# Deciding Emptiness for Constraint Automata on Strings with the Prefix and Suffix Order

**Dominik Peteler** ✉
Universität Leipzig, Germany

**Karin Quaas** ✉
Universität Leipzig, Germany

## Abstract

We study constraint automata that accept data languages on finite string values. Each transition of the automaton is labelled with a constraint restricting the string value at the current and the next position of the data word in terms of the prefix and the suffix order. We prove that the emptiness problem for such constraint automata with Büchi acceptance condition is NL-complete. We remark that since the constraints are formed by two partial orders, prefix and suffix, we cannot exploit existing techniques for similar formalisms. Our decision procedure relies on a decidable characterization for those infinite paths in the graph underlying the automaton that can be completed with string values to yield a Büchi-accepting run. Our result is - to the best of our knowledge - the first work in this context that considers both prefix and suffix, and it is a first step into answering an open question posed by Demri and Deters.

**2012 ACM Subject Classification** Theory of computation → Automata over infinite objects

**Keywords and phrases** Data Languages, Strings, Constraints, Prefix, Suffix, Automata, Linear Temporal Logic

**Digital Object Identifier** 10.4230/LIPIcs.MFCS.2022.76

**Funding** *Karin Quaas*: Funded by the Deutsche Forschungsgemeinschaft (DFG), project 406907430.

**Acknowledgements** We would like to thank the anonymous reviewers for their very careful reading and many insightful comments that led to the improvement of this article.

## 1 Introduction

Motivated by applications in formal verification, automated reasoning and databases, logics and automata over *infinite* alphabets are in the focus of active and broad research activities in theoretical computer science. A typical example for logics over infinite alphabets is *constraint linear temporal logic*, CLTL for short [2, 10, 11, 9, 6, 5, 14]. CLTL extends classical LTL with a finite set of variables ranging over the domain of some infinite relational structure like $(\mathbb{N}; =)$ or $(\mathbb{Q}; <)$. Atomic formulas in CLTL are *constraints* in terms of the variables and the relation symbols from the structure; atomic formulas can be combined with Boolean operations and the usual temporal modalities. Models of CLTL formulas are *data words*, that is, infinite sequences of data values coming from the domain of the relational structure. For instance, the CLTL formula $\mathsf{G}(\mathsf{X}x < x)$ over the relational structure $(\mathbb{Z}; <)$ states: "globally, the value of the variable $x$ at the next position is smaller than the value of $x$ at the current position". The data word $4, 3, 2, 1, 0, -1, -2, \ldots$ is a model of this formula. Note that the same formula has no model if instead of $(\mathbb{Z}; <)$ we evaluate the formula over the relational structure $(\mathbb{N}; <)$ – this to illustrate that deciding the satisfiability of a given formula in this logic heavily relies on the considered relational structure.

A natural counterpart to CLTL are *constraint automata* [7, 15, 28, 19]. Like CLTL, constraint automata are parameterized over a relational structure. Transitions are labelled with constraints in terms of the variables of the automaton and the relation symbols from the


47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022).
Editors: Stefan Szeider, Robert Ganian, and Alexandra Silva; Article No. 76; pp. 76:1–76:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

structure; the satisfaction of the constraints along a transition determines the behaviour of the automaton. Constraint automata generalize Büchi automata and accept data languages, that is sets of data words. Following the classical automata-theoretic approach by Vardi and Wolper, one can reduce the satisfiability problem for CLTL to the non-emptiness problem for constraint automata (cf. [29, 14]). We remark that constraint automata are very much related to the well known class of register automata [17, 24, 13, 4, 8].

For CLTL, constraint automata, and other formalisms parameterized over relational structures, a lot of remarkable results concerning satisfiability, model-checking, and the emptiness problem have been achieved, see [14] for a recent survey. This includes results for specific relational structures – for instance, the satisfiability problem for CLTL over $(\mathbb{Z}; <)$ is PSPACE-complete [2, 10] – but there are also noteworthy unifying approaches that capture logics and automata over certain *classes* of, e.g., linear orders [28, 6, 5] or oligomorphic data domains [3].

In contrast to linear orders, relatively little is known about relational structures where the domain equals the set of strings $A^*$ over some fixed (finite or countably infinite) alphabet $A$, and relations defined over $A^*$, like the *prefix order* or the *subsequence order*. While relational structures over linear orders are useful for analysing systems that manipulate counters or constrain real-timed variables, relational structures over strings are interesting for reasoning about systems that manipulate pushdown stacks, queues, or other data structures that involve strings. Reasoning on string variables has a long tradition in theoretical computer science, with roots in algebra and combinatorics on words, and recent developments in the area of string constraint solving (see [1] for a recent survey). Several works concern first-order (FO) logics over finite strings [20, 18, 16, 21]. Thereof, a recent undecidability result [16] for the $\Sigma_1$-fragment of FO logic over $(\Sigma^*; \leq_{\text{sub}}, (=w)_{w\in\Sigma^*})$, where $\Sigma$ is a finite alphabet and $\leq_{\text{sub}}$ denotes the subsequence order over $\Sigma^*$, immediately implies the undecidability of the satisfiability problem for CLTL over that structure. Regarding the relational structure $(A^*; <_p, =, (=w)_{w\in A^*})$, where $<_p$ is the prefix order over $A^*$, we know: the satisfiability problem for constraint LTL is PSPACE-complete (by an interesting reduction to the same problem for $(\mathbb{N}; <, =, (=n)_{n\in\mathbb{N}})$) [9]. The emptiness problem for constraint automata is PSPACE-complete [19]. On the other hand, a unifying, model-theoretic approach for a large family of temporal logics, including ECTL*, which is applicable to linear orders, fails for the prefix order over finite strings [5].

Demri and Deters proposed to study the satisfiability problem for CLTL when evaluated over the structure $(A^*; <_p, <_s, =, (=w)_{w\in A^*})$ with both the prefix and the suffix order [9]. This enables us to express properties like "the beginning of the content of a string is equal to the end of some other string". Using the obvious symmetry between the prefix order and the suffix order, one can conclude that the above mentioned results for the prefix order hold for the relational structure where $<_p$ is *replaced* by $<_s$ [9]. However, the situation changes drastically when both $<_p$ and $<_s$ are in the relational structure. For instance, the FO theory on the prefix order alone is decidable [27], but becomes undecidable for the relational structure containing both prefix and suffix (this follows from the undecidability result for the FO theory for the substring (infix) order [20], and the fact that the substring order is FO-definable using prefix and suffix). For finite strings over a finite alphabet, it has been remarked in [9] that the $\Sigma_1$-fragment of FO logics is decidable, using an algorithm based on the word equation approach by Makanin [22, 26]. It is thus far from clear whether satisfiability for CLTL, or, equivalently, the emptiness problem for constraint automata, is decidable or not. The techniques used in other works for, e.g. the prefix order alone, or linear orders, turn out to be not applicable at all.

In this paper, we prove that the emptiness problem for constraint automata over prefix and suffix is decidable in NL if the automaton uses only a single variable. This is a standard restriction, comparable to one-counter automata [12] or single-clock timed automata [25]. Our decision procedure relies on a reduction to reachability queries on the finite graph underlying the automaton, and it applies to finite strings over both finite and countably infinite alphabets. We may also test whether the string equals the empty string (similar to a zero test in one-counter automata). We further obtain NL-completeness for the emptiness problem for single-register automata over this relational structure. Last but not least, our result implies PSPACE-completeness for the satisfiability of CLTL for the case that the formulas in CLTL only use a single variable.

We leave open the decidability status for the case where equality with arbitrary finite strings over $A$ and/or constraints involving more than one variable are allowed, that is, by now we cannot fully answer the question raised by Demri and Deters. We remark that both extensions may be harmful: for instance, while emptiness is decidable for one-counter automata [12], it is undecidable for two-counter automata [23]; while the $\Sigma_1$-fragment of FO logic for finite strings over a finite alphabet with the subsequence order *without constants* is decidable [20], it is undecidable as soon as we allow constants in the relational structure [16].

## 2 Preliminaries

A *relational signature* $\sigma = \{R_1, R_2, \dots\}$ is a countable set of relation symbols. Each symbol $R_i$ is associated with some non-negative arity $k_i$. A *relational structure over* $\sigma$, or $\sigma$*-structure* for short, is a tuple $\mathcal{D} = (D; R_1^{\mathcal{D}}, R_2^{\mathcal{D}}, \dots)$, where $D$ is the domain of the structure, and $R_i^{\mathcal{D}} \subseteq D^{k_i}$ is the interpretation of the symbol $R_i$ in $D$. We will often omit the symbol $\mathcal{D}$ in $R_i^{\mathcal{D}}$ and simply write $R_i$ instead.

We use $\Sigma$ to denote a finite alphabet, $\mathbb{N}$ as a countably infinite alphabet, and $A$ as finite or countably infinite alphabet. We use $A^*$ to denote the set of finite strings over $A$. The symbol $\varepsilon$ denotes the empty string, and we use $A^+$ to denote the set $A^* \setminus \{\varepsilon\}$ of non-empty strings over $A$. Given $u, v \in A^*$, we say that $u$ is a *strict prefix* (*strict suffix*, respectively) of $v$, written $u <_p v$ ($u <_s v$, respectively), if $v = u \cdot u'$ ($v = u' \cdot u$, respectively) for some $u' \in A^+$. We say that $u$ and $v$ are *incomparable with respect to the prefix order*, written $u \perp_p v$, if $u = w \cdot a \cdot u'$ and $v = w \cdot b \cdot v'$ for some $w \in A^*$, $a, b \in A$ such that $a \neq b$, and $u', v' \in A^*$. *Incomparability with respect to the suffix order*, written $u \perp_s v$, is defined analogously.

Let $\sigma^{ps}$ be the signature consisting of the binary symbols $<_p$, $<_s$, and $=$. In this paper, we are interested in the $\sigma^{ps}$-structures $(\Sigma^*; <_p, <_s, =)$ and $(\mathbb{N}^*; <_p, <_s, =)$, where, in both structures, $<_p$ and $<_s$ are interpreted as the prefix and the suffix order over the set of strings over $\Sigma$ and $\mathbb{N}$, respectively, and $=$ is interpreted as the identity. If the context is clear, we may write $\Sigma^*$ and $\mathbb{N}^*$ to denote the respective structures, and $A^*$ to denote any of these structures.

*Constraint automata* are generalizations of Büchi automata that are parameterized by $\sigma$-structures, where $\sigma$ is a relational signature. The transitions are labelled with Boolean combinations of atomic formulas, called *constraints*, in terms of the relations of the $\sigma$-structure. A constraint automaton processes *data words*. A data word is a finite or infinite sequence $d_1, d_2, d_3 \dots$, where $d_i \in D$ is a data value from the domain of the $\sigma$-structure. A transition of a constraint automaton can be taken if the current and the next data value of the processed data word satisfy the constraint labelling the transition.

In the following, we assume that constraint automata are parameterized by the $\sigma^{ps}$-structures $\Sigma^*$ or $\mathbb{N}^*$, and the transitions are labelled by Boolean combinations of atomic formulas of the form $z \bowtie z'$, where $z, z' \in \{x, y\}$ and $\bowtie \in \sigma^{ps}$. Intuitively, $x$ stands for

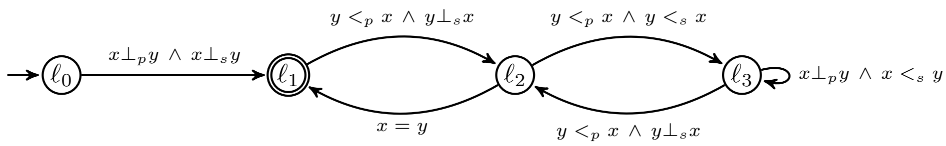


**Figure 1** Example of a constraint automaton over $A^*$

the string at the current position, and $y$ stands for the string at the next position of the processed data word. For the sake of readability we will restrict the labels of the transitions to be *maximally consistent*. Formally, we define $\Psi$ to be the set of formulas $\psi(x, y)$ of the following form:

$$\begin{array}{cccc} x = y, & x <_p y \wedge x <_s y, & x <_p y \wedge x \perp_s y, & x \perp_p y \wedge x <_s y \\ y <_p x \wedge y <_s x, & y <_p x \wedge y \perp_s x, & y \perp_p x \wedge y <_s x, & x \perp_p y \wedge x \perp_s y. \end{array}$$

Each of these formulas is called *constraint*. Constraints that contain the formula $y <_p x$ or $y <_s x$ are called *reducing* as they reduce the length of the string at the next position with regard to the string at the current position. All other constraints except for $x = y$ are called *generous*, because they allow for infinitely many choices of the string value at the next position. The satisfaction relation $\models$ is defined in the obvious way. For instance, if $\psi$ is of the form $x <_p y \wedge x <_s y$, then we have $\mathbb{N}^* \models \psi(0, 01210)$ and $\Sigma^* \not\models \psi(a, abab)$.

A *constraint automaton* over a $\sigma^{ps}$-structure $A^*$ is a tuple $\mathcal{A} = (\mathcal{L}, \ell_{\text{in}}, \mathcal{L}_{\text{acc}}, E)$, where

- $\mathcal{L}$ is a finite set of locations (control states);
- $\ell_{\text{in}} \in \mathcal{L}$ is the initial location;
- $\mathcal{L}_{\text{acc}} \subseteq \mathcal{L}$ is the set of accepting locations; and
- $E \subseteq \mathcal{L} \times \Psi \times \mathcal{L}$ is the set of edges.

A *path* of $\mathcal{A}$ is a finite or infinite sequence $\ell_0, \ell_1, \ell_2, \ldots$ of locations satisfying, for all $i \geq 1$, $(\ell_{i-1}, \psi_i, \ell_i) \in E$ for some constraint $\psi_i \in \Psi$. We may sometimes also write $\ell_0 \xrightarrow{\psi_1} \ell_1 \xrightarrow{\psi_2} \ell_2, \ldots$ to indicate the precise edges that are used. A finite path $\ell_0 \xrightarrow{\psi_1} \ell_1 \xrightarrow{\psi_2} \ell_2 \ldots \xrightarrow{\psi_n} \ell_n$ is *stable* if $\psi_i$ is of the form $x = y$ for all $1 \leq i \leq n$; it is *generous* if there exists some $1 \leq i \leq n$ such that $\psi_i$ is generous. A path as above is a *cycle starting in* $\ell_0$ if $\ell_0 = \ell_n$.

A *state* of $\mathcal{A}$ is a pair $(\ell, w)$, where $\ell \in \mathcal{L}$ and $w \in A^*$ is the current value of the string variable. We postulate a labelled transition relation $\rightarrow$ over the set $\mathcal{L} \times A^*$ of states of $\mathcal{A}$, as follows: $(\ell, w) \rightarrow (\ell, w')$ if there exists a transition $(\ell, \psi(x, y), \ell') \in E$ such that $A^* \models \psi(w, w')$. A *run* of $\mathcal{A}$ is a finite or infinite sequence of transitions of $\mathcal{A}$. A run $(\ell_0, w_0) \rightarrow (\ell_1, w_1) \rightarrow (\ell_2, w_2) \ldots$ is *initialized* if $\ell_0 = \ell_{\text{in}}$. A run is *Büchi-accepting* if it is initialized and it contains infinitely many locations in $\mathcal{L}_{\text{acc}}$. We define the language of $\mathcal{A}$ by $L(\mathcal{A}) = \{(w_0 w_1 w_2 \cdots \mid (\ell_0, w_0) \rightarrow (\ell_1, w_1) \rightarrow (\ell_2, w_2) \ldots$ is a Büchi-accepting run of $\mathcal{A}\}$.

For an example, consider the constraint automaton $\mathcal{A} = (\{\ell_0, \ell_1, \ell_2, \ell_3\}, \ell_0, E, \{\ell_1\})$ over $\mathbb{N}^*$, where $E$ is as depicted in Figure 1. A finite initialized run of this automaton is $(\ell_0, 20) \rightarrow (\ell_1, 346345346343) \rightarrow (\ell_2, 34634534634) \rightarrow (\ell_3, 34634) \rightarrow (\ell_2, 346)$ (cf. Example 6).

The *emptiness problem* for constraint automata is to decide, given a constraint automaton $\mathcal{A}$, whether $L(\mathcal{A}) = \emptyset$. In section 4, we prove that this problem is NL-complete.

## 3 Rewriting Operation

For deciding the emptiness problem, we will prove the existence of string values that satisfy the constraints that occur in a given path. During the process of defining such string values, we will need to change already defined string values using a *rewriting operation*. In this section, we define this operation and prove some important properties.

Recall that $A$ denotes a finite or countably infinite alphabet. Given $w \in A^*$ and two non-empty strings $u, u' \in A^+$, we define the *left-to-right rewriting operation of $u$ to $u'$ in $w$*, denoted by $w[u \leftarrow u']_\triangleright$, to be the string that is obtained from $w$ by replacing, from left to right, every occurrence of $u$ in $w$ by $u'$. Formally, assume $w = a_1 a_2 \dots a_n$ and define, recursively, $w[u \leftarrow u']_\triangleright := w$ if $a_i a_{i+1} \dots a_{i+|u|-1} \neq u$ for all $1 \leq i \leq n$ (that is, $u$ does not occur in $w$), $w[u \leftarrow u']_\triangleright := a_1 \dots a_{i-1} \cdot u' \cdot ((a_{i+|u|} \dots a_n)[u \leftarrow u']_\triangleright)$ if $1 \leq i \leq n$ is the minimal index such that $a_i a_{i+1} \dots a_{i+|u|-1} = u$. Note that if $u$ occurs in $a_1 \dots a_{i-1} \cdot u'$, then $u$ is not replaced in any further steps of the recursive definition. For instance, $1100210[10 \leftarrow 1]_\triangleright = 11021$. We define completely analogously the right-to-left version of this operation, that is, $w[u \leftarrow v]_\triangleleft := w$ if $u$ does not occur in $w$, and $w[u \leftarrow v]_\triangleleft := ((a_1 \dots a_{i-1})[u \leftarrow u']_\triangleright) \cdot u' \cdot a_{i+|u|} \dots a_n$, if $i \geq 1$ is the maximal index such that $a_i \dots a_{i+|u|-1} = u$. Note that $w[u \leftarrow v]_\triangleright$ may be different from $w[u \leftarrow v]_\triangleleft$; for instance, $w = 111$, $u = 11$ and $v = 0$ yields $w[u \leftarrow v]_\triangleright = 01$ and $w[u \leftarrow v]_\triangleleft = 10$. It is easy to see that this is the case if there exist two overlapping occurrences of $u$ in $w$. Formally, we say that *$u$ is overlapping in $w$* if there exist $1 \leq i < j < i + |u| \leq n$ such that $a_i a_{i+1} \dots a_{i+|u|-1} = a_j a_{j+1} \dots a_{j+|u|-1} = u$. The proof of the following lemma is simple.

▶ **Lemma 1.** *For all $w \in A^*$ and $u, u' \in A^+$, if $u$ is not overlapping in $w$, then we have $w[u \leftarrow u']_\triangleright = w[u \leftarrow u']_\triangleleft$.*

In Subsection 4.1 we will guarantee that the rewriting operation is only applied to strings $w$ and $u$ such that $u$ is not overlapping in $w$, so that the left and right versions of rewriting yield the same string. The reason why we still define both the left and right version of rewriting is that certain properties of the prefix order – stated in the next lemma – can be proved very conveniently using the left-to-right rewriting operation, and the same properties can be proved symmetrically for the suffix order using the right-to-left rewriting operation (Lemma 3).

▶ **Lemma 2.** *For all $u, u' \in A^+$ with $u <_p u'$, for all $w, w' \in A^*$, and for all $\bowtie \in \{=, <_p, \perp_p\}$ we have*

$$w \bowtie w' \iff w[u \leftarrow u']_\triangleright \bowtie w'[u \leftarrow u']_\triangleright.$$

**Proof.** Let $u, u' \in A^+$ be such that $u <_p u'$, that is, there exists some $u'' \in A^+$ such that $u' = u \cdot u''$. Let $w, w' \in A^*$ be of the form $w = a_1 a_2 \dots a_m$ and $w' = a'_1 a'_2 \dots a'_n$. Let $N$ and $N'$, respectively, be the number of (non-overlapping, from left to right) occurrences of $u$ in $w$ and $w'$, respectively. The proof is by induction on the sum $i := N + N'$. For the induction base, assume $i = 0$. But then $w[u \leftarrow u']_\triangleright = w$ and $w'[u \leftarrow u']_\triangleright = w'$, so that the claim clearly holds. For the induction step, suppose that the claim holds for all $0 \leq j < i$. We prove the claim for $i$. We distinguish three cases:

1. $N = i$ and $N' = 0$. By $N = i > 0$, $w$ contains $u$. Since $u <_p u'$, also $w[u \leftarrow u']_\triangleright$ contains $u$. By $N' = 0$, $w'$ does not contain $u$ and $w'[u \leftarrow u']_\triangleright = w'$, so that $w'[u \leftarrow u']_\triangleright$ does not contain $u$ either. Using this, it is easy to see that none of the following cases can hold: $w = w'$, $w <_p w'$, $w[u \leftarrow u']_\triangleright = w'[u \leftarrow u']_\triangleright$ and $w[u \leftarrow u']_\triangleright <_p w'[u \leftarrow u']_\triangleright$. So let us prove $w \perp_p w' \iff w[u \leftarrow u']_\triangleright \perp_p w'$. For this suppose $w[u \leftarrow u']_\triangleright$ is of the form

$b_1 b_2 \dots b_q$. Let $1 \le d \le e \le m$ be such that $a_d \dots a_e = u$ is the *first* occurrence of $u$ in $w$. By $u <_p u'$, $b_d \dots b_e = u$ is also the first occurrence of $u$ in $w[u \leftarrow u']_\triangleright$, and hence $a_1 \dots a_e = b_1 \dots b_e$. We hence obtain

$$\begin{aligned} & w \perp_p w' \\ \iff \quad & \exists k \le e \text{ such that } a_1 \dots a_{k-1} = a'_1 \dots a'_{k-1} \text{ and } a_k \ne a'_k \\ \iff \quad & \exists k \le e \text{ such that } b_1 \dots b_{k-1} = a'_1 \dots a'_{k-1} \text{ and } b_k \ne a'_k \\ \iff \quad & w[u \leftarrow u']_\triangleright \perp_p w' \end{aligned}$$

where $k \le e$ holds by the fact that $u$ is not contained in $w'$.

2. $N = 0$ and $N' = i$. By $N = 0$, $w$ does not contain $u$ and hence neither does $w[u \leftarrow u']_\triangleright$ as $w[u \leftarrow u']_\triangleright = w$. By $N' = i > 0$, $w'$ contains $u$. Using this, it is easy to see that the following two cases cannot hold: $w = w'$ and $w[u \leftarrow u']_\triangleright = w'[u \leftarrow u']_\triangleright$. The proof for $w \perp_p w' \iff w \perp_p w'[u \leftarrow u']_\triangleright$ is symmetric to the proof in the previous case. So let us prove $w <_p w' \iff w <_p w'[u \leftarrow u']_\triangleright$. For this let $w'[u \leftarrow u']_\triangleright$ be of the form $b'_1 \dots b'_q$. Let $1 \le d \le e \le n$ be such that $a'_d \dots a'_e = u$ is the *first* occurrence of $u$ in $w'$. By $u <_p u'$, $b'_d \dots b'_e = u$ is also the first occurrence of $u$ in $w'[u \leftarrow u']_\triangleright$, and hence $a'_1 \dots a'_e = b'_1 \dots b'_e$. We hence obtain

$$\begin{aligned} & w <_p w' \\ \iff \quad & a_1 \dots a_m = a'_1 \dots a'_m \\ \iff \quad & a_1 \dots a_m = b'_1 \dots b'_m \\ \iff \quad & w <_p w'[u \leftarrow u']_\triangleright \end{aligned}$$

where the second equivalence holds because $m < e$ (as otherwise $u$ would be contained in $w$).

3. $N > 0$ and $N' > 0$. Let $1 \le d \le e \le m$ be such that $a_d \dots a_e = u$ is the first occurrence of $u$ in $w$, and similarly, let $1 \le d' \le e' \le n$ be such that $a'_{d'} \dots a'_{e'} = u$ is the first occurrence of $u$ in $w'$. In other words, we can write

$$w = a_1 \dots a_{d-1} \cdot u \cdot v \quad \text{and} \quad w' = a'_1 \dots a'_{d'-1} \cdot u \cdot v',$$

where $v = a_e \dots a_m$ and $v' = a'_{e'} \dots a'_n$. By definition and $u' = u \cdot u''$, we have

$$w[u \leftarrow u']_\triangleright = a_1 \dots a_{d-1} \cdot u \cdot u'' \cdot (v[u \leftarrow u']_\triangleright)$$

and

$$w'[u \leftarrow u']_\triangleright = a'_1 \dots a'_{d'-1} \cdot u \cdot u'' \cdot (v'[u \leftarrow u']_\triangleright).$$

We distinguish four cases:

a. $a_1 \dots a_{d-1} \cdot u = a'_1 \dots a'_{d'-1} \cdot u$. This implies

$$w \bowtie w' \iff v \bowtie v' \quad \text{and} \quad w[u \leftarrow u']_\triangleright \bowtie w'[u \leftarrow u']_\triangleright \iff v[u \leftarrow u']_\triangleright \bowtie v'[u \leftarrow u']_\triangleright$$

for $\bowtie \in \{<_p, =, \perp_p\}$. The sum $M + M'$ of the occurrences of $u$ in $v$ and $v'$ must be strictly smaller than $i$. By induction hypothesis,

$$v \bowtie v' \iff v[u \leftarrow u']_\triangleright \bowtie v'[u \leftarrow u']_\triangleright.$$

Hence the result.

b. $a_1 \dots a_{d-1} \cdot u <_p a'_1 \dots a'_{d'-1} \cdot u$. This contradicts the minimality of $d'$, and hence this case cannot happen.

**c.** $a'_1 \dots a'_{d'-1} \cdot u <_p a_1 \dots a_{d-1} \cdot u$. This contradicts the minimality of $d$, and hence this case cannot happen.

**d.** $a_1 \dots a_{d-1} \cdot u \perp_p a'_1 \dots a'_{d'-1} \cdot u$. Hence there exists some $1 \leq j \leq \min(d-1, d'-1)$ such that $a_1 \dots a_{j-1} = a'_1 \dots a'_{j-1}$ and $a_j \neq a'_j$. This immediately implies $w \perp_p w'$ and also $w[u \leftarrow u']_\triangleright \perp_p w'[u \leftarrow u']_\triangleright$, hence the result. ◀

A proof for the following lemma can be done symmetrically to the proof of Lemma 2.

▶ **Lemma 3.** *For all $u, u' \in A^+$ and $w, w' \in A^*$, if $u <_s u'$, then $w \bowtie w'$ if, and only if, $w[u \leftarrow u']_\triangleleft \bowtie w'[u \leftarrow u']_\triangleleft$ for all $\bowtie \in \{=, <_s, \perp_s\}$.*

The following lemma will be crucial in Subsection 4.1.

▶ **Lemma 4.** *For all $v, w \in A^*$ and $u, u' \in A^+$, if $u$ is not overlapping in $v$, $u$ is not overlapping in $w$, $u <_p u'$, and $u <_s u'$, then $A^* \models \psi(v, w)$ if, and only if, $A^* \models \psi(v[u \leftarrow u']_\triangleright, w[u \leftarrow u']_\triangleright)$ for all $\psi \in \Psi$.*

**Proof.** The proof is an easy case distinction depending on the form of $\psi$. We give the proof for $\psi$ being of the form $x <_p y \wedge x \perp_s y$.

$$\begin{aligned}
& \mathbb{N}^* \models \psi(v, w) \\
\iff & v <_p w \text{ and } v \perp_s w \quad \text{(by definition)} \\
\iff & v[u \leftarrow u']_\triangleright <_p w[u \leftarrow u']_\triangleright \text{ and } v \perp_s w \quad \text{(by Lemma 2)} \\
\iff & v[u \leftarrow u']_\triangleright <_p w[u \leftarrow u']_\triangleright \text{ and } v[u \leftarrow u']_\triangleleft \perp_s w[u \leftarrow u']_\triangleleft \quad \text{(by Lemma 3)} \\
\iff & v[u \leftarrow u']_\triangleright <_p w[u \leftarrow u']_\triangleright \text{ and } v[u \leftarrow u']_\triangleright \perp_s w[u \leftarrow u']_\triangleright \quad \text{(by Lemma 1)} \\
\iff & \mathbb{N}^* \models \psi(v[u \leftarrow u']_\triangleright, w[u \leftarrow u']_\triangleright) \quad \text{(by definition)}
\end{aligned}$$

The proofs for the other cases are completely analogous. ◀

## 4 Deciding Emptiness for Constraint Automata over $\mathbb{N}^*$

In this section, we solve the emptiness problem for constraint automata over $\mathbb{N}^*$. We start in the next subsection with presenting an algorithm that returns for every *finite* sequence of constraints $\psi_1, \dots, \psi_n$ a sequence of string values $w_0, w_1, \dots, w_n$ such that $\mathbb{N}^* \models \psi_i(w_{i-1}, w_i)$ for all $1 \leq i \leq n$. The sequence $\psi_1, \dots, \psi_n$ may correspond to the sequence of constraints occurring in a finite path $\pi = \ell_0 \xrightarrow{\psi_1} \dots \xrightarrow{\psi_n} \ell_n$ of a constraint automaton $\mathcal{A}$, and by constructing string values $w_0, w_1, \dots, w_n$ we actually prove that $\pi$ can be completed to a finite run $(\ell_0, w_0) \xrightarrow{\psi_1} \dots \xrightarrow{\psi_n} (\ell_n, w_n)$ of $\mathcal{A}$. This already implies NL-membership of the reachability problem for constraint automata (Corollary 8).

We remark that for *infinite* sequences of constraints $\psi_1, \psi_2, \dots$ it is not the case that we can always find string values $w_0, w_1, \dots$ satisfying $\mathbb{N}^* \models \psi_i(w_{i-1}, w_i)$. Consider for instance the sequence $(x \perp_p y \wedge x \perp_s y)\,(y <_p x \wedge y \perp_s x)^\omega$ (cf. Figure 1). The constraint $y <_p x \wedge y \perp_s x$ is reducing and requires the strings in the $\omega$-sequence to become shorter infinitely often, which is impossible. In Subsection 4.2 we give a characterization for when infinite paths can be extended to infinite runs, which, together with the results obtained before, yields a decision procedure for the emptiness problem.

For the rest of this section, let $\mathcal{A} = (\mathcal{L}, \ell_{\text{in}}, \mathcal{L}_{\text{acc}}, E)$ be a constraint automaton over $\mathbb{N}^*$.

## 4.1 Extending Finite Paths to Finite Runs

The main part of this subsection is dedicated to prove the following result.

▶ **Proposition 5.** *For every sequence $\psi_1, \ldots, \psi_n$ of constraints in $\Psi$ and non-empty string $w_{\text{in}} \in \mathbb{N}^+$, we can define non-empty strings $w_0, w_1, \ldots, w_n \in \mathbb{N}^+$ such that $\mathbb{N}^* \models \psi_i(w_{i-1}, w_i)$ for all $1 \leq i \leq n$. Moreover, if $\psi_1$ is generous, then $w_0 = w_{\text{in}}$.*

Let $\psi_1, \ldots, \psi_n$ be a finite sequence of constraints in $\Psi$, and let $w_{\text{in}}$ be an initial non-empty string value. The idea is to construct, one after the other, string values that satisfy the constraints. During the process, formerly defined string values may need to be rewritten using the left-to-right-rewriting operation defined in Section 3 (so that $w_0$ may not be equal to the input string $w_{\text{in}}$). We take advantage of the fact that we have an unbounded supply of "fresh" letters as we operate on the infinite alphabet $\mathbb{N}$: we can assign string values in such a way that constraints that are satisfied *before* a rewriting still hold true *after* a rewriting. Given a string $w \in \mathbb{N}^*$, we let $\max(w)$ be the maximal number occurring as a letter in $w$ if $w \neq \varepsilon$ and $\max(w) = 0$ otherwise.

For $1 \leq i \leq n$, suppose we have already defined string values $w_0^{i-1}, w_1^{i-1}, \ldots, w_{i-1}^{i-1}$ such that $\mathbb{N}^* \models \psi_j(w_{j-1}^{i-1}, w_j^{i-1})$ for all $1 \leq j < i$, where $w_0^0 = w_{\text{in}}$. Define $M_i = \max\{w_0^{i-1}, \ldots, w_{i-1}^{i-1}\} + 1$, so that $M_i$ is a "fresh" letter not occurring in any of the already defined string values. Depending on the form of $\psi_i$, we define $w_0^i, w_1^i, \ldots, w_i^i$ such that $\mathbb{N}^* \models \psi_j(w_{j-1}^i, w_j^i)$ for all $1 \leq j \leq i$. We consider the following cases:

1. $\psi_i$ is of the form $x = y$. Define $w_i^i = w_{i-1}^{i-1}$, and $w_j^i = w_j^{i-1}$ for all $0 \leq j < i$.
2. $\psi_i$ is of the form $x <_p y \wedge x <_s y$. Define $w_i^i = w_{i-1}^{i-1} \cdot M_i \cdot w_{i-1}^{i-1}$, and $w_j^i = w_j^{i-1}$ for all $0 \leq j < i$.
3. $\psi_i$ is of the form $x <_p y \wedge x \perp_s y$. Define $w_i^i = w_{i-1}^{i-1} \cdot M_i$, and $w_j^i = w_j^{i-1}$ for all $0 \leq j < i$.
4. $\psi_i$ is of the form $x \perp_p y \wedge x <_s y$. Define $w_i^i = M_i \cdot w_{i-1}^{i-1}$, and $w_j^i = w_j^{i-1}$ for all $0 \leq j < i$.
5. $\psi_i$ is of the form $y <_p x \wedge y <_s x$. Define $w_i^i = w_{i-1}^{i-1}$, and for all $0 \leq j < i$ define $w_j^i = w_j^{i-1}[w_{i-1}^{i-1} \leftarrow w_{i-1}^{i-1} \cdot M_i \cdot w_{i-1}^{i-1}]_\triangleright$.
6. $\psi_i$ is of the form $y <_p x \wedge y \perp_s x$. Define $w_i^i = w_{i-1}^{i-1} \cdot M_i$, and for all $0 \leq j < i$ define $w_j^i = w_j^{i-1}[w_{i-1}^{i-1} \leftarrow w_{i-1}^{i-1} \cdot M_i \cdot w_{i-1}^{i-1}]_\triangleright$.
7. $\psi_i$ is of the form $y \perp_p x \wedge y <_s x$. Define $w_i^i = M_i \cdot w_{i-1}^{i-1}$, and for all $0 \leq j < i$ define $w_j^i = w_j^{i-1}[w_{i-1}^{i-1} \leftarrow w_{i-1}^{i-1} \cdot M_i \cdot w_{i-1}^{i-1}]_\triangleright$.
8. $\psi_i$ is of the form $x \perp_p y \wedge x \perp_s y$. Define $w_i^i = M_i$ and $w_j^i = w_j^{i-1}$ for all $0 \leq j < i$.

▶ **Example 6.** Let us illustrate the construction with the sequence $\psi_1, \psi_2, \psi_3$ and $w_{\text{in}} = 3$, where $\psi_1 = \psi_3 = (y <_p x \wedge y \perp_s x)$, and $\psi_2 = (y <_p x \wedge y <_s x)$. For $i = 1$, we are in case **6**. We have $M_1 = 4$ and obtain $w_1^1 = w_0^0 \cdot M_1 = 34$ and $w_0^1 = w_0^0[3 \leftarrow 343]_\triangleright = 343$. Clearly $\mathbb{N}^* \models \psi_1(w_0^1, w_1^1)$. For $i = 2$, we are in case **5** and define $w_2^2 = w_1^1 = 34$. We further rewrite $w_1^2 = w_1^1[34 \leftarrow 34534]_\triangleright = 34534$, and $w_0^2 = w_0^1[34 \leftarrow 34534]_\triangleright = 345343$. So even after rewriting, we have $\mathbb{N}^* \models \psi_i(w_{i-1}^2, w_i^2)$ for $i = 1, 2$. For $i = 3$, we are again in case **6**. We obtain $w_3^3 = 346$; $w_2^3$, $w_1^3$ and $w_0^3$, respectively, are rewritten to 34634, 34634534634 and 346345346343, respectively. All constraints are indeed satisfied.

Let us state an important property of the construction, which will be key for the correctness of the construction.

▶ **Invariant 7.** *For every $0 \leq i \leq n$ and every $0 \leq j \leq i$, $w_i^i$ is not overlapping in $w_j^i$.*

**Proof.** The proof is by induction on $i$. The induction base, $i = 0$, is trivial. So assume that the claim holds for all $0 \leq k < i$. We prove it for $i$. We consider different cases, based on the form of $\psi_i$. Let $0 \leq j < i$ (the case $j = i$ is trivial).

1. Suppose we are in cases **1**, **2**, **3**, **4**, or **8**, that is, $w_j^i = w_j^{i-1}$ (no rewriting happens). In case **1**, $w_i^i = w_{i-1}^{i-1}$, we can apply the induction hypothesis to obtain the result. In the remaining four cases, the string $w_i^i$ contains the letter $M_i$, which, by definition, does not occur in $w_j^{i-1}$. Hence $w_i^i$ cannot occur at all in $w_j^{i-1}$.
2. Suppose we are in cases **5**, **6**, or **7**, that is, $w_j^i = w_j^{i-1}[w_{i-1}^{i-1} \leftarrow w_{i-1}^{i-1} \cdot M_i \cdot w_{i-1}^{i-1}]$ (rewriting happens). By induction hypothesis, $w_{i-1}^{i-1}$ is not overlapping in $w_j^{i-1}$. If $N$ is the number of occurrences of $w_{i-1}^{i-1}$ in $w_j^{i-1}$, we can hence write

$$w_j^{i-1} = u_0 \cdot w_{i-1}^{i-1} \cdot u_1 \cdot w_{i-1}^{i-1} \cdot u_2 \dots u_{N-1} \cdot w_{i-1}^{i-1} \cdot u_N$$

for some $u_0, \dots, u_N \in \mathbb{N}^*$. By definition,

$$w_j^i = u_0 \cdot w_{i-1}^{i-1} \cdot M_i \cdot w_{i-1}^{i-1} \cdot u_1 \cdot w_{i-1}^{i-1} \cdot M_i \cdot w_{i-1}^{i-1} \cdot u_2 \dots u_{N-1} \cdot w_{i-1}^{i-1} \cdot M_i \cdot w_{i-1}^{i-1} \cdot u_N.$$

In case **5**, $w_i^i = w_{i-1}^{i-1}$, so that $w_i^i$ does not contain $M_i$ by definition. The only way for $w_i^i$ to be overlapping in $w_j^i$ is in $u_0 \cdot w_{i-1}^{i-1}$, in $w_{i-1}^{i-1} \cdot u_k \cdot w_{i-1}^{i-1}$ for some $1 \leq k < N$, or in $w_{i-1}^{i-1} \cdot u_N$. But this would contradict that $w_{i-1}^{i-1}$ is not overlapping in $w_j^{i-1}$. In case **6**, $w_i^i = w_{i-1}^{i-1} \cdot M_i$. By definition, $w_j^{i-1}$ does not contain $M_i$. Hence the only way for $w_i^i$ to be contained at all in $w_j^i$ is so that no overlap can occur. The reasoning for case **7**, where $w_i^i = M_i \cdot w_{i-1}^{i-1}$, is analogous. ◀

Let us finally prove the correctness of the construction, that is, for all $1 \leq i \leq n$, we have $\mathbb{N}^* \models \psi_j(w_{j-1}^i, w_j^i)$ for all $1 \leq j \leq i$. The proof is by induction on $i$. For the base case $i = 1$ observe that $w_0^1$ and $w_1^1$ are defined such that $\mathbb{N}^* \models \psi_1(w_0^1, w_1^1)$. So suppose that the claim holds for all $1 \leq k < i$. We prove it for $i$. For $j = i$, it is again easy to see that $\mathbb{N}^* \models \psi_i(w_{i-1}^i, w_i^i)$. So let $1 \leq j < i$. By induction hypothesis, we have $\mathbb{N}^* \models \psi_j(w_{j-1}^{i-1}, w_j^{i-1})$. Depending on the form of $\psi_i$, we either have

- $w_{j-1}^i = w_{j-1}^{i-1}$ and $w_j^i = w_j^{i-1}$, so that we have $\mathbb{N}^* \models \psi_j(w_{j-1}^i, w_j^i)$; or
- $w_{j-1}^i = w_{j-1}^{i-1}[w_{i-1}^{i-1} \leftarrow w_{i-1}^{i-1} \cdot M_i \cdot w_{i-1}^{i-1}]_\triangleright$ and $w_j^i = w_j^{i-1}[w_{i-1}^{i-1} \leftarrow w_{i-1}^{i-1} \cdot M_i \cdot w_{i-1}^{i-1}]_\triangleright$. By Invariant 7, $w_{i-1}^{i-1}$ is not overlapping in $w_{j-1}^{i-1}$, $w_{i-1}^{i-1}$ is not overlapping in $w_j^{i-1}$, and $w_{i-1}^{i-1} <_p w_{i-1}^{i-1} \cdot M_i \cdot w_{i-1}^{i-1}$, and $w_{i-1}^{i-1} <_s w_{i-1}^{i-1} \cdot M_i \cdot w_{i-1}^{i-1}$. By Lemma 4 we obtain $\mathbb{N}^* \models \psi_j(w_{j-1}^i, w_j^i)$.

Setting $w_i = w_i^n$ for all $0 \leq i \leq n$, we are done with the proof of the first claim of Proposition 5.

Let us prove the second claim and suppose that $\psi_1$ is generous. We prove below that for all $1 \leq i \leq n$, $w_i^i$ contains some letter not occurring in $w_0^i$. Note that this implies $w_0^0 = w_0^1 = \dots = w_0^n$. The proof is by induction on $i$. For the induction base, set $i = 1$. Since $\psi_1$ is generous, we are in one of the cases **2**, **3**, **4**, or **8**. Here, $w_1^1$ contains $M_1$, which, by definition, is not occurring in $w_0^0$, and $w_0^1 = w_0^0$. Hence $w_1^1$ contains a letter not occurring in $w_0^1$. For the induction step, suppose the claim holds for all $1 \leq j < i$. We prove it for $i$. Note that depending on the form of $\psi_i$, $w_i^i$ is defined as $w_{i-1}^{i-1}$, a fresh letter $M_i$, or a composition of these two. For $w_0^i$, we either have $w_0^i = w_0^{i-1}$, in which case the induction hypothesis and/or freshness of $M_i$ immediately establishes the claim, or we have $w_0^i = w_0^{i-1}[w_{i-1}^{i-1} \leftarrow w_{i-1}^{i-1} \cdot M_i \cdot w_{i-1}^{i-1}]$. But by induction hypothesis, $w_{i-1}^{i-1}$ contains some letter not occurring in $w_0^{i-1}$, so that $w_{i-1}^{i-1}$ cannot occur in $w_0^{i-1}$ and thus $w_0^{i-1}[w_{i-1}^{i-1} \leftarrow w_{i-1}^{i-1} \cdot M_i \cdot w_{i-1}^{i-1}] = w_0^{i-1}$. Hence $w_i^i$ contains some letter not occurring in $w_0^i$. This finishes the proof of Proposition 5.

The *(control-state) reachability problem for constraint automata* is the problem to decide, given a constraint automaton $\mathcal{A} = (\mathcal{L}, \ell_{\text{in}}, \mathcal{L}_{\text{acc}}, E)$ over $\mathbb{N}^*$ and some target location $\ell \in \mathcal{L}$, whether there exists a run from $(\ell_{\text{in}}, w_0)$ to $(\ell, w)$, for some $w_0, w \in \mathbb{N}^*$.

▶ **Corollary 8.** *The reachability problem for constraint automata is NL-complete.*

**Proof.** For the upper bound, it suffices to decide whether there exists a path from $\ell_{\text{in}}$ to $\ell$, which can be done in NL. If no such path exists, then there exists no run from $(\ell_{\text{in}}, w_0)$ to $(\ell, w)$ for some $w_0, w \in \mathbb{N}^*$. If such a path, say $\ell_0 \xrightarrow{\psi_1} \dots \xrightarrow{\psi_n} \ell_n$, exists, we use Proposition 5 with $\psi_1, \dots, \psi_n$ and some $w_{\text{in}} \in \mathbb{N}^+$ to obtain $w_0, w_1, \dots, w_n \in \mathbb{N}^+$ such that $\mathbb{N}^* \models \psi_i(w_{i-1}, w_i)$ for all $1 \leq i \leq n$. Then $(\ell_0, w_0) \xrightarrow{\psi_1} \dots \xrightarrow{\psi_n} (\ell_n, w_n)$ is a finite run of $\mathcal{A}$. A reduction from the reachability problem for finite directed graphs yields the lower bound. ◀

## 4.2 Characterization for Büchi-accepting Runs

As mentioned above, there are infinite sequences of constraints $\psi_1, \psi_2, \dots$ for which it may not be possible to find $w_0, w_1, w_2 \dots$ such that $\mathbb{N}^* \models \psi(w_{i-1}, w_i)$ for all $i \geq 1$. In the following proposition, we give a decidable characterization for when an infinite path of $\mathcal{A}$ can be completed with string values to obtain an infinite run of $\mathcal{A}$.

▶ **Proposition 9.** *The following three statements are equivalent:*

1. *There exists a Büchi-accepting run of $\mathcal{A}$.*
2. *There exists an infinite path $\pi$ of $\mathcal{A}$ satisfying the following conditions:*
   a. *$\pi$ starts in $\ell_{\text{in}}$,*
   b. *$\pi$ contains infinitely many occurrences of $\ell_{\text{acc}}$, for some $\ell_{\text{acc}} \in \mathcal{L}_{\text{acc}}$, and*
   c. *if $\pi$ contains only finitely many generous constraints, then $\pi$ contains only finitely many reducing constraints.*
3. *There exists a path from $\ell_{\text{in}}$ to $\ell_{\text{acc}}$, for some $\ell_{\text{acc}} \in \mathcal{L}_{\text{acc}}$, and one of the following holds:*
   a. *there exists some stable cycle starting in $\ell_{\text{acc}}$, or*
   b. *there exists some generous cycle starting in $\ell_{\text{acc}}$.*

**Proof.** For the proof from **1.** to **2.**, let $(\ell_0, w_0) \xrightarrow{\psi_1} (\ell_1, w_1) \xrightarrow{\psi_2} \dots$ be a Büchi-accepting run of $\mathcal{A}$. Define $\pi$ to be the infinite path $\ell_0 \xrightarrow{\psi_1} \ell_1 \xrightarrow{\psi_2} \dots$. Clearly, $\pi$ satisfies conditions **2.a** and **2.b**; we prove that condition **2.c** also holds. Towards contradiction, suppose that $\pi$ contains finitely many generous constraints but infinitely many reducing constraints. Then there exists some $i \geq 1$ such that $\psi_j$ is reducing or of the form $x = y$, for all $j \geq i$. Note that this implies $|w_j| \geq |w_{j+1}|$ for all $j \geq i$. Moreover, since there are infinitely many reducing constraints, there exists an infinite sequence $i \leq i_1 < i_2 < i_3 \dots$ of indices such that $\psi_{i_j}$ is reducing and hence $|w_{i_j}| > |w_{i_j+1}|$. Since $|w_j|$ is finite, this leads to a contradiction.

For the proof from **2.** to **1.**, let $\pi = \ell_0 \xrightarrow{\psi_1} \ell_1 \xrightarrow{\psi_2} \dots$ be an infinite path of $\mathcal{A}$ satisfying the three conditions stated in **2**. Using condition **2.c**, we prove that we can complete $\pi$ with string values to yield an infinite run of $\mathcal{A}$; that this run is Büchi-accepting, follows by conditions **2.a** and **2.b**. We distinguish two cases.

- Suppose $\pi$ contains only finitely many generous constraints. By condition **2.c**, $\pi$ contains only finitely many reducing constraints. Then there exists some $i \geq 0$ such that $\psi_j$ is of the form $x = y$ for all $j > i$. Use Proposition 5 with the sequence $\psi_1, \dots, \psi_i$ and $w_{\text{in}} = 0$ to obtain string values $w_0, w_1, \dots, w_i \in \mathbb{N}^+$ such that $\mathbb{N}^* \models \psi_j(w_{j-1}, w_j)$ for all $1 \leq j < i$. Then $(\ell_0, w_0) \xrightarrow{\psi_1} (\ell_1, w_1) \xrightarrow{\psi_2} \dots \xrightarrow{\psi_i} (\ell_i, w_i) \xrightarrow{\psi_{i+1}} (\ell_{i+1}, w_i) \xrightarrow{\psi_{i+2}} (\ell_{i+2}, w_i) \xrightarrow{\psi_{i+3}} \dots$ is an infinite run of $\mathcal{A}$.
- Suppose that $\pi$ contains infinitely many generous constraints. Let $i_1, i_2, i_3 \dots$ be the sequence of all $j \geq 1$ such that $\psi_{i_j}$ is generous. Use Proposition 5 with the sequence $\psi_1, \dots, \psi_{i_1-1}$ and $w_{\text{in}} = 0$ to obtain string values $w_0, w_1, \dots, w_{i_1-1} \in \mathbb{N}^+$ such that $\mathbb{N}^* \models \psi_k(w_{k-1}, w_k)$ for all $1 \leq k < w_{i_1-1}$. For every $j \geq 1$, use Proposition 5 with the

sequence $\psi_{i_j}, \ldots, \psi_{i_{j+1}-1}$ and the initial string value $w_{\text{in}}^j := w_{i_j-1}$ to obtain string values $w_{i_j-1}, w_{i_j}, \ldots, w_{i_{j+1}-1} \in \mathbb{N}^+$ such that $\mathbb{N}^* \models \psi_k(w_{k-1}, w_k)$ for all $i_j \leq k < i_{j+1} - 1$. By the second claim of Proposition 5, since $\psi_{i_j}$ in $\pi_j$ is generous, the initial string value $w_{i_j-1}$ is never rewritten. Hence $\Pi_{i \geq 0} \rho_i$ is an infinite run of $\mathcal{A}$, where $\rho_0 := (\ell_0, w_0) \xrightarrow{\psi_1} (\ell_1, w_1) \xrightarrow{\psi_2} \ldots \xrightarrow{\psi_{i_1-1}} (\ell_{i_1-1}, w_{i_1-1})$, and $\rho_j := (\ell_{i_j-1}, w_{i_j-1}) \xrightarrow{\psi_{i_j}} (\ell_{i_j}, w_{i_j}) \xrightarrow{\psi_{i_j+1}} \ldots \xrightarrow{\psi_{i_{j+1}-1}} (\ell_{i_{j+1}-1}, w_{i_{j+1}-1})$ for all $j \geq 1$.

For the proof from **2.** to **3.**, let $\pi = \ell_0 \xrightarrow{\psi_1} \ell_1 \xrightarrow{\psi_2} \ldots$ be an infinite path of $\mathcal{A}$ satisfying the three conditions stated in **2.** We distinguish two cases:

- Suppose $\pi$ contains only finitely many generous constraints. By condition **2.c**, $\pi$ contains only finitely many reducing constraints. Then there exists some $i \geq 0$ such that $\psi_j$ is of the form $x = y$ for all $j > i$. By condition **2.b**, there are infinitely many indices $j > i$ such that $\ell_j = \ell_{\text{acc}}$. Pick two such indices $j < k$ satisfying $\ell_j = \ell_k = \ell_{\text{acc}}$. We have $\ell_0 = \ell_{\text{in}}$ by condition **2.a**, so that clearly, the path $\ell_0 \xrightarrow{\psi_1} \ldots \xrightarrow{\psi_j} \ell_j$ is a path from $\ell_{\text{in}}$ to $\ell_{\text{acc}}$, and the path $\ell_j \xrightarrow{\psi_{j+1}} \ldots \xrightarrow{\psi_{k-1}} \ell_k$ is a stable cycle starting in $\ell_{\text{acc}}$.
- Suppose $\pi$ contains infinitely many generous constraints. By condition **2.b**, there are infinitely many indices $i \geq 0$ such that $\ell_i = \ell_{\text{acc}}$, so that we can clearly pick three indices $j, k, n$ such that $0 \leq j < k \leq n$, $\ell_j = \ell_n = \ell_{\text{acc}}$, and $\psi_k$ is generous. We have $\ell_0 = \ell_{\text{in}}$ by condition **2.a**, so that clearly, the path $\ell_0 \xrightarrow{\psi_1} \ldots \xrightarrow{\psi_j} \ell_j$ is a path from $\ell_{\text{in}}$ to $\ell_{\text{acc}}$, and the path $\ell_j \xrightarrow{\psi_{j+1}} \ldots \xrightarrow{\psi_{k-1}} \ell_k$ is a generous cycle starting in $\ell_{\text{acc}}$.

For the proof from **3.** to **2.**, suppose $\pi_{\text{in}}$ is a path from $\ell_{\text{in}}$ to $\ell_{\text{acc}}$ for some accepting location $\ell_{\text{acc}} \in \mathcal{L}_{\text{acc}}$, and $\pi_{\text{cyc}}$ is a cycle starting in $\ell_{\text{acc}}$. Clearly $\pi_{\text{in}} \cdot (\pi_{\text{cyc}})^\omega$ is an infinite path of $\mathcal{A}$ satisfying conditions **2.a** and **2.b**. If $\pi_{\text{cyc}}$ is stable, then this path contains only finitely many reducing constraints. If $\pi_{\text{cyc}}$ is generous, then this path contains infinitely many generous constraints. Hence, condition **2.c** holds, too. ◀

▶ **Theorem 10.** *The emptiness problem for constraint automata over $\mathbb{N}^*$ is NL-complete.*

**Proof.** For the upper bound, by Proposition 9, it suffices to decide whether there exists some $\ell_{\text{acc}} \in \mathcal{L}_{\text{acc}}$ such that there exists a path from $\ell_{\text{in}}$ to $\ell_{\text{acc}}$, and one of the following two conditions hold:

- there exists a stable cycle starting in $\ell_{\text{acc}}$, or
- there exists some genereous transition $(\ell, \psi(x, y), \ell') \in E$ such that there exists a path from $\ell_{\text{acc}}$ to $\ell$, and there exists a path from $\ell'$ to $\ell_{\text{acc}}$.

All conditions can be checked in NL. A reduction from the emptiness problem for Büchi automata yields the lower bound. ◀

# 5 Further Results

## 5.1 Testing Equality with the Empty String

We extend the signature $\sigma^{ps}$ by a new symbol $=\varepsilon$, which is interpreted as *equality with the empty string*. This enables us to test whether the string value equals the empty string – very similar to testing whether the value of a counter in a counter automaton is equal to zero, or whether the stack of a pushdown automaton is empty. Let us use $\sigma^{ps\varepsilon}$ to denote this signature. We can give a decidable characterization for Büchi-accepting runs of $\mathcal{A}$ and hence obtain:

▶ **Theorem 11.** *The emptiness problem for constraint automata over the extended signature $\sigma^{ps\varepsilon}$ with domain $\mathbb{N}^*$ is NL-complete.*

## 5.2 Emptiness for Constraint Automata over $\Sigma^*$

The decision procedure for solving the emptiness problem for constraint automata over $\mathbb{N}^*$ relies heavily on the existence of "fresh" letters, of which there are unboundedly many in $\mathbb{N}$. We can clearly not apply this algorithm if the constraint automaton is over the structure $\Sigma^*$, where $\Sigma$ is a *finite* alphabet.

Let $\sigma$ be a relational signature, and let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two $\sigma$-structures. A mapping $h : \mathcal{D}_1 \to \mathcal{D}_2$ is a $\sigma$-embedding if $h$ is injective, and for all symbols $R$ of arity $k$, and all $a_1, \dots, a_k \in \mathcal{D}_1$, $R^{\mathcal{D}_1}(a_1, \dots, a_k)$ holds if, and only if, $R^{\mathcal{D}_2}(h(a_1), \dots, h(a_k))$.

Let $\Sigma = \{a, b\}$. Define the mapping $g : \mathbb{N} \to \Sigma$ by $n \mapsto ab^n a$ for all $n \in \mathbb{N}$, and let $h : \mathbb{N}^* \to \Sigma^*$ be its homomorphic extension, that is, $h(n_1 \dots n_k) = g(n_1) \dots g(n_k)$ for all $n_1, \dots, n_k \in \mathbb{N}$, and $h(\varepsilon) = \varepsilon$. One can easily see that $h$ is a $\sigma^{ps\varepsilon}$-embedding. We can conclude that a constraint automaton over $\Sigma^*$ is a positive instance of the emptiness problem iff the same constraint automaton over $\mathbb{N}^*$ is a positive instance; thus:

▶ **Theorem 12.** *The emptiness problem for constraint automata over the extended signature $\sigma^{ps\varepsilon}$ with domain $\Sigma^*$ is NL-complete.*

## 5.3 Emptiness for Single-Register Automata over $A^*$

*Register automata* (also known as *finite-memory automata*) [17, 13, 24] are a very popular computational model for the analysis of data languages. Like constraint automata, register automata are parameterized by a $\sigma$-structure; in contrast to constraint automata, register automata are "fed" with some input data word, that is a finite or infinite sequence of data values in the domain of the $\sigma$-structure. The data language accepted by such an automaton is the set of input data words for which there is an accepting run. Different to the transitions in constraint automata, the transitions of register automata are labelled with constraints of the form $r \bowtie d$, where $r$ corresponds to one of finitely many *registers* of the automaton, $d$ corresponds to the current datum of the input data word, and $\bowtie$ is a binary relation in $\sigma$. Further, the current input data value can be stored into one of the registers of the automaton after a transition has been taken.

So far, register automata have mostly been studied for the structure $(\mathbb{N}; =)$ and linear dense orders like $(\mathbb{Q}; <, =)$ [17, 24, 13, 4, 8]. The emptiness problem for register automata is decidable and PSPACE-complete (NL-complete if only one register is used) [13]; the decision procedure relies on a finite abstraction of the infinite state space induced by the input register automaton. This abstraction cannot be applied to register automata over $\sigma^{ps\varepsilon}$-structures $\Sigma^*$ and $\mathbb{N}^*$.

A register automaton with a single register that stores the current input datum *in every transition* into the register can actually be regarded as a constraint automaton as defined in Section 2: the current value of the register $r$ corresponds to the value of the variable $x$, and the input datum $d$ corresponds to the value of the variable $y$ at the next position in a run. However, it might be the case that some of the transitions in the register automaton may compare the value of the register *without storing the input datum into the register.* There is no direct way to translate this into constraint automata as defined above. However, an easy extension of our model where we compare $x$ with $y$, but then set the value of $y$ to $x$, would make such a translation possible. It can be easily seen that this extension does not cause any problems when applying the developed decision procedure for solving the emptiness problem, so that we can conclude:

▶ **Theorem 13.** *The emptiness problem for single-register automata over the extended signature $\sigma^{ps\varepsilon}$ (with domains $\mathbb{N}^*$ or $\Sigma^*$) is NL-complete.*

## 5.4 Constraint LTL with a Single Variable over $A^*$

Our result for constraint automata can be used to partially answer the question raised by Demri and Deters [9] concerning the decidability status for CLTL over $\sigma^{ps\varepsilon}$. More detailed, we can prove PSPACE-completeness for the fragment of CLTL that only uses a single variable.

Let $\mathbb{P}$ be a countably infinite set of propositional variables. The set of formulas in $\mathrm{CLTL}_1$ is defined by the following grammar

$$\varphi ::= p \mid \psi \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\varphi \mid \varphi \mathsf{U} \varphi,$$

where $p \in \mathbb{P}$ and $\psi \in \Psi$. $\mathrm{CLTL}_1$ formulas are evaluated over data words over $2^{\mathbb{P}}$ and $A^*$. Formally, let $u = (a_1, w_1)(a_2, w_2)\dots$ and $i \geq 1$. The satisfaction relation $\models$ is defined as follows:

$$\begin{aligned}
(u,i) \models p \quad &\Leftrightarrow \quad p \in a_i \\
(u,i) \models \psi \quad &\Leftrightarrow \quad A^* \models \psi(w_i, w_{i+1}) \\
(u,i) \models \neg\varphi \quad &\Leftrightarrow \quad \mathrm{not}(u,i) \models \varphi \\
(u,i) \models \varphi_1 \vee \varphi_2 \quad &\Leftrightarrow \quad (u,i) \models \varphi_1 \text{ or } (u,i) \models \varphi_2 \\
(u,i) \models \mathsf{X}\varphi \quad &\Leftrightarrow \quad (u,i+1) \models \varphi \\
(u,i) \models \varphi_1 \mathsf{U} \varphi_2 \quad &\Leftrightarrow \quad \exists j \geq i (u,j) \models \varphi_2, \forall i \leq k < j (u,k) \models \varphi_1
\end{aligned}$$

We define $L(\varphi) = \{u \in (2^{\mathbb{P}} \times A^*)^\omega \mid (u,1) \models \varphi\}$. Following the standard translation from LTL to Büchi automata by Vardi and Wolper [29], one can construct from every formula $\varphi$ a constraint automaton $\mathcal{A}_\varphi$ such that $L(\mathcal{A}_\varphi)$ corresponds to $L(\varphi)$ (cf. [14]). In other words, deciding the satisfiability of $\varphi$ can be reduced to deciding the non-emptiness of $L(\mathcal{A}_\varphi)$, so that we obtain the following result.

▶ **Theorem 14.** *The satisfiability problem for $\mathrm{CLTL}_1$ over the extended signature $\sigma^{ps\varepsilon}$ (with domains $\mathbb{N}^*$ or $\Sigma^*$) is PSPACE-complete.*

## References


1. Roberto Amadini, Graeme Gange, Peter Schachte, Harald Søndergaard, and Peter J. Stuckey. String constraint solving: Past, present and future. In *ECAI 2020 - 24th European Conference on Artificial Intelligence*, volume 325, pages 2875–2876. IOS Press, 2020. doi:10.3233/FAIA200431.
2. Philippe Balbiani and Jean-François Condotta. Computational complexity of propositional linear temporal logics based on qualitative spatial or temporal reasoning. In *Frontiers of Combining Systems*, pages 162–176, 2002.
3. Mikolaj Bojanczyk. *Atom Book.* https://www.mimuw.edu.pl/ bojan/paper/atom-book, 2019. URL: https://www.mimuw.edu.pl/~bojan/paper/atom-book.
4. Mikolaj Bojanczyk, Bartek Klin, and Joshua Moerman. Orbit-finite-dimensional vector spaces and weighted register automata. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021. doi:10.1109/LICS52264.2021.9470634.
5. Claudia Carapelle, Shiguang Feng, Alexander Kartzow, and Markus Lohrey. Satisfiability of ECTL* with local tree constraints. *Theory Comput. Syst.*, 61(2):689–720, 2017. doi:10.1007/s00224-016-9724-y.

6 Claudia Carapelle, Alexander Kartzow, and Markus Lohrey. Satisfiability of ECTL$^*$ with constraints. *J. Comput. Syst. Sci.*, 82(5):826–855, 2016. doi:10.1016/j.jcss.2016.02.002.

7 Karlis Cerans. Deciding properties of integral relational automata. In *Automata, Languages and Programming, 21st International Colloquium, ICALP94*, pages 35–46, 1994. doi:10.1007/3-540-58201-0_56.

8 Wojciech Czerwinski, Antoine Mottet, and Karin Quaas. New techniques for universality in unambiguous register automata. In *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021*, volume 198 of *LIPIcs*, pages 129:1–129:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ICALP.2021.129.

9 Stéphane Demri and Morgan Deters. Temporal logics on strings with prefix relation. *J. Log. Comput.*, 26(3):989–1017, 2016. doi:10.1093/logcom/exv028.

10 Stéphane Demri and Deepak D'Souza. An automata-theoretic approach to constraint LTL. *Inf. Comput.*, 205(3):380–415, 2007. doi:10.1016/j.ic.2006.09.006.

11 Stéphane Demri and Régis Gascon. Verification of qualitative Z constraints. *Theor. Comput. Sci.*, 409(1):24–40, 2008. doi:10.1016/j.tcs.2008.07.023.

12 Stéphane Demri and Régis Gascon. The effects of bounding syntactic resources on presburger LTL. *J. Log. Comput.*, 19(6):1541–1575, 2009. doi:10.1093/logcom/exp037.

13 Stéphane Demri and Ranko Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3):16:1–16:30, 2009. doi:10.1145/1507244.1507246.

14 Stéphane Demri and Karin Quaas. Concrete domains in logics: a survey. *ACM SIGLOG News*, 8(3):6–29, 2021. doi:10.1145/3477986.3477988.

15 Régis Gascon. An automata-based approach for CTL$^*$ with constraints. *Electr. Notes Theor. Comput. Sci.*, 239:193–211, 2009. doi:10.1016/j.entcs.2009.05.040.

16 Simon Halfon, Philippe Schnoebelen, and Georg Zetzsche. Decidability, complexity, and expressiveness of first-order logic over the subword ordering. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017*, pages 1–12. IEEE Computer Society, 2017. doi:10.1109/LICS.2017.8005141.

17 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994. doi:10.1016/0304-3975(94)90242-9.

18 Prateek Karandikar and Philippe Schnoebelen. Decidability in the logic of subsequences and supersequences. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015*, volume 45 of *LIPIcs*, pages 84–97. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.84.

19 Alexander Kartzow and Thomas Weidner. Model checking constraint LTL over trees. *CoRR*, abs/1504.06105, 2015. arXiv:1504.06105.

20 Dietrich Kuske. Theories of orders on the set of words. *RAIRO Theor. Informatics Appl.*, 40(1):53–74, 2006. doi:10.1051/ita:2005039.

21 Dietrich Kuske and Georg Zetzsche. Languages ordered by the subword order. In *Foundations of Software Science and Computation Structures - 22nd International Conference, FOSSACS 2019*, volume 11425 of *LNCS*, pages 348–364. Springer, 2019. doi:10.1007/978-3-030-17127-8_20.

22 Gennady S. Makanin. The problem of solvability of equations in a free semi-group. *Math. USSR Sbornik*, 32(2):129–198, 1977.

23 Marvin Minsky. *Computation, Finite and Infinite Machines.* Prentice Hall, 1967.

24 Frank Neven, Thomas Schwentick, and Victor Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004. doi:10.1145/1013560.1013562.

25 Joël Ouaknine and James Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *19th IEEE Symposium on Logic in Computer Science (LICS 2004), 14-17 July 2004, Turku, Finland, Proceedings*, pages 54–63. IEEE Computer Society, 2004. doi:10.1109/LICS.2004.1319600.

26 Wojciech Plandowski. Satisfiability of word equations with constants is in PSPACE. *J. ACM*, 51(3):483–496, 2004. doi:10.1145/990308.990312.

27 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the AMS*, 141:1–23, 1969.

28 Luc Segoufin and Szymon Torunczyk. Automata based verification over linearly ordered data domains. In *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011*, volume 9 of *LIPIcs*, pages 81–92. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011. A technical report containing proofs that we refer here to can be found under `https://www.mimuw.edu.pl/~szymtor//papers/regdata.pdf`.

29 Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings of the Symposium on Logic in Computer Science (LICS '86)*, pages 332–344, 1986.