\mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1)$, $\pi_0, 0 \models p^s_{\leq}$ implies $\pi_0, 0 \models p_{\leq}$ (in $\mathfrak{M}_{\Sigma^*}$) and $\pi_0, 0 \models \neg p^s_{\leq}$ implies $\pi_0, 0 \models \neg p_{\leq}$. These last properties still hold for any path π' starting from ε.

Induction step: Suppose that $\mathfrak{M}_{\Sigma^*}$ is defined until the state $w \neq \varepsilon$, for all its strict prefixes and for all the extensions of the strict prefixes by adding exactly one more direction. Let us define $L_{\Sigma^*}(w\gamma)$ for all $\gamma \in [1, D]$. The induction hypothesis also assumes that for all terms $\mathtt{t}$, $\mathtt{t}'$ in $\mathtt{T}$, for all $j \in [1, i]$, for all infinite paths π with prefix $w(-1)\cdots w(j-2)$ (by convention $w(-1) = \varepsilon$), for all $\gamma' \in [1, d]$ with $L(w(-1)\cdots w(j-2)\pi(j))(\mathtt{z}_{\gamma'}) = 0$, we have

$$\mathrm{clen}([\![\mathtt{t}]\!]_{\pi,j-1}, [\![\mathtt{t}']\!]_{\pi,j-1}) = L(w(-1)\cdots w(j-2))(\mathrm{clen}(\mathtt{t}, \mathtt{t}')^{\gamma'}).$$

So, for all paths π' visiting w at the $i = \mathrm{len}(w)$th position, for all $j < i$, for all counters $\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0)$, $\mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1)$, $\pi', j \models p^s_{\leq}$ implies $\pi', j \models p_{\leq}$ (in $\mathfrak{M}_{\Sigma^*}$) and $\pi', j \models \neg p^s_{\leq}$ implies $\pi', j \models \neg p_{\leq}$. Observe that when $\mathrm{len}(w) = 1$, we have indeed the satisfaction of the above condition in the induction hypothesis since this amounts to check that for all terms $\mathtt{t}$, $\mathtt{t}'$ in $\mathtt{T}$, for all $\gamma' \in [1, d]$, for all infinite paths π with $L(\pi(1))(\mathtt{z}_{\gamma'}) = 0$, $\mathrm{clen}([\![\mathtt{t}]\!]_{\pi,0}, [\![\mathtt{t}']\!]_{\pi,0}) = L(\varepsilon)(\mathrm{clen}(\mathtt{t}, \mathtt{t}')^{\gamma'})$.

Let γ' be the unique element of $[1, d]$ such that $L(w\gamma)(\mathtt{z}_{\gamma'}) = 0$ (existence and unicity are guaranteed by satisfaction of ϕ^{dtree}). We have seen that $L(w)_{|\gamma'}$ can be viewed as a counter valuation over the γ'-counters with respect to $\mathtt{T}$. Let $\mathfrak{s}$ be a string valuation such that for every $j \in [1, q]$, we have $\mathfrak{s}(\mathtt{y}_j) \stackrel{\text{def}}{=} w_j$ and for every $j \in [1, r]$, we have $\mathfrak{s}(\mathtt{x}_j) \stackrel{\text{def}}{=} L_{\Sigma^*}(w)(\mathtt{x}_j)$. Since $\pi_0, 0 \models \phi'$, this implies that $\pi', \mathrm{len}(w) \models \psi'_{\mathrm{I}} \wedge \psi'_{\mathrm{II}} \wedge \psi'_{\mathrm{III}}$ for any path π' starting at ε and visiting w. So, by definition, the restriction of $L(w)_{|\gamma'}$ is string-compatible with respect to $\mathtt{T} = X \cup Y$ where $Y = Y_0 \cup \{\mathtt{x}_1, \ldots, \mathtt{x}_r\}$ and $X = \{\mathtt{X}\mathtt{x}_1, \ldots, \mathtt{X}\mathtt{x}_r\}$. Moreover, $\mathrm{clen}(\mathfrak{s}(\mathtt{y}_i), \mathfrak{s}(\mathtt{y}_j)) = L(w)(\mathrm{clen}(\mathtt{y}_i, \mathtt{y}_j)^{\gamma'}) = c_{i,j}$ (by satisfaction of the formula ϕ^{rig}) for all $i, j \in [1, q]$. Moreover, for all $\mathtt{t}, \mathtt{t}' \in Y$, $\mathrm{clen}(\mathfrak{s}(\mathtt{t}), \mathfrak{s}(\mathtt{t}')) = L(w)(\mathrm{clen}(\mathtt{t}, \mathtt{t}')^{\gamma'})$ by the induction hypothesis, i.e. $L(w) \approx^{\gamma'}_{Y} \mathfrak{s}$. By application of Lemma 6, there is a string valuation $\mathfrak{s}'$ that is a conservative extension of $\mathfrak{s}$, such that $\mathrm{clen}(\mathfrak{s}'(\mathtt{t}), \mathfrak{s}'(\mathtt{t}')) = L(w)(\mathrm{clen}(\mathtt{t}, \mathtt{t}')^{\gamma'})$ for all $\mathtt{t}, \mathtt{t}' \in \mathtt{T}$, i.e. $L(w) \approx^{\gamma'}_{\mathtt{T}} \mathfrak{s}'$.

We define $L_{\Sigma^*}(w\gamma)$ such that for all $j \in [1, r]$, we have $L_{\Sigma^*}(w\gamma)(\mathtt{x}_j) \stackrel{\text{def}}{=} \mathfrak{s}'(\mathtt{X}\mathtt{x}_j)$. Of course, we also require that for all $j \in [1, q]$, we have $L_{\Sigma^*}(w\gamma)(\mathtt{y}_j) \stackrel{\text{def}}{=} w_j$. Hence, assuming $w' = w\gamma$, for all terms $\mathtt{t}$, $\mathtt{t}'$ in $\mathtt{T}$, for all $j \in [1, i+1]$, for all infinite paths π with prefix $w'(-1)\cdots w'(j-2)$ for all $\gamma' \in [1, d]$ with $L(w'(-1)\cdots w'(j-2)\pi(j))(\mathtt{z}_{\gamma'}) = 0$, we have

$$\mathrm{clen}([\![\mathtt{t}]\!]_{\pi,j-1}, [\![\mathtt{t}']\!]_{\pi,j-1}) = L(w'(-1)\cdots w'(j-2))(\mathrm{clen}(\mathtt{t}, \mathtt{t}')^{\gamma'}).$$

This guarantees the propagation of the induction hypothesis. Only the case $j = i + 1$ needs to be newly established (which amounts to check that $\mathrm{clen}(\mathfrak{s}'(\mathtt{t}), \mathfrak{s}'(\mathtt{t}')) = L(w)(\mathrm{clen}(\mathtt{t}, \mathtt{t}')^{\gamma'})$ but $\mathfrak{s}'$ is precisely defined to satisfy such a property).

Now that the construction of $\mathfrak{M}_{\Sigma^*}$ is done, we have that for all paths π' starting at ε, for all $i \in \mathbb{N}$, $\pi', i \models p^s_{\leq}$ implies $\pi', i \models p_{\leq}$ for all expressions $\mathrm{clen}(\mathtt{t}_0, \mathtt{t}'_0)$, $\mathrm{clen}(\mathtt{t}_1, \mathtt{t}'_1)$ occurring in ϕ. Similarly, $\pi', i \models \neg p^s_{\leq}$ implies $\pi', i \models \neg p_{\leq}$. Actually, by structural induction, one can show that for all paths π' starting at ε, for all $i \in \mathbb{N}$, $\pi', i \models (\psi)^{subst}$ implies $\pi', i \models \psi$ (in $\mathfrak{M}_{\Sigma^*}$) where ψ is a subformula of ϕ. We have seen that the base case of the induction is already true and the cases in the induction step for Boolean connectives and for linear-time temporal operators can be done as in the proof of Lemma 6. By way of example, let us consider the case when $\psi = \mathtt{A}\ \varphi$.

$\pi', i \models (\mathtt{A}\ \varphi)^{subst}$	implies	$\pi', i \models \mathtt{A}\ \varphi^{subst}$	(by definition of $(\cdot)^{subst}$)
	implies	for all paths π'' that agree on π' on the positions $0, \ldots, i$, we have $\pi'', i \models \varphi^{subst}$	(by definition of $\models$)
	implies	for all paths π'' that agree on π' on the positions $0, \ldots, i$, we have $\pi'', i \models \varphi$ in $\mathfrak{M}_{\Sigma^*}$	(by IH)
	implies	$\pi', i \models \mathtt{A}\ \varphi$ in $\mathfrak{M}_{\Sigma^*}$	(by definition of $\models$)

Since $\pi_0, 0 \models \phi^{subst}$, we conclude that $\pi_0, 0 \models \phi$ (in $\mathfrak{M}_{\Sigma^*}$). □

Theorem 2. *For every non-empty countable alphabet Σ, the satisfiability problem for the branching-time temporal logic* $\mathrm{CTL}^\star(\Sigma^*, \mathrm{clen})$ *is decidable.*

By Lemma 10, there is a satisfiability-preserving reduction from $\mathrm{CTL}^\star(\Sigma^*, \mathrm{clen})$ to $\mathrm{CTL}^\star(\mathbb{N}, \leq)$ and we know that the satisfiability problem for $\mathrm{CTL}^\star(\mathbb{N}, \leq)$ is decidable [CKL13]. Again, as for the linear-time case, we can state a small alphabet property for checking satisfiability of formulae in $\mathrm{CTL}^\star(\mathbb{N}^*, \preceq_p)$.

Corollary 4. *Let ϕ be a satisfiable formula in $CTL^\star(\mathbb{N}^*, \preceq_p)$ with maximal letter N and with no occurrence of $\mathtt{X}^i\mathtt{x}$ with $i > 1$. Then, ϕ has a model in which all the strings are interpreted in the subalphabet $[0, max(N, 3 \times \mathrm{size}(\phi))]$.*

5 Conclusion

We have shown that the logic $\mathrm{LTL}(\Sigma^*, \preceq_p)$ admits a PSPACE-complete satisfiability problem (by using the PSPACE upper bound from [DD07,DG08] for the logic $\mathrm{LTL}(\mathbb{N}, \leq)$) and that $\mathrm{CTL}^\star(\Sigma^*, \mathrm{clen})$ is decidable by taking advantage of the decidability result on natural numbers from [CKL13]. This solves open problems mentioned in [DG05,Dem07,Gas07,CKL13,CKL14] and our proof technique allows us to get a PSPACE upper bound for other extensions of LTL such as linear μ-calculus or Past LTL. A similar result has been very recently shown independently in [KW15] with a different proof technique based on constraint automata. Our proof method is quite original, explicit and conceptually simple since prefix constraints are transformed into length constraints about longest common prefixes, so that only constraints on counters are ultimately considered. Since the prefix relation is not a total ordering (unlike a lexicographical ordering on Σ^* for example), it is not possible to take advantage of techniques from [ST11] dedicated to temporal logics on concrete domains that are linearly ordered structures. Moreover, our results imply an EXPSPACE upper bound for the **uniform satisfiability problem** defined below but it is open whether this is an optimal upper bound (the alphabet is now part of the input):

input: alphabet $\Sigma = [0, k-1]$ (k in unary) or $\Sigma = \mathbb{N}$, and a formula ϕ in $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$
question: is ϕ satisfiable in $\mathrm{LTL}(\Sigma^*, \mathrm{clen})$?

The recent work [KW15] answers positively to this question.

By using the obvious symmetry between the prefix relation $\preceq_p$ and the suffix relation $\preceq_s$, we can also conclude that the satisfiability problem for $\mathrm{LTL}(\Sigma^*, \preceq_s)$ is PSPACE-complete and

$\mathrm{CTL}^\star(\Sigma^*, \preceq_s)$ is decidable. Interestingly enough, when both the prefix relation *and* the suffix relation are in the concrete domain, the decidability status of $\mathrm{LTL}(\Sigma^*, \preceq_p, \preceq_s)$ remains open and it is unclear whether the counter abstraction presented herein for the prefix relation can be extended further with the additional presence of the suffix relation. This also applies if we consider regularity constraints with the prefix constraints. By contrast, adding word equations at the atomic level is a natural extension but this leads to undecidability since concatenation allows to simulate increments and decrements in Minsky machines. Considering the subword relation $\sqsubseteq$ would be another direction to investigate as the decidability status of $\mathrm{LTL}(\{0,1\}^*, \sqsubseteq)$ is open too. It is obvious that the fact that $\sqsubseteq$ is a well-quasi-ordering by Higman's Lemma can be used to design a decision procedure.

Funding: This work was supported by the EU Seventh Framework Programme [grant number PIOF-GA-2011-301166 (DATAVERIF) to S.D.]; the U.S. Air Force Office of Scientific Research [grant number FA9550-09-1-0596 to M.D.]; and the U.S. National Science Foundation [grant number 0644299 to M.D.]. This work was done, in part, while the first author visited the ACSys group at New York University.

Acknowledgments: We would like to thank W. Plandowski (Warsaw University) for explanations about the encoding of Boolean combinations of prefix constraints as word equations. Furthermore, we thank Th. Colcombet (LIAFA, Paris) for several discussions we had that helped us to clarify some aspects of this work. We thank also Ph. Schnoebelen (LSV, Cachan) for directing our attention to the work [Kus06]. Furthermore we are deeply indebted to the two anonymous referees for their suggestions and remarks that help us to improve the manuscript; this has been highly appreciated especially for the proof of Lemma 6 and for simplifications in the formulae built in Section 4.

References

[AAC+14] P.A. Abdulla, M.F. Atig, Y.-F. Chen, L. Holík, A. Rezine, Ph. Rümmer, and J. Stenman. String constraints for verification. In *CAV'14*, volume 8559 of *Lecture Notes in Computer Science*, pages 150–166. Springer, 2014.

[BC02] Ph. Balbiani and J.F. Condotta. Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning. In *FroCoS'02*, volume 2309 of *Lecture Notes in Artificial Intelligence*, pages 162–173. Springer, 2002.

[BC06] M. Bojańczyk and Th. Colcombet. Bounds in w-Regularity. In *LICS'06*, pages 285–296. IEEE Computer Society, 2006.

[BG06] L. Bozzelli and R. Gascon. Branching-time temporal logic extended with Presburger constraints. In *LPAR'06*, volume 4246 of *Lecture Notes in Computer Science*, pages 197–211. Springer, 2006.

[BGL12] F. Baader, S. Ghilardi, and C. Lutz. LTL over description logic axioms. *ACM Transactions on Computational Logic*, 13(3):21, 2012.

[Boj04] M. Bojańczyk. A bounding quantifier. In *CSL'04*, volume 3210 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

[BPT15] M. Bojańczyk, P. Parys, and S. Torunczyk. The MSO+U theory of (N, <) is undecidable. *CoRR*, abs/1502.04578, 2015.

[BT76] I. Borosh and L. Treybig. Bounds on positive integral solutions of linear Diophantine equations. *American Mathematical Society*, 55:299–304, 1976.

[BT12] M. Bojańczyk and S. Torunczyk. Weak MSO+U over infinite trees. In *STACS'12*, LIPIcs, pages 648–660, 2012.

[Čer94] K. Čerāns. Deciding properties of integral relational automata. In *ICALP'94*, volume 820 of *Lecture Notes in Computer Science*, pages 35–46. Springer, 1994.

[CKL13] C. Carapelle, A. Kartzow, and M. Lohrey. Satisfiability of CTL* with constraints. In *CONCUR'13*, Lecture Notes in Computer Science, pages 455–463. Springer, 2013.

[CKL14] C. Carapelle, A. Kartzow, and M. Lohrey. Satisfiability of ECTL* with constraints, October 2014. Under submission. Retrieved from `http://www.informatik.uni-leipzig.de/~kartzow/Paper/ECTL-With-Constraints.pdf`.

[DD07] S. Demri and D. D'Souza. An automata-theoretic approach to constraint LTL. *Information & Computation*, 205(3):380–415, 2007.

[Dec92] R. Dechter. From local to global consistency. *Artificial Intelligence*, pages 87–107, 1992.

[Dem06] S. Demri. Linear-time temporal logics with Presburger constraints: An overview. *Journal of Applied Non-Classical Logics*, 16(3–4):311–347, 2006.

[Dem07] S. Demri. *Logiques pour la spécification et vérification.* Mémoire d'habilitation, Université Paris 7, Paris, France, 2007.

[DG05] S. Demri and R. Gascon. Verification of qualitative $\mathbb{Z}$-constraints. In *CONCUR'05*, volume 3653 of *Lecture Notes in Computer Science*, pages 518–532. Springer, 2005.

[DG08] S. Demri and R. Gascon. Verification of qualitative Z constraints. *Theoretical Computer Science*, 409(1):24–40, 2008.

[DHV14] A. Deutsch, R. Hull, and V. Vianu. Automatic verification of database-centric systems. *SIGMOD Record*, 43(3):5–17, 2014.

[DJP14] V. Diekert, A. Jez, and W. Plandowski. Finding all solutions of equations in free groups and monoids with involution. In *CSR'14*, volume 8476 of *Lecture Notes in Computer Science*, pages 1–15, 2014.

[EH86] E.A. Emerson and J. Halpern. 'Sometimes' and 'not never' revisited: on branching versus linear time temporal logic. *Journal of the Association for Computing Machinery*, 33:151–178, 1986.

[ES84] E.A. Emerson and P. Sistla. Deciding full branching time logic. *Information and Control*, 61:175–201, 1984.

[Gab98] D. Gabbay. *Fibring Logics.* Oxford University Press, 1998.

[Gas07] R. Gascon. *Spécification et vérification de propriétés quantitatives sur des automates à contraintes.* PhD thesis, ENS de Cachan, November 2007.

[Gas09] R. Gascon. An automata-based approach for CTL* with constraints. *Electronic Notes in Theoretical Computer Science*, 239:193–211, 2009.

[GK03] P. Gastin and D. Kuske. Satisfiability and model checking for MSO-definable temporal logics are in PSPACE. In *CONCUR'03, Marseille, France*, volume 2761 of *Lecture Notes in Computer Science*, pages 222–236. Springer, 2003.

[GKA+11] V. Ganesh, A. Kiezun, S. Artzi, Ph. Guo, P. Hooimeijer, and M. D. Ernst. HAMPI: A string solver for testing, analysis and vulnerability detection. In *CAV'11*, volume 6806 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2011.

[HL14] M. Hutagalung and M. Lange. Model checking for string problems. In *CSR'14*, volume 8476 of *Lecture Notes in Computer Science*, pages 190–203. Springer, 2014.

[Kus06] D. Kuske. Theories of orders on the set of words. *Theoretical Informatics and Applications*, 40:53–74, 2006.

[KW15] A. Kartzow and Th. Weidner. Model checking constraint LTL over trees. *CoRR*, arXiv:1504.06105, 2015.

[Lia14] T. Liang. *Automated reasoning over string constraints.* PhD thesis, University of Iowa, December 2014.

[LM05] C. Lutz and M. Miličić. A tableau algorithm for description logics with concrete domains and GCIs. In *TABLEAUX'05*, volume 3702 of *Lecture Notes in Computer Science*, pages 201–216. Springer, 2005.

[LRT+14] T. Liang, A. Reynolds, C. Tinelli, C. Barrett, and M. Deters. A DPLL(T) theory solver for a theory of strings and regular expressions. In *CAV'14*, volume 8559 of *Lecture Notes in Computer Science*, pages 646–662, 2014.

[Lut04] C. Lutz. NEXPTIME-complete description logics with concrete domains. *ACM Transactions on Computational Logic*, 5(4):669–705, 2004.

[Mak77] G.S. Makanin. The problem of solvability of equations in a free semigroup (english translation). *Mathematics of USSR-Sorbnik*, 32:147–236, 1977.

[Mar04] N. Markey. Past is for free: On the complexity of verifying linear temporal properties with past. *Acta Informatica*, 40(6–7):431–458, 2004.

[Pap81] Chr. Papadimitriou. On the complexity of integer programming. *Journal of the Association for Computing Machinery*, 28(4):765–768, 1981.

[Pla04] W. Plandowski. Satisfiability of word equations with constants is in PSPACE. *Journal of the Association for Computing Machinery*, 51(3):483–496, 2004.

[Pla06] W. Plandowski. An efficient algorithm for solving word equations. In *STOC'06*, pages 467–476. ACM, 2006.

[Pot91] L. Pottier. Minimal Solutions of Linear Diophantine Systems: Bounds and Algorithms. In *RTA'91*, pages 162–173. Springer, 1991.

[Qui46] W.V. Quine. Concatenation as a basis for arithmetic. *The Journal of Symbolic Logic*, 11(4):105–114, 1946.

[Sav70] W.J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[ST11] L. Segoufin and S. Torunczyk. Automata based verification over linearly ordered data domains. In *STACS'11*, pages 81–92, 2011.

[Var88] M. Vardi. A temporal fixpoint calculus. In *POPL'88*, pages 250–259. ACM, 1988.

[VW94] M. Vardi and P. Wolper. Reasoning about infinite computations. *Information & Computation*, 115:1–37, 1994.

[Wol83] P. Wolper. Temporal logic can be more expressive. *Information & Computation*, 56:72–99, 1983.

[ZZG13] Y. Zheng, X. Zhang, and V. Ganesh. Z3-str: a Z3-based string solver for web application analysis. In *ESEC/FSE'13*, pages 114–124. ACM, 2013.