New Techniques for Universality in Unambiguous Register Automata

Wojciech Czerwiński
University of Warsaw, Poland

Antoine Mottet
Department of Algebra, Faculty of Mathematics and Physics,
Charles University, Prague, Czech Republic

Karin Quaas
University of Leipzig, Germany

Abstract

Register automata are finite automata equipped with a finite set of registers ranging over the domain of some relational structure like $(\mathbb{N}; =)$ or $(\mathbb{Q}; <)$. Register automata process words over the domain, and along a run of the automaton, the registers can store data from the input word for later comparisons. It is long known that the universality problem, i.e., the problem to decide whether a given register automaton accepts all words over the domain, is undecidable. Recently, we proved the problem to be decidable in 2-ExpSpace if the register automaton under study is over $(\mathbb{N}; =)$ and unambiguous, i.e., every input word has at most one accepting run; this result was shortly after improved to 2-ExpTime by Barloy and Clemente. In this paper, we go one step further and prove that the problem is in ExpSpace, and in PSpace if the number of registers is fixed. Our proof is based on new techniques that additionally allow us to show that the problem is in PSpace for single-register automata over $(\mathbb{Q}; <)$. As a third technical contribution we prove that the problem is decidable (in ExpSpace) for a more expressive model of unambiguous register automata, where the registers can take values nondeterministically, if defined over $(\mathbb{N}; =)$ and only one register is used.

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

Keywords and phrases Register Automata, Data Languages, Unambiguity, Unambiguous, Universality, Containment, Language Inclusion, Equivalence

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.129

Category Track B: Automata, Logic, Semantics, and Theory of Programming

Funding *Wojciech Czerwiński*: Supported by the European Research Council (ERC) grant LIPA, grant agreement No 683080.
Antoine Mottet: This author has received funding from the ERC under the European Union's Horizon 2020 research and innovation programme (grant agreement No 771005).
Karin Quaas: Funded by the Deutsche Forschungsgemeinschaft (DFG), project 406907430.

Acknowledgements We thank Lorenzo Clemente, Sławomir Lasota, and Radosław Piórkowski for inspiring discussions on URA.

1 Introduction

Certainly, determinism plays a central role in the research about computation models. Recently, a lot of active research work [1, 6, 3, 16, 14] is devoted to its weaker form: *unambiguity*. A system is *unambiguous* if for every input word there is at most one accepting run. Unambiguous systems exhibit elegant properties; in particular many natural computational problems turn out to be easier compared to the general case. A prominent example is the *universality problem* for finite automata, *i.e.*, the problem of deciding whether a given automaton accepts *every* input word. It is in PTime [18] and even in NC^2 [19] in the unambiguous case, as opposed to PSpace-complete in the general case.

In his seminal overview article about unambiguity, Colcombet [5] states some very natural conjectures about unambiguous systems that are so fundamental that one can be surprised that they are still open. An example conjecture, motivated by the fact that the universality problem for unambiguous finite automata is in PTime, was that for every unambiguous finite automaton the complement of its language can be accepted by another unambiguous finite automaton with at most polynomial size with respect to the size of the original automaton. This conjecture was surprisingly resolved negatively by Raskin [17], who provided a family of automata where a blowup $\Theta(n^{\log\log\log(n)})$ is unavoidable. Still, a lot of other natural questions remain unresolved. Some of them are not algorithmic (as the above one), while others ask for the existence of faster algorithms in the unambiguous case.

Usually one cannot hope for designing more efficient algorithms for the emptiness problem, as it is often easy to transform a nondeterministic system to a deterministic (and thus unambiguous) system which has empty language if and only if the accepted language of the original system is empty. Indeed, it is often sufficient to change the labelling of every transition of the system to its unique transition name. This transformation preserves the emptiness property, but not much more. Therefore there is a hope that the unambiguity assumption may result in faster solving of problems like universality, equivalence and language containment. Recently there was a substantial amount of research in this area [11, 4, 7, 14, 6, 1]. The considered problem is often the universality problem. Indeed, the universality problem is probably the easiest nontrivial problem for which there is a hope to obtain an improvement in the unambiguous case. Equivalence and containment are often not much harder, even though sometimes a bit more involved techniques are needed.

For *register automata*, this line of research was started in [14]. Register automata (RA, for short) extend finite automata with a finite set of registers that take values from an infinite data domain for later comparisons. More detailed, RA are defined over a relational structure, like $(\mathbb{N};=)$ or $(\mathbb{Q};<,=)$; they process finite words over the domain of the relational structure, and the registers can store values from the input word for comparing them using the relations provided by the relational structure. In the more expressive model of register automata *with guessing* (GRA) the registers can even take arbitrary values. In [14] it is shown that for unambiguous RA (URA) over $(\mathbb{N};=)$ the containment problem is in 2-ExpSpace and in ExpSpace for a fixed number of registers. Without the unambiguity assumption, this problem is known to be much harder. Concretely, the universality problem is undecidable as soon as the automaton uses two registers [12, 15, 9], and Ackermann-complete in the one-register case [10]. In the case of GRA even the one-register case is undecidable.[1] The result for URA in [14] was improved by Barloy and Clemente [1] who have shown that the problem is in 2-ExpTime and in ExpTime for a fixed number of registers, using very different tools such as linear recursive sequences in two dimensions.

Our contribution

Our result improves statements of Barloy and Clemente [1] even further. We provide three results shown by two different techniques. Our first technique is to show that in some cases one can assume that only a linear or exponential number of different configurations can be reached via an input word. This claim immediately provides an improved upper bound compared to [1].

[1] A proof for undecidability can be done using a reduction from the undecidable reachability problem for Minsky machines, following the lines of the proof of Theorem 5.2 in [8]. The nondeterministic guessing can be used to express that there exists some decrement for which there is no matching preceding increment.

▶ **Theorem 1.** *The containment problem $L(\mathcal{A}) \subseteq L(\mathcal{B})$ is in* ExpSpace*, if $\mathcal{A}$ is an* RA *and $\mathcal{B}$ is a* URA *over $(\mathbb{N}; =)$. The containment problem is in* PSpace *on inputs $\mathcal{A}, \mathcal{B}$ both having a bounded number of registers.*

This approach can also be applied to unambiguous one-register automata over $(\mathbb{Q}; <, =)$.

▶ **Theorem 2.** *The universality problem for one-register* URA *over $(\mathbb{Q}; <, =)$ is in* PSpace*.*

Our techniques used to show Theorems 1 and 2 differ from techniques used previously in [14]. In [14] a reached configuration was compressed if it was too big. More concretely, if two different data values were equivalent in a certain sense with respect to the current configuration, one of them was eliminated from the configuration, and the resulting configuration was equivalent. In that way, a compressed configuration had only non-equivalent data values and therefore its size was bounded. In this work, we analyse a reachable configuration much more locally, looking only at states with the same location. We show that if there are too many states with the same location in a certain configuration, then it is not possible that the considered unambiguous automaton is universal. In that quite different way we provide a bound on the size of a reachable configuration in an unambiguous universal automaton.

However, we will see that the techniques for URA do not work for unambiguous GRA (GURA), not even in the one-register case. In that case we solve the universality problem, and even the containment problem, with the use of more sophisticated analysis, closer related to the technique from [14]. In short, we show that we can modify the set of reachable configurations such that it becomes small and equivalent in some sense, which also allows us to obtain a more efficient algorithm.

▶ **Theorem 3.** *The containment problem $L(\mathcal{A}) \subseteq L(\mathcal{B})$ is in* ExpSpace*, if $\mathcal{A}$ is a* GRA *over $(\mathbb{N}; =)$ and $\mathcal{B}$ is a one-register* GURA *over $(\mathbb{N}; =)$.*

Independently from our work, Bojańczyk, Klin and Moerman [2] have recently published a result about orbit-finite vector spaces, which, for GURA with register values comparable wrt. a linear order, implies an algorithm that works in ExpTime if the number of registers is not fixed, and in PTime if the number of registers is fixed. However, we believe that our contribution does not only provide an improved complexity of the considered problem (compared with previous results), but also techniques that can be useful in future research on unambiguous systems.

2 Preliminaries

In this section, we define *register automata*, introduced by Kaminski et al. [12, 13]. We start with some basic notions used throughout the paper. We use Σ to denote a finite alphabet, and $\mathbb{N}$ and $\mathbb{Q}$ denote the set of non-negative integers and rational numbers, respectively. Given $a, b \in \mathbb{N}$ with $a \leq b$, we write $[a, b]$ to denote the set $\{a, a+1, \ldots, b\}$.

A *relational structure* is a tuple $\mathcal{D} = (\mathbb{D}; R_1, \ldots, R_k)$, where $\mathbb{D}$ is an infinite domain, and $R_1, \ldots, R_k$ are binary relations over $\mathbb{D}$, and we assume that R_k is the equality relation. In this paper, we are mainly interested in the relational structures $(\mathbb{N}; =)$ of the non-negative integers with equality, and $(\mathbb{Q}; <, =)$ of the rationals with the usual order and equality relations.

A *data word* is a finite sequence $(\sigma_1, d_1) \ldots (\sigma_k, d_k) \in (\Sigma \times \mathbb{D})^*$. If $\Sigma = \{\sigma\}$ is a singleton set, we may write $d_1 \cdot d_2 \cdot \ldots \cdot d_k$ shortly for $(\sigma, d_1)(\sigma, d_2) \ldots (\sigma, d_k)$. We use ε to denote the empty data word. A *data language* is a set of data words. We use $\mathrm{data}(w)$ to denote the set $\{d_1, \ldots, d_k\}$ of all data occurring in w.

Let $\mathbb{D}_\perp$ denote the set $\mathbb{D} \cup \{\perp\}$, where $\perp \notin \mathbb{D}$. We let $\perp \neq d$ for all $d \in \mathbb{D}$, and $\perp$ is incomparable with respect to $\leq$ to all $d \in \mathbb{D}$. We use boldface lower-case letters like $\boldsymbol{a}, \boldsymbol{b}, \ldots, \boldsymbol{u} \ldots$ to denote tuples in $\mathbb{D}_\perp^n$, where $n \in \mathbb{N}$. Given a tuple $\boldsymbol{a} \in \mathbb{D}_\perp^n$, we write a_i for its i-th component, and $\mathrm{data}(\boldsymbol{a})$ denotes the set $\{a_1, \ldots, a_n\} \subseteq \mathbb{D}_\perp$ of all data occurring in $\boldsymbol{a}$.

Let $\mathcal{R} = \{r_1, \ldots, r_n\}$ be a finite set of *registers*. A *register valuation* is a mapping $\boldsymbol{u} : \mathcal{R} \to \mathbb{D}_\perp$; we may write u_i as shorthand for $\boldsymbol{u}(r_i)$. Let $\mathbb{D}_\perp^{\mathcal{R}}$ denote the set of all register valuations. A *register constraint over $\mathcal{D}$ and $\mathcal{R}$* is defined by the grammar

$$\phi ::= \texttt{true} \mid R(t_1, t_2) \mid \neg\phi \mid \phi \wedge \phi$$

where R is a binary relation symbol from the relational structure $\mathcal{D}$, and $t_i \in \{\#\} \cup \{r, \dot{r} \mid r \in \mathcal{R}\}$. Here $\#$ is a symbol representing the current input datum, r refers to the current value of the register r, and $\dot{r}$ refers to the future value of the register r. We use $\Phi(\mathcal{D}, \mathcal{R})$ to denote the set of all register constraints over $\mathcal{D}$ and $\mathcal{R}$. The satisfaction relation $\models$ on $\mathbb{D}_\perp^{\mathcal{R}} \times \mathbb{D} \times \mathbb{D}_\perp^{\mathcal{R}}$ is defined by structural induction as follows. We only give some atomic cases; the other cases can be derived easily. We have $(\boldsymbol{u}, d, \boldsymbol{v}) \models \phi$ if

- ϕ is of the form $\texttt{true}$,
- ϕ is of the form $R(r_i, \#)$ and $\mathcal{D} \models R(u_i, d)$,
- ϕ is of the form $R(\dot{r}_i, r_i)$ and $\mathcal{D} \models R(v_i, u_i)$,
- ϕ is of the form $R(\dot{r}_i, \#)$ and $\mathcal{D} \models R(v_i, d)$.

For example, $\phi := \neg(r = \#) \wedge (\dot{r} = r)$ is a register constraint over $(\mathbb{N}; =)$ and $\mathcal{R} = \{r\}$, and we have $(1, 2, 1) \models \phi$, whereas $(1, 2, 3) \not\models \phi$.

It is important to note that only register constraints of the form $\dot{r} = r$ and $\dot{r} = \#$ uniquely determine the new value of r. In absence of such a register constraint, the register r can nondeterministically take any of infinitely many data values from $\mathbb{D}$, with the following restrictions: the register constraint $\neg(\dot{r} = \#)$ requires that the new value of r is different from the current input datum, so that r may take any datum in $\mathbb{D}$ except for the input datum. Likewise, the register constraint $\neg(\dot{r} = r)$ requires that r takes any datum in $\mathbb{D}$ except for the current value of r. Register automata that allow for such nondeterministic *guessing* of future register values are also called *register automata with guessing*. Formally, a register automaton with guessing (GRA) over $\mathcal{D}$ and Σ is a tuple $\mathcal{A} = (\mathcal{R}, \mathcal{L}, \ell_{\mathrm{init}}, \mathcal{L}_{\mathrm{acc}}, E)$, where

- $\mathcal{R}$ is a finite set of registers,
- $\mathcal{L}$ is a finite set of locations,
- $\ell_{\mathrm{init}} \in \mathcal{L}$ is the initial location,
- $\mathcal{L}_{\mathrm{acc}} \subseteq \mathcal{L}$ is the set of accepting locations,
- $E \subseteq \mathcal{L} \times \Sigma \times \Phi(\mathcal{D}, \mathcal{R}) \times \mathcal{L}$ is a finite set of edges.

If every edge of $\mathcal{A}$ contains some constraint of the form $\dot{r} = r$ or $\dot{r} = \#$, for every $r \in \mathcal{R}$, so that the future value of every register is uniquely determined, then we simply speak of *register automata* (RA, for short), *i.e.*, register automata without guessing. If the number of registers of a GRA (RA, respectively) is fixed to $k \in \mathbb{N}$, then we speak of k-GRA (k-RA, respectively).

A *state* of $\mathcal{A}$ is a pair $(\ell, \boldsymbol{u}) \in \mathcal{L} \times \mathbb{D}_\perp^{\mathcal{R}}$, where ℓ is the current location and $\boldsymbol{u}$ is the current register valuation. Abusing notation a bit, we usually write $\ell(\boldsymbol{u})$ instead of $(\ell, \boldsymbol{u})$. The state $\ell_{\mathrm{init}}(\boldsymbol{u}_{\mathrm{init}})$, where $\boldsymbol{u}_{\mathrm{init}}$ maps every register $r \in \mathcal{R}$ to $\perp$, is called the *initial state*, and a state $\ell(\boldsymbol{u})$ is called *accepting* if $\ell \in \mathcal{L}_{\mathrm{acc}}$. Given two states $\ell(\boldsymbol{u})$ and $\ell'(\boldsymbol{u'})$ and some input letter $(\sigma, d) \in \Sigma \times \mathbb{D}$, we postulate a transition $\ell(\boldsymbol{u}) \xrightarrow{\sigma,d}_{\mathcal{A}} \ell'(\boldsymbol{u'})$ if there exists some edge $(\ell, \sigma, \phi, \ell') \in E$ such that $(\boldsymbol{u}, d, \boldsymbol{u'}) \models \phi$. A *run* of $\mathcal{A}$ on the data word $(\sigma_1, d_1) \ldots (\sigma_k, d_k)$ is a sequence $\ell_0(\boldsymbol{u^0}) \xrightarrow{\sigma_1,d_1}_{\mathcal{A}} \ell_1(\boldsymbol{u^1}) \xrightarrow{\sigma_2,d_2}_{\mathcal{A}} \ldots \xrightarrow{\sigma_k,d_k}_{\mathcal{A}} \ell_k(\boldsymbol{u^k})$ of such transitions. We say that a run as above *starts in* $\ell_0(\boldsymbol{u^0})$; similarly, the run *ends in* $\ell_k(\boldsymbol{u^k})$. A state $\ell(\boldsymbol{u})$ is *reachable*

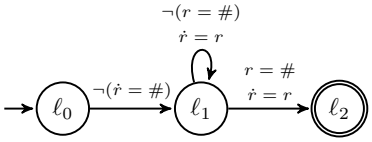


Figure 1 A 1-GURA.

Table 1 Universality over $(\mathbb{N}; =)$.

# registers	RA	URA
$*$	undecidable [8]	**in ExpSpace (Th. 7)**
1	Ackermann-cpl. [10]	**in PSpace (Th. 7)**
# registers	GRA	GURA
$*$	undecidable [13]	
1	undecidable	**in ExpSpace (Th. 3)**

in $\mathcal{A}$ if there exists a run that ends in $\ell(\boldsymbol{u})$. A run is *initialized* if it starts in the initial state, and a run is *accepting* if it ends in some accepting state. A data word w is *accepted from* $\ell(\boldsymbol{u})$ if there exists an accepting run on w that starts in $\ell(\boldsymbol{u})$. The data language *accepted* by $\mathcal{A}$, denoted by $L(\mathcal{A})$, is the set of data words that are accepted from the initial state.

A GRA is *unambiguous* if for every input data word w there is at most one initialized accepting run. Note that unambiguity is a semantic condition; it can be checked in polynomial time [5]. We write GURA and URA to denote unambiguous GRA and RA, respectively.

▶ **Example 4.** Let us study the behaviour of the 1-GRA depicted in Figure 1. The GRA is over $(\mathbb{N}; =)$ and the singleton alphabet $\Sigma = \{\sigma\}$ (we omit the letter σ from all transitions in the figure). Suppose the first input letter is d_1. In order to satisfy the constraint of the transition from ℓ_1 to ℓ_2, the automaton has to nondeterministically guess some datum $d' \neq d_1$ and store it into its register r. Being in the state $\ell_1(d')$, the automaton can only move to the accepting location ℓ_2 if the next input datum is equal to d' (indicated by the constraint $r = d$); for every other input letter, the automaton satisfies the constraint $\neg(r = d)$ and stays in ℓ_1, and it keeps the register value to satisfy the constraint $\dot{r} = r$. In this way, the automaton accepts the language $\{d_1 \cdot \ldots \cdot d_k \mid \exists k \geq 2\ \forall 1 \leq i < k.\ d_i \neq d_k\}$. Note that the automaton is unambiguous: for every input data word there is only one accepting run. We remark that the accepted data language cannot be accepted by any RA (without guessing) [13]. Hence, GRA are more expressive than RA.

In this paper, we study the *universality problem*: given a GRA $\mathcal{A}$, is $\mathcal{A}$ universal, *i.e.*, does $L(\mathcal{A}) = (\Sigma \times \mathbb{D})^*$ hold? In Table 1, we give an overview of the decidability status for register automata over $(\mathbb{N}; =)$, in bold the new results for unambiguous register automata that we present in this paper.

3 Basic Notions for Deciding Universality

For many computational models, a standard approach for solving the universality problem is to explore the (potentially infinite) state space of the automaton under study. Starting from the initial state, the basic idea is to input one letter after the other, and keep track of the *sets of states* that are reached, building a reachability graph whose nodes are the reached sets of states (per input letter). The key property of this state space is that it contains sufficient information to decide whether the automaton under study is universal: this is the case if, and only if, every node of the graph contains an accepting state. Let us formalize this intuition for register automata.

Fix a k-GRA $\mathcal{A} = (\mathcal{R}, \mathcal{L}, \ell_{\text{init}}, \mathcal{L}_{\text{acc}}, E)$ over $\mathcal{D}$ and Σ, for some $k \in \mathbb{N}$. A *configuration of* $\mathcal{A}$ is a subset of $\mathcal{L} \times \mathbb{D}_\perp^k$. The set C_{init}, denoting the singleton set containing the initial state of $\mathcal{A}$, is a configuration, henceforth called the *initial configuration*. Let C be a configuration, and let $(\sigma, d) \in (\Sigma \times \mathbb{D})$. We use $\text{Succ}_{\mathcal{A}}(C, (\sigma, d))$ to denote the *successor of* C *on the input* (σ, d), formally defined by

$$\mathrm{Succ}_{\mathcal{A}}(C,(\sigma,d)) := \{\ell(\boldsymbol{u}) \mid \exists\, \ell'(\boldsymbol{u'}) \in C\ \ell'(\boldsymbol{u'}) \xrightarrow{\sigma,d}_{\mathcal{A}} \ell(\boldsymbol{u})\}.$$

In order to extend this definition to data words, we define inductively $\mathrm{Succ}_{\mathcal{A}}(C,\varepsilon) := C$ and $\mathrm{Succ}_{\mathcal{A}}(C, w\cdot(\sigma,d)) := \mathrm{Succ}_{\mathcal{A}}(\mathrm{Succ}_{\mathcal{A}}(C,w),(\sigma,d))$. We say that a configuration C is *reachable in $\mathcal{A}$ by the data word w* if $C = \mathrm{Succ}_{\mathcal{A}}(C_{\mathrm{init}}, w)$; we say that C is *reachable in $\mathcal{A}$* if there exists some data word w such that C is reachable in $\mathcal{A}$ by w. We say that C is *coverable* if there exists some $C' \supseteq C$ such that C' is reachable in $\mathcal{A}$. Given a configuration C, we use $\mathrm{data}(C)$ to denote the set $\bigcup_{\ell(\boldsymbol{u})\in C}\mathrm{data}(\boldsymbol{u})$ of data occurring in C. Notice that every configuration reachable in an RA (without guessing) is necessarily finite. In contrast, the configuration $\{\ell_1(d') \mid d' \in \mathbb{N}, d' \neq d_1\}$ is reachable in the GRA in Figure 1 by the single-letter data word (σ, d_1). If $C = \{\ell(\boldsymbol{u})\}$ is a singleton set, then we may, in slight abuse of notation, omit the curly brackets and write $\ell(\boldsymbol{u})$.

We say that a configuration C is *accepting* if there exists $\ell(\boldsymbol{u}) \in C$ such that $\ell \in \mathcal{L}_{\mathrm{acc}}$; otherwise we say that C is *non-accepting*. Clearly, $\mathcal{A}$ is universal if, and only if, every configuration reachable in $\mathcal{A}$ is accepting. This suggests to reduce the universality problem to a reachability problem for the state space corresponding to the given input GRA. However, the state space of a GRA is infinite, in two different aspects.

First of all, the state space is *infinitely branching*, as each of the infinite data in $\mathbb{D}$ may give rise to a unique successor configuration. The standard approach for solving this complication is to abstract from concrete data, using the simple observation that, *e.g.*, the data word $3\cdot 4$ is accepted from the state $\ell(4)$ if, and only if, $5\cdot 2$ is accepted from the state $\ell(2)$. This is formalized in the following paragraph.

A *partial isomorphism of $\mathcal{D}_\perp$* is an injective mapping $\pi\colon D \to \mathbb{D}_\perp$ with domain $\mathrm{dom}(\pi) := D \subseteq \mathbb{D}$ such that:

- for every relation R of $\mathcal{D}$ and $a, b \in \mathrm{dom}(\pi)$, we have $(a,b) \in R \Leftrightarrow (\pi(a),\pi(b)) \in R$,
- if $\perp \in D$ then $\pi(\perp) = \perp$.

Let π be a partial isomorphism of $\mathcal{D}_\perp$ and let C be a configuration such that $\mathrm{data}(C) \subseteq \mathrm{dom}(\pi)$. We define the configuration $\pi(C) := \{\ell(\pi(d_1),\dots,\pi(d_k))) \mid \ell(d_1,\dots,d_k) \in C\}$; likewise, if $\{d_1,\dots,d_k\} \subseteq \mathrm{dom}(\pi)$, we define the data word $\pi(w) = (\sigma_1,\pi(d_1))\dots(\sigma_k,\pi(d_k))$. We write $\langle C, w\rangle \sim \langle C', w'\rangle$ if there exists a partial isomorphism of $\mathcal{D}_\perp$ such that $\pi(C) = C'$ and $\pi(w) = w'$.

▶ **Proposition 5.** *Let $\mathcal{A}$ be a* GRA*. If $\langle C, w\rangle \sim \langle C', w'\rangle$, then $\mathrm{Succ}_{\mathcal{A}}(C,w) \sim \mathrm{Succ}_{\mathcal{A}}(C',w')$.*

Secondly, there can be infinitely many reachable configurations even up to the equivalence relation $\sim$. As an example, consider the GURA in Figure 1. For every $n \geq 1$, the configuration $C_n := \{\ell_1(d') \mid d' \in \mathbb{N}\backslash\{d_1,\dots,d_n\}\} \cup \{\ell_2(d_n)\}$ with pairwise distinct data values $d_1,\dots,d_n$ is reachable by the data word $d_1\cdot d_2\cdot\ldots\cdot d_n$, and $C_n \not\sim C_{n'}$ for $n \neq n'$. There are similar examples also for URA, cf. [14].

In order to obtain our results, we will prove that one can solve the reachability problem for the state space of $\mathcal{A}$ by focussing on a subset of configurations reachable in the automaton under study. The concrete methods are different for URA and GURA, however, for both models we will take advantage of Proposition 5 and its simple consequence (cf. [14]).

▶ **Corollary 6.** *Let $\mathcal{A}$ be a* GRA*. If $\langle C, w\rangle \sim \langle C', w'\rangle$ and $\mathrm{Succ}_{\mathcal{A}}(C,w)$ is non-accepting (accepting, respectively), then $\mathrm{Succ}_{\mathcal{A}}(C',w')$ is non-accepting (accepting, respectively).*

4 The Universality Problem for URA over $(\mathbb{N}; =)$

In this section, we study the complexity of the universality problem for URA over the relational structure $(\mathbb{N}; =)$. We prove the following theorem.

▶ **Theorem 7.** *The universality problem is*
- *in* PSpace *for* k-URA *for any fixed* $k \in \mathbb{N}$,
- *in* ExpSpace *for* URA.

We start by showing that we can assume URA to have a specific form that simplifies the coming proofs. Given some k-URA $\mathcal{A}$, we say that $\mathcal{A}$ is *pruned* if for every state $\ell(\boldsymbol{u})$ that is reachable in $\mathcal{A}$ there exists a data word w that is accepted from $\ell(\boldsymbol{u})$, and $u_i \neq u_j$ for all $1 \leq i < j \leq k$, *i.e.*, no datum appears more than once in $\boldsymbol{u}$. The proof of the following proposition is simple and omitted.

▶ **Proposition 8.** *For every* k-URA *one can compute in polynomial time an equivalent pruned* k-URA.

In the following we always assume that a k-URA is pruned, even if we do not explicitly mention it. For simplicity, we also assume that the alphabet of the URA we consider are singletons. The techniques we develop can be easily lifted to the more general case where Σ is not a singleton.

We introduce some constants that bound from above the number of states with the same location occurring in a configuration reachable in a universal URA. Let $\mathcal{A}$ be a k-URA. For a configuration C of $\mathcal{A}$, define $M_C \in \mathbb{N}$ to be the maximal number M such that in C there are M different states with the same location. Define $M_{\mathcal{A}} \in \mathbb{N} \cup \{\infty\}$ to be the supremum of M_C, for C ranging over all the configurations C reachable in $\mathcal{A}$, if $\mathcal{A}$ is a *universal* k-URA, *i.e.*, $L(\mathcal{A}) = \mathbb{D}^*$. In the sequel, we show that $M_{\mathcal{A}} < \infty$. In order to do so, for $k \in \mathbb{N}$, define $\mathbf{M}_k \in \mathbb{N} \cup \{\infty\}$ to be the supremum of all the $M_{\mathcal{A}}$, for $\mathcal{A}$ ranging over pruned and universal k-URA. The main technical result of this section is showing that $\mathbf{M}_k$ is finite and moreover upper-bounded by an exponential function of k.

Let n be the number of locations of $\mathcal{A}$. First observe that showing $\mathbf{M}_k \in \mathbb{N}$ easily implies the existence of a NPSpace algorithm deciding whether $\mathcal{A}$ is universal. Indeed, if $\mathbf{M}_k < \infty$, then every configuration C reachable in $\mathcal{A}$ has size at most $n \cdot \mathbf{M}_k$, as otherwise C contains more than $\mathbf{M}_k$ states with the same location. Thus, in order to decide whether $\mathcal{A}$ is not universal, we can apply the following algorithm:
- By Corollary 6, $\mathcal{A}$ is not universal iff $\mathcal{A}$ does not accept some data word $(\sigma_1, d_1)(\sigma_2, d_2)\ldots$, where $d_i \in \{0, \ldots, i\}$ for all i.
- Guess, letter by letter, an input data word $(\sigma_1, d_1)(\sigma_2, d_2)\ldots$, where $d_i \in \{0, \ldots, i\}$.
- For each $i \geq 1$, define $C_i := \mathrm{Succ}_{\mathcal{A}}(C_{i-1}, (\sigma_i, d_i))$, where $C_0 = C_{\mathrm{init}}$.
- If for some $i \geq 1$, the configuration C_i is not accepting or its size exceeds $n \cdot \mathbf{M}_k$, we know that $\mathcal{A}$ is not universal.
- Otherwise we keep the configuration in the space linear with respect to n and count the length of the word. If the length exceeds the number of possible configurations, then this run is not accepting. The length counter can be also kept in linear space.

The above is hence a PSpace-algorithm for deciding non-universality for k-URA. By Savitch's theorem, there also exists one for deciding universality for k-URA. Moreover, if we show that $\mathbf{M}_k$ is exponential in k, then the above algorithm works in space exponential with respect to k, so is in ExpSpace even without fixing the number of registers k. Therefore, in order to show Theorem 7, it is enough to prove that $\mathbf{M}_k$ is bounded by some exponential function of k. The rest of this Section is devoted mainly to showing the following lemma.

▶ **Lemma 9.** $\mathbf{M}_k \leq (k \cdot 4^k \cdot k!)^k$.

We first give here an argument showing that $\mathbf{M}_k$ is bounded by some doubly-exponential function in k. We show this argument in order to illustrate the techniques, which needed to be refined in our proof of Lemma 9. First recall that the *Ramsey number* $R_m(n)$ is the smallest number of vertices k of the graphs such that any clique of k vertices with its edges coloured on m different colours contain a monochromatic subgraph G of n vertices, namely such that all the edges in G are of the same colour. It can be shown by induction that $R_m(n)$ is finite, and indeed its growth is bounded by $2^{n^{O(m)}}$.

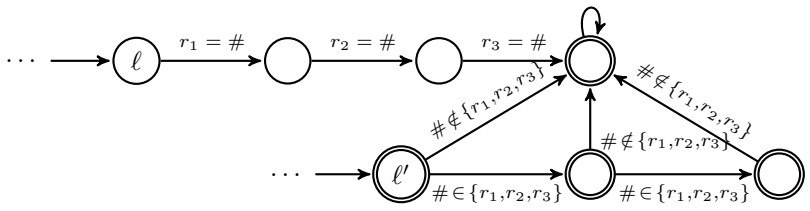
▶ **Proposition 10.** *Every set $T \subseteq \mathbb{D}_\perp^k$ of size at least $R_{k+1}(4^k(k+1)!+1)$ contains a k-full subset of size at least $k+1$.*

Proof. Construct a graph with vertices being tuples from T and edge between $\boldsymbol{t}$ and $\boldsymbol{t'}$ be coloured by the number of data values that $\boldsymbol{t}$ and $\boldsymbol{t'}$ have in common. Clearly the colour belongs to the set $\{0, \ldots, k\}$, so there are $k+1$ colours. Because $|S| \geq R_{k+1}(4^k(k+1)!+1)$ we know from Ramsey's theorem that there are at least $4^k(k+1)!+1$ tuples such that every intersection is of the same size - assume this size to be m. Let S be a set of $4^k(k+1)!+1$ such tuples and let $\boldsymbol{s} \in S$ be one of them. Let $\boldsymbol{s} = (d_1, \ldots, d_k)$. Divide all the other tuples $\boldsymbol{s'}$ into $\binom{k}{m}^2 \cdot m!$ sets depending on which m data values from $\{d_1, \ldots, d_k\}$ belong to $\boldsymbol{s'}$ (there are $\binom{k}{m}$ options), on which positions they are located in $\boldsymbol{s'}$ (also $\binom{k}{m}$ options) and in which order ($m!$ options). It is easy to see that $\binom{k}{m}^2 \cdot m! \leq 4^k k!$, as $\binom{k}{m} \leq 2^k$ and $m! \leq k!$. We divide $4^k(k+1)!$ tuples (we omit $\boldsymbol{s}$) into at most $4^k k!$ sets, so by the pigeonhole principle at least one of them contains at least $k+1$ elements: let these elements be $\boldsymbol{s^1}, \ldots, \boldsymbol{s^{k+1}}$. Notice now that the tuples $\boldsymbol{s^1}, \ldots, \boldsymbol{s^{k+1}}$ form a k-full set: indeed on positions on which they have the m shared data they are identical and on the other positions all the data values are totally different. Thus T contains a k-full set of size $k+1$, which finishes the proof. ◀

Before refining our argument to give an exponential bound, we remark that these techniques alone *cannot* be used to lower even more the complexity of the universality problem for k-URA or for URA. This is because $\mathbf{M}_k \geq k!$, which is the subject of the following lemma.

▶ **Lemma 11.** $\mathbf{M}_k \geq k!$.

Proof. We define a family of pruned universal k-URA $(\mathcal{A}_k)_{k\geq 1}$ over $\Sigma = \{\sigma\}$ such that $M_{\mathcal{A}_k} \geq k!$. Consider the following part of a pruned universal k-URA $\mathcal{A}_k$ (shown for the case $k = 3$):



The rest of the automaton makes sure that the configuration

$$\{\ell(\boldsymbol{u}) \mid \boldsymbol{u} \in \{1, \ldots, k\}^k \text{ is a permutation}\} \cup \{\ell'(1, \ldots, k)\}$$

is reachable in $\mathcal{A}_k$ by the k-letter data word $1 \cdot 2 \cdot \ldots \cdot k$ (*e.g.*, each $\ell(\boldsymbol{u})$ is reached by a path storing the input data in a different order). By taking the (disjoint) union with an unambiguous automaton accepting every data word of length $< k$ and every k letter word that has a repeated data value, we obtain a universal automaton. ◀

Our main tool to prove Lemma 9 is a structural observation, which delivers an understanding of how reachable configurations in universal k-URA can look like. Before diving into it we present an intuition by the following example.

▶ **Example 12.** Let C be a configuration reachable in some universal 2-URA $\mathcal{A}$ over $\Sigma = \{\sigma\}$ by some data word w, and assume that C contains three states $\ell(1,2)$, $\ell(3,4)$ and $\ell(5,6)$ sharing the same location ℓ. We will argue that this is impossible. Assume that from $\ell(1,2)$ the data word $1 \cdot 2 \cdot 7$ is accepted. Then clearly also $3 \cdot 4 \cdot 7$ is accepted from $\ell(3,4)$, and $5 \cdot 6 \cdot 7$ is accepted from $\ell(5,6)$. Let us now consider the data word $8 \cdot 9 \cdot 7$, where 8 and 9 are *fresh* data values, that is, they do not occur in w. Since A is universal, the data word $8 \cdot 9 \cdot 7$ must be accepted from some state $\ell'(d_1, d_2)$ in C. The set $\{d_1, d_2\}$ has only two elements, and so the intersection with at least one of the sets $\{1,2\}$, $\{3,4\}$ and $\{5,6\}$ must be empty. For instance, assume that $\{d_1, d_2\} \cap \{1,2\} = \emptyset$ and $(d_1, d_2) = (3,6)$. Note that $\langle \ell'(3,6), 8 \cdot 9 \cdot 7 \rangle \sim \langle \ell'(3,6), 1 \cdot 2 \cdot 7 \rangle$, so that by Corollary 6, the data word $1 \cdot 2 \cdot 7$ is accepted from state $\ell'(3,6)$, too. But then that there are two accepting runs for $w \cdot 1 \cdot 2 \cdot 7$, a contradiction to the unambiguity of $\mathcal{A}$. Below we generalise this reasoning, in particular to the case where some registers in the reached states keep the same value (*i.e.*, not all are different, as 1, 2, 3, 4, 5 and 6 in the above example). However the intuition stays the same.

We say that a set of tuples $T \subseteq \mathbb{D}_\perp^m$ is *m-full* (or simply *full* if m is clear from the context) if there exists a set of indices $I \subseteq [1,m]$ such that:

- all the tuples in T are identical in indices from I, namely for all $i \in I$ and all $\boldsymbol{t}, \boldsymbol{t'} \in T$ we have $t_i = t'_i$;
- all the data values occurring in tuples in T on indices outside I are different, namely for all $i \notin I$, all $j \in \{1, \ldots, m\}$, and all $\boldsymbol{t}, \boldsymbol{t'} \in T$ we have $t_i \neq t'_j$ unless both $\boldsymbol{t} = \boldsymbol{t'}$ and $i = j$. Note that in particular, this condition applies to the case $\boldsymbol{t'} = \boldsymbol{t}$, and thus $t_i \neq t_j$ whenever $i \notin I$ and $j \in \{1, \ldots, m\}$ are different.

For instance, the set containing the tuples $(1,3,2,4), (1,3,5,6), (1,3,7,8)$ is a 4-full set with $I = \{1,2\}$, and the set $\{(3,7,2,10,8)\}$ is a 5-full set, where $I \subseteq [1,5]$ can be chosen arbitrarily. In contrast, $\{(1,2),(3,1)\}$ is *not* a full set.

For a location ℓ and set of tuples $T \subseteq \mathbb{D}_\perp^k$ we write $\ell(T) = \{\ell(\boldsymbol{t}) \mid \boldsymbol{t} \in T\}$. The following lemma delivers the key observation, which uses the notion of k-full sets.

▶ **Lemma 13.** *If $\mathcal{A}$ is a pruned universal k-URA, then there exists no configuration C reachable in $\mathcal{A}$ such that $\ell(T) \subseteq C$ for some location $\ell \in \mathcal{L}$ and some k-full set of tuples $T \subseteq \mathbb{D}_\perp^k$ of size more than k.*

Proof. Let $\mathcal{A}$ be a pruned universal k-URA, and suppose towards contradiction that there exists a configuration C reachable in $\mathcal{A}$ such that $\ell(T) \subseteq C$ for some location ℓ and some k-full set $T \subseteq \mathbb{D}_\perp^k$ of size more than k. Let w be the data word such that $C = \mathrm{Succ}_{\mathcal{A}}(C_{\mathrm{init}}, w)$. Assume without loss of generality that the indices on which tuples from T are identical are $I = \{1, \ldots, n\}$ for some $n \leq k$. Let us choose some $k+1$ tuples from T, let the i-th of it be of the form $\boldsymbol{t}^i = (c_1, \ldots, c_n, o_1^i, \ldots, o_m^i)$, where $n + m = k$. We call the c_j the *common* data values and the o_j^i the *own* data values of $\boldsymbol{t}^i$. $\mathcal{A}$ is pruned and $\ell(\boldsymbol{t}^1)$ is reachable in $\mathcal{A}$, so there must exist a data word $w_1 \in (\Sigma \times \mathbb{D})^*$ that is accepted from $\ell(\boldsymbol{t}^1)$. Without loss of generality we can assume that w_1 does not contain the own data values of any of the other tuples $\boldsymbol{t}^2, \ldots, \boldsymbol{t}^{k+1}$. Indeed, if this is the case, we can replace synchronously all occurrences of such a data value by a *fresh* data value not occurring in $\mathrm{data}(w)$; the resulting data word is still accepted from $\ell(\boldsymbol{t}^1)$. For every $i \in [2, k+1]$, let w_i be the word w_1 in which for each $j \in [1,m]$ the own data value o_j^1 is replaced by the data value o_j^i. Clearly, for every $i \in [2, k+1]$ $\langle \ell(\boldsymbol{t}^1), w_1 \rangle \sim \langle \ell(\boldsymbol{t}^i), w_i \rangle$, so that by Corollary 6 the data word w_i is accepted from $\ell(\boldsymbol{t}^i)$.

Let us now consider the data word w_{fresh} that is obtained from w_1 by replacing synchronously every occurrence of every o^1_j by some fresh data value each. As $\mathcal{A}$ is universal, also the data word w_{fresh} needs to be accepted from some state in C. Let $q_{\text{fresh}} = \ell'(e_1, \ldots, e_k)$ be the state in C from which w_{fresh} is accepted. Notice that we do not enforce $\ell \neq \ell'$, similarly e_i may be equal to some of the c_i or o^j_i, but this does not have an effect on our reasoning. For each tuple $\boldsymbol{t}^i = (c_1, \ldots, c_n, o^i_1, \ldots, o^i_m)$, let the set of its own data values be $O_i = \{o^i_1, \ldots, o^i_m\}$. By assumption all the sets $O_1, \ldots, O_{k+1}$ are pairwise disjoint. As there are $k+1$ of them, we know that at least one of them is disjoint from the set of data values in the state q_{fresh}, namely with $E = \{e_1, \ldots, e_k\}$. So let $1 \leq i \leq k+1$ be such that $O_i \cap E = \emptyset$. Then we have $\langle q_{\text{fresh}}, w_i\rangle \sim \langle q_{\text{fresh}}, w_{\text{fresh}}\rangle$, so that by Corollary 6 w_i is also accepted from q_{fresh}. In consequence, there are at least two accepting runs over w_i from configuration C, one from $\ell(\boldsymbol{t}^1)$ and one from q_{fresh}. Hence there are at least two initialized accepting runs over $w \cdot w_i$. This is a contradiction to the unambiguity of $\mathcal{A}$. ◀

The following result directly implies Lemma 9, when setting $B = n = k$.

▶ **Lemma 14.** *Every set $T \subseteq \mathbb{D}^n_\perp$ of size at least $(B \cdot 4^n \cdot n!)^n + 1$ contains an n-full subset of size larger than B.*

Proof. Let us denote by $D_{B,n}$ the maximal size of the set of n-tuples such that any n-full subset has size at most B; in other words, $D_{B,n}$ is the least integer such that if X is a set of n-tuples of size $D_{B,n}+1$, then X contains an n-full subset of size $B+1$. Our aim is to show that $D_{B,n} \leq (B \cdot 4^n \cdot n!)^n$. We show it by induction on n.

For the induction base assume $n = 1$. Then any set of data values is a full set, so clearly $D_{B,1} \leq B \leq B \cdot 4^1 \cdot 1!$.

Assume now that $D_{B,m} \leq (B \cdot 4^m \cdot m!)^m$ for all $m < n$ and consider some set $T \subseteq \mathbb{D}^n_\perp$ of n-tuples. Assume that T contains no n-full subset of size larger than B. Pick some tuple $\boldsymbol{t} = (d_1, \ldots, d_n) \in T$. We first show that there can be at most $4^n \cdot n! \cdot D_{B,n-1}$ tuples in T whose data intersect data($\boldsymbol{t}$). Let us denote $N = 4^n \cdot n! \cdot D_{B,n-1}$. Let S be the set of those tuples, assume towards contradiction that the size of S exceeds the bound N. For each tuple $\boldsymbol{s} \in S$ there are at most $2^n - 1$ choices for data($\boldsymbol{s}$) $\cap$ data($\boldsymbol{t}$), so by the pigeonhole principle there are more than $2^n \cdot n! \cdot D_{B,n-1}$ tuples which have the same set data($\boldsymbol{s}$) $\cap$ data($\boldsymbol{t}$). Those data values can occur in tuples from S on at most 2^n different sets of indices, and in at most $n!$ different orders, so by the pigeonhole principle more than $D_{B,n-1}$ tuples from S have the same data values shared with $\boldsymbol{t}$ on the same indices. After ignoring the indices shared with $\boldsymbol{t}$ at most $n-1$ indices remain on these tuples. So by induction assumption there is some full set of size more than B among these tuples, which leads to a contradiction to the assumption that for more than N tuples from T their data intersects data($\boldsymbol{t}$).

Therefore we know that all the tuples but the mentioned N ones have data disjoint with data($\boldsymbol{t}$). Let use denote $\boldsymbol{t^1} = \boldsymbol{t}$ and T_1 to be the set of tuples with data disjoint from data($\boldsymbol{t^1}$). Let $\boldsymbol{t^2} \in T_1$. We now repeat the argument for $\boldsymbol{t^2}$ similarly as for $\boldsymbol{t^1}$ and get that there are at most N tuples with data intersecting data($\boldsymbol{t^2}$). Repeating this argument we get a sequence of tuples $\boldsymbol{t^1}, \boldsymbol{t^2}, \ldots, \boldsymbol{t^m}$ such that for each $i \neq j$ we have data($\boldsymbol{t^i}$) $\cap$ data($\boldsymbol{t^j}$) $= \emptyset$. After adding each tuple $\boldsymbol{t^j}$ to the sequence we define the set T_{j+1} of elements, which have disjoint data with all the tuples $\boldsymbol{t^1}, \ldots, \boldsymbol{t^j}$. As long as T_{j+1} is nonempty we can continue the process. It is easy to see that $|T_{j+1}| \geq |T_j| - N$. Assume now towards contradiction that $D_{B,n} > (B \cdot 4^n \cdot n!)^n$, which implies that $D_{B,n} > (B \cdot 4^n \cdot n!) \cdot D_{B,n-1} = B \cdot N$. We can see now that $|T_B| > 0$, which means that we can construct tuples $\boldsymbol{t^1}, \boldsymbol{t^2}, \ldots, \boldsymbol{t^B}, \boldsymbol{t^{B+1}}$ such that for each $i \neq j$ we have data($\boldsymbol{t^i}$) $\cap$ data($\boldsymbol{t^j}$) $= \emptyset$. This however means that $\{\boldsymbol{t^1}, \ldots, \boldsymbol{t^{B+1}}\}$ is a full set of size $B+1$, which is more than B. This contradicts the assumption, which shows that $D_{B,n} \leq (B \cdot 4^n \cdot n!)^n$ and finishes the proof. ◀

We can now apply a reduction from containment to universality provided by Barloy and Clemente (Lemma 8 in [1]) to obtain Theorem 1 from the introduction.

5 Universality for URA over $(\mathbb{Q}; <, =)$

In this section, we prove Theorem 2 by using the techniques developed in the preceding section. Let us define constants $\mathbf{M}_k^{\mathcal{O}}$ for k-URA with order similarly as $\mathbf{M}_k$ for k-URA. The main technical lemma is the following; Theorem 2 follows.

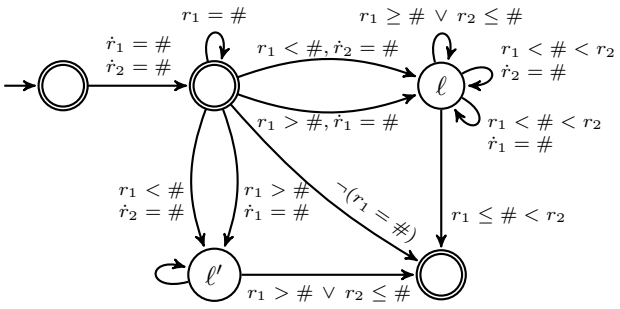
▶ **Lemma 15.** $\mathbf{M}_1^{\mathcal{O}} = 1$.

Proof. Towards contradiction suppose that for some pruned universal 1-URA $\mathcal{A}$ with order there is a configuration C reachable in $\mathcal{A}$ by a data word $w_{\text{pref}} \in (\Sigma \times \mathbb{Q})^*$, such that $\ell(d_1), \ell(d_2) \in C$ for some location ℓ and data values $d_1 < d_2$. Because $\mathcal{A}$ is pruned, there exists a data word $w_1 \in (\Sigma \times \mathbb{Q})^*$ that is accepted from $\ell(d_1)$. Without loss of generality we can assume that w_1 does not contain any data in $(d_1, d_2]$. Indeed, if w_1 contains some datum in $(d_1, d_2]$, then we can replace it synchronously by some datum greater than d_2, while taking care that that the relative order of all data in w_1 is preserved, so that, for the resulting data word w, we have $\langle \ell(d_1), w_1 \rangle \sim \langle \ell(d_1), w \rangle$. By Corollary 6, the resulting data word w is also accepted from $\ell(d_1)$. Notice that for similar reasons, also the data word w_2 obtained from w_1 by replacing every occurrence of d_1 by d_2 is accepted from $\ell(d_2)$. Now, if w_1 does not contain d_1, then $w_1 = w_2$. Hence w_1 is accepted from both $\ell(d_1)$ and $\ell(d_2)$, contradiction to unambiguity of $\mathcal{A}$. So let us assume w_1 contains d_1. Pick some data value d_{fresh} that is fresh, *i.e.*, it does not occur in w_{pref}, and additionally $d_1 < d_{\text{fresh}} < d_2$. We clearly can choose such a fresh data value, as there are infinitely many rational numbers between d_1 and d_2 and only finitely many of them occur in w_{pref}. Let w_{fresh} be the word obtained from w_1 by synchronously replacing every occurrence of d_1 by d_{fresh}. The word w_{fresh} is accepted from some configuration in C, let it be $\ell'(d')$. Notice now that if $d' < d_{\text{fresh}}$, then $\langle \ell'(d'), w_{\text{fresh}} \rangle \sim \langle \ell'(d'), w_2 \rangle$, so that $\ell'(d')$ accepts also w_2 by Corollary 6; in the other case, *i.e.*, if $d' > d_{\text{fresh}}$, then we have $\langle \ell'(d'), w_{\text{fresh}} \rangle \sim \langle \ell'(d'), w_1 \rangle$, so that $\ell'(d')$ also accepts w_1. Therefore in the first case automaton $\mathcal{A}$ has two accepting runs over $w_{\text{pref}} \cdot w_2$ and in the second case over $w_{\text{pref}} \cdot w_1$. This is a contradiction to the unambiguity of $\mathcal{A}$. ◀

The following lemma shows that our techniques by themselves are not sufficient to solve the case of 2-URA with order.

▶ **Lemma 16.** $\mathbf{M}_2^{\mathcal{O}} = \infty$.

Proof. For all $n \geq 1$, consider the configuration $C_n := \{\ell'(1, n)\} \cup \{\ell(1, 2), \ldots, \ell(n-1, n)\}$, which is for all $n \geq 1$ a subset of a configuration reachable in the following pruned universal 2-URA.



The state $\ell'(d_1, d_2)$ keeps track of the first two distinct data read, with $d_1 < d_2$. It is responsible for accepting any datum d outside the interval $[d_1, d_2)$. The state $\ell(x, y)$ is such that $d_1 \leq x < y \leq d_2$ and it is responsible for accepting every datum d' in the interval $[x, y)$. Moreover, if $d' \in (x, y)$, then $\ell(x, y)$ splits into $\ell(x, d')$ and $\ell(d', y)$. The automaton ensures that no two intervals $[x, y), [x', y')$ overlap, and that all the intervals $[x, y)$ present in a configuration cover the interval $[d_1, d_2)$, thus the automaton is unambiguous and universal. ◀

6 Containment for GURA over $(\mathbb{N}; =)$

In this section, we aim to prove the decidability of the universality problem for the more expressive model of GURA. Let us first argue that the techniques developed in Section 4 do not work for GURA.

▶ **Example 17.** One can easily construct a universal 1-GURA with reachable configuration C containing $\ell(0), \ell(1)$, and $\{\ell'\} \times \{n \in \mathbb{N} \mid n \neq 0, 1\}$. If from both ℓ and ℓ' there are outgoing edges with constraint $r = \#$ to some accepting state, then every data word n is accepted from C. In particular, the word 0 is accepted from $\ell(0)$, but we cannot replace 0 by some *fresh* datum to obtain a contradiction as in Example 12.

The example shows that we need more sophisticated methods to solve the universality problem. Moreover, and in contrast to the result for RA, we cannot rely on the reduction from containment to universality by Barloy and Clemente [1], as it holds for RA without guessing only. We hence present a direct proof for containment as stated in Theorem 3. The idea is based on exploring a sufficiently big part of the infinite *synchronized state space* of both automata $\mathcal{A}$ and $\mathcal{B}$, following the approach in [14]. The main difference with [14] lies in the complications that arise due to the fact that a configuration of a GURA may be *infinite*.

6.1 Synchronized Configurations and Bounded Supports

For the rest of this section, let $\mathcal{A} = (\mathcal{R}^{\mathcal{A}}, \mathcal{L}^{\mathcal{A}}, \ell^{\mathcal{A}}_{\text{init}}, \mathcal{L}^{\mathcal{A}}_{\text{acc}}, E^{\mathcal{A}})$ be a GRA with $\mathcal{R}^{A} = \{r_1, \ldots, r_m\}$, and let $\mathcal{B} = (\mathcal{R}^{\mathcal{B}}, \mathcal{L}^{\mathcal{B}}, \ell^{\mathcal{B}}_{\text{init}}, \mathcal{L}^{\mathcal{B}}_{\text{acc}}, E^{\mathcal{B}})$ be a GURA with a single register r.

We aim to reduce the containment problem $L(\mathcal{A}) \subseteq L(\mathcal{B})$ to a reachability problem in $(\mathbb{S}, \Rightarrow)$ where:

- $\mathbb{S}$ is the set of *synchronized configurations* $(\ell(\boldsymbol{d}), C)$, where $\ell(\boldsymbol{d}) \in (\mathcal{L}^{\mathcal{A}} \times \mathbb{N}_{\perp}^{\mathcal{R}^{\mathcal{A}}})$ is a single state of $\mathcal{A}$, and C is a configuration of $\mathcal{B}$,
- $(\ell(\boldsymbol{d}), C) \Rightarrow (\ell'(\boldsymbol{d'}), C')$ if there exists a letter $(\sigma, d) \in (\Sigma \times \mathbb{N})$ such that $\ell(\boldsymbol{d}) \xrightarrow{\sigma, d}_{\mathcal{A}} \ell'(\boldsymbol{d'})$, and $\text{Succ}_{\mathcal{B}}(C, (\sigma, d)) = C'$.

We define $S_{\text{init}} := (\ell^{\mathcal{A}}_{\text{init}}(\boldsymbol{v}_{\text{init}}), C_{\text{init}})$ to be the *initial synchronized configuration of* $\mathcal{A}$ *and* $\mathcal{B}$. We say that a synchronized configuration S' is *reachable* from S if there is a $\Rightarrow$-path from S to S'. S is *reachable* if it is reachable from S_{init}. Call a synchronized configuration $(\ell(\boldsymbol{d}), C)$ *bad* if $\ell \in \mathcal{L}^{\mathcal{A}}_{\text{acc}}$ is an accepting location and C is non-accepting, *i.e.*, $\ell' \notin \mathcal{L}^{\mathcal{B}}_{\text{acc}}$ for all $(\ell', u) \in C$. Thus, a bad synchronized configuration is reachable iff $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$.

We extend the equivalence relation $\sim$ defined in Section 3 to synchronized configurations in a natural manner, *i.e.*, $(\ell(\boldsymbol{d}), C) \sim (\ell(\boldsymbol{d'}), C')$ if there exists a partial isomorphism π of $\mathbb{N}_{\perp}$ such that $\text{data}(\boldsymbol{d}) \cup \text{data}(C) \subseteq \text{dom}(\pi)$ and satisfying $\pi(\boldsymbol{d}) = \boldsymbol{d'}$ and $\pi(C) = C'$. Clearly, an analogue of Corollary 6 holds for this extended relation. In particular, we have the following:

▶ **Proposition 18.** *Let S, S' be two synchronized configurations of $(\mathbb{S}, \Rightarrow)$ such that $S \sim S'$. If S reaches a bad synchronized configuration, so does S'.*

The *support* of a configuration C of $\mathcal{B}$ is the set $\mathrm{supp}(C)$ of data d' such that at least one of the following two conditions holds:

- $\ell(d') \in C$ for some $\ell \in \mathcal{L}$ such that $(\{\ell\} \times \mathbb{D}) \cap C$ is finite,
- $\ell(d') \notin C$ for some $\ell \in \mathcal{L}$ such that $(\{\ell\} \times \mathbb{D}) \cap C$ is cofinite.

Note that $\mathrm{supp}(C) \subseteq \mathrm{data}(w)$ whenever $C = \mathrm{Succ}_{\mathcal{B}}(C_{\mathrm{init}}, w)$.

Let $S = (\ell(\boldsymbol{d}), C)$ be a synchronized configuration, and let $a, b \in \mathrm{supp}(C)$ be two data values in the support of C. We say that *a and b are indistinguishable in S*, written $a \equiv_S b$, if $a, b \notin \mathrm{data}(\boldsymbol{d})$ and $\{\ell \in \mathcal{L} \mid \ell(a) \in C\} = \{\ell \in \mathcal{L} \mid \ell(b) \in C\}$.

Given a configuration C of $\mathcal{B}$, we define for every datum $d \in \mathbb{N}$ the sets $C_d^+ := \{\ell(d) \in \mathcal{L} \times \{d\} \mid \ell(d) \in C \text{ and } \mathrm{data}(C \cap (\{\ell\} \times \mathbb{N})) \text{ is finite}\}$, and $C_d^- := \{\ell(d) \in \mathcal{L} \times \{d\} \mid \ell(d) \notin C \text{ and } \mathrm{data}(C \cap (\{\ell\} \times \mathbb{N})) \text{ is infinite}\}$.

We give here an example for the definition of C_d^+ and C_d^-.

▶ **Example 19.** Let $C = \{\ell_1(0), \ell_1(1)\} \cup \{\ell_2(d) \mid d \in \mathbb{N}\backslash\{1,2\}\} \cup \{\ell_3(d) \mid d \in \mathbb{N}\backslash\{0,1\}\}$. Then

$$\begin{array}{lll} C_0^+ = \{\ell_1(0)\} & C_1^+ = \{\ell_1(1)\} & C_2^+ = \emptyset \\ C_0^- = \{\ell_3(0)\} & C_1^- = \{\ell_2(1), \ell_3(1)\} & C_2^- = \{\ell_2(2)\} \end{array}$$

We say that a configuration C is *essentially coverable* if for every two $\ell(u), \ell'(u') \in C$, the set $\{\ell(u), \ell'(u')\}$ is coverable.

▶ **Proposition 20.** *Let C be an essentially coverable configuration, and let $b \in \mathrm{supp}(C)$. Then $(C\backslash C_b^+) \cup C_b^-$ is essentially coverable, too.*

Proof. Let $\ell(c), \ell'(c') \in ((C\backslash C_b^+) \cup C_b^-)$. If $\ell(c), \ell'(c') \in C\backslash C_b^+$, then $\{\ell(c), \ell'(c')\}$ is coverable by essential coverability of C. Suppose $\ell(c), \ell'(c') \in C_b^-$. By definition of C_b^-, $c = c' = b$. Pick some value $e \in \mathbb{N}\backslash\{b\}$ such that $\ell(e), \ell'(e) \in C$. Note that such a value e must exist, as by definition of C_b^-, the sets $\mathrm{data}((\{\ell\} \times \mathbb{N}) \cap C)$ and $\mathrm{data}((\{\ell'\} \times \mathbb{N}) \cap C)$ are cofinite, and hence their intersection is non-empty. By essential coverability of C, $\{\ell(e), \ell'(e)\}$ is coverable. There must thus exist some data word w such that $\{\ell(e), \ell'(e)\} \subseteq \mathrm{Succ}(\ell_{\mathrm{init}}(\bot), w)$. Let π be any partial isomorphism satisfying $\pi(e) = b$ and whose domain contains $\mathrm{data}(w)$. Clearly, $\{\ell(b), \ell'(b)\} \subseteq \mathrm{Succ}(\ell_{\mathrm{init}}(\bot), \pi(w))$, and hence $\{\ell(b), \ell'(b)\}$ is coverable. Finally, suppose $\ell(c) \in C \setminus C_b^+$ and $\ell'(c') \in C_b^-$. By definition, we have $c \neq b = c'$. Since $\mathrm{data}((\ell' \times \mathbb{N}) \cap C)$ is cofinite, there is $d \neq c$ such that $\ell'(d) \in C$. By essential coverability of C, there exists a data word w such that $\{\ell(c), \ell'(d)\} \subseteq \mathrm{Succ}(\ell_{\mathrm{init}}(\bot), w)$. By picking a partial isomorphism π such that $\pi(d) = c'$ and $\pi(c) = c$, we obtain that $\{\ell(c), \ell'(c')\} \subseteq \mathrm{Succ}(\ell_{\mathrm{init}}(\bot), \pi(w))$, which concludes the proof. ◀

The following is the main technical result of this section.

▶ **Proposition 21.** *Let $S = (\ell^{\mathcal{A}}(\boldsymbol{d}), C)$ be a synchronized configuration of $\mathcal{A}$ and $\mathcal{B}$ such that C is essentially coverable, and let $a \neq b$ be such that $a, b \in \mathrm{supp}(C)$ and $a \equiv_S b$. Then S reaches a bad configuration in $(\mathbb{S}, \Rightarrow)$ if, and only if, $S' := (\ell^{\mathcal{A}}(\boldsymbol{d}), (C \setminus C_b^+) \cup C_b^-)$ reaches a bad configuration in $(\mathbb{S}, \Rightarrow)$.*

Proof. ($\Leftarrow$) Suppose there exists some data word w such that there exists an accepting run of $\mathcal{A}$ on w that starts in $\ell^{\mathcal{A}}(\boldsymbol{d})$, and $\mathrm{Succ}_{\mathcal{B}}(C\backslash C_b^+ \cup C_b^-, w)$ is non-accepting. We assume in the following that $\mathrm{Succ}_{\mathcal{B}}(C_b^+, w)$ is accepting; otherwise we are done. Let $\ell^+(b) \in C_b^+$ be the unique state such that $\mathrm{Succ}_{\mathcal{B}}(\ell^+(b), w)$ is accepting. In the following, we prove that we can without loss of generality assume that w does not contain any a's. Pick some $a' \in \mathbb{N}$ such that $a' \notin \mathrm{data}(w) \cup \mathrm{supp}(C) \cup \mathrm{data}(\boldsymbol{d})$. Let π be the isomorphism defined by $\pi(a) = a'$,

$\pi(a') = a$, and $\pi(d) = d$ for all $d \in \mathbb{N}_\perp \backslash \{a, a'\}$. Then $\pi\langle \ell^{\mathcal{A}}(\boldsymbol{d}), w\rangle = \langle \ell^{\mathcal{A}}(\boldsymbol{d}), \pi(w)\rangle$ (as $a \notin \mathrm{data}(\boldsymbol{d})$ by $a \equiv_S b$), and $\pi\langle \ell^+(b), w\rangle = \langle \ell^+(b), \pi(w)\rangle$. By Corollary 6, there exists an accepting run of $\mathcal{A}$ on $\pi(w)$ that starts in $\ell^{\mathcal{A}}(\boldsymbol{d})$, and $\mathrm{Succ}_{\mathcal{B}}(\ell^+(b), \pi(w))$ is accepting. We prove that $\mathrm{Succ}_{\mathcal{B}}(\ell(c), \pi(w))$ is non-accepting, for every $\ell(c) \in C \setminus \{\ell^+(b)\} \cup C_b^-$: first, let $\ell(c) \in C \setminus \{\ell^+(b)\}$. By essential coverability of C, $\{\ell^+(b), \ell(c)\}$ is coverable. By unambiguity of $\mathcal{B}$, $\mathrm{Succ}_{\mathcal{B}}(\ell(c), \pi(w))$ must be non-accepting. Second, let $\ell(c) \in C_b^-$. But then $c = b$, and hence $\pi\langle \ell(c), w\rangle = \langle \ell(c), \pi(w)\rangle$. By assumption, $\mathrm{Succ}_{\mathcal{B}}(\ell(c), w)$ is non-accepting, so that by Corollary 6, $\mathrm{Succ}_{\mathcal{B}}(\ell(c), \pi(w))$ is non-accepting, too. Note that $\pi(w)$ indeed does not contain any a's. We can hence continue the proof assuming that w does not contain any a's.

Next, we prove that if we replace all b's occurring in w by some fresh datum not occurring in $\mathrm{supp}(C) \cup \mathrm{data}(w) \cup \mathrm{data}(\boldsymbol{d})$, we obtain a data word that guides S to a bad synchronized configuration. Formally, pick some datum $b' \notin \mathrm{data}(w) \cup \mathrm{supp}(C) \cup \mathrm{data}(\boldsymbol{d})$, and let π be the partial isomorphism defined by $\pi(b) = b'$, $\pi(b') = b$, and $\pi(d) = d$ for all $d \in \mathbb{N}_\perp \backslash \{b, b'\}$. Note that $\pi(w)$ does not contain any a's or b's. Clearly, $\pi\langle \ell^{\mathcal{A}}(\boldsymbol{d}), w\rangle = \langle \ell^{\mathcal{A}}(\boldsymbol{d}), \pi(w)\rangle$. By Corollary 6, there still exists an accepting run of $\mathcal{A}$ on $\pi(w)$ that starts in $\ell^{\mathcal{A}}(\boldsymbol{d})$. We prove that $\mathrm{Succ}_{\mathcal{B}}(C, \pi(w))$ is non-accepting. Let $\ell(c) \in C$. We distinguish three cases.

1. Let $c \notin \{b, b'\}$. Then $\pi\langle \ell(c), w\rangle = \langle \ell(c), \pi(w)\rangle$. Since $\mathrm{Succ}_{\mathcal{B}}(\ell(c), w)$ is non-accepting by assumption, so that by Corollary 6 also $\mathrm{Succ}_{\mathcal{B}}(\ell(c), \pi(w))$ is non-accepting.
2. Let $c = b$. By $a \equiv_C b$, the state $\ell(a)$ is in C and $\ell(a), \pi(w) \sim \ell(c), \pi(w)$ since a and c do not appear in w. By essential coverability of C, $\{\ell(a), \ell(c)\} \subseteq C$ is coverable. By unambiguity of $\mathcal{B}$, we obtain that $\mathrm{Succ}_{\mathcal{B}}(\ell(c), \pi(w))$ is non-accepting.
3. Let $c = b'$. Note that $\pi\langle \ell(b), w\rangle = \langle \ell(b'), \pi(w)\rangle$. Recall that $b' \notin \mathrm{supp}(C)$. This implies that $\mathrm{data}(C \cap (\{\ell\} \times \mathbb{N}_\perp))$ is cofinite. We distinguish two cases.
   - $b \in \mathrm{data}(C \cap (\{\ell\} \times \mathbb{N}_\perp))$, i.e., $\ell(b) \in C$. But note that $\ell(b) \notin C_b^+$ by cofiniteness of $\mathrm{data}(C \cap (\{\ell\} \times \mathbb{N}_\perp))$. Hence $\ell(b) \in C \backslash \{\ell^+(b)\}$.
   - $b \notin \mathrm{data}(C \cap (\{\ell\} \times \mathbb{N}_\perp))$, i.e., $\ell(b) \in C_b^-$.

   In both cases, we have proved above that $\mathrm{Succ}(\ell(b), w)$ is non-accepting. By $\pi\langle \ell(b), w\rangle = \langle \ell(b'), \pi(w)\rangle$ and Corollary 6, $\mathrm{Succ}_{\mathcal{B}}(\ell(b'), \pi(w))$ is non-accepting, too.

Altogether we have proved that $\mathrm{Succ}_{\mathcal{B}}(C, \pi(w))$ is non-accepting, while there exists some accepting run of $\mathcal{A}$ on $\pi(w)$ starting in $\ell^{\mathcal{A}}(\boldsymbol{d})$. This concludes the proof for the ($\Leftarrow$)-direction.

($\Rightarrow$) Suppose there exists some data word w such that there exists some accepting run of $\mathcal{A}$ on w starting in $\ell^{\mathcal{A}}(\boldsymbol{d})$, and $\mathrm{Succ}_{\mathcal{B}}(C, w)$ is non-accepting. We assume in the following that $\mathrm{Succ}_{\mathcal{B}}(C \setminus C_b^+ \cup C_b^-, w)$ is accepting; otherwise we are done. Let $\ell^-(b)$ be a state in C_b^- such that $\mathrm{Succ}_{\mathcal{B}}(\ell^-(b), w)$ is accepting. Pick some datum $a' \in \mathbb{N}_\perp$ such that $a' \notin \mathrm{data}(w) \cup \mathrm{supp}(C) \cup \mathrm{data}(\boldsymbol{d})$. Let π be the isomorphism defined by $\pi(b) = a$, $\pi(a) = a'$, $\pi(a') = b$, and $\pi(d) = d$ for all $d \in \mathbb{N} \backslash \{a, b, a'\}$. Clearly, $\pi\langle \ell^{\mathcal{A}}(\boldsymbol{d}), w\rangle = \langle \ell^{\mathcal{A}}(\boldsymbol{d}), \pi(w)\rangle$, so that by Corollary 6, there exists some accepting run of $\mathcal{A}$ on $\pi(w)$ starting in $\ell^{\mathcal{A}}(\boldsymbol{d})$. We prove that $\mathrm{Succ}_{\mathcal{B}}(C \backslash C_b^+ \cup C_b^-, \pi(w))$ is non-accepting. Let $\ell(c) \in C \backslash C_b^+ \cup C_b^-$. We distinguish the following cases:

1. Let $c = a$, i.e., $\ell(a) \in C$. By $a \equiv_S b$, we also have $\ell(b) \in C$. Note that $\pi\langle \ell(b), w\rangle = \langle \ell(a), \pi(w)\rangle$ and that $\ell(b) \neq \ell^-(b)$. By assumption, $\mathrm{Succ}_{\mathcal{B}}(\ell(b), w)$ is non-accepting. By Corollary 6, $\mathrm{Succ}_{\mathcal{B}}(\ell(a), \pi(w))$ is non-accepting, too.
2. Let $c \neq a$. Note that also $\pi\langle \ell^-(b), w\rangle = \langle \ell^-(a), \pi(w)\rangle$. Recall that $\mathrm{Succ}_{\mathcal{B}}(\ell^-(b), w)$ is accepting. By Corollary 6, $\mathrm{Succ}_{\mathcal{B}}(\ell^-(a), \pi(w))$ is accepting. We prove below that $\{\ell^-(a), \ell(c)\}$ is coverable. By unambiguity of $\mathcal{B}$, this directly implies that $\mathrm{Succ}_{\mathcal{B}}(\ell(c), \pi(w))$ is non-accepting.

   Recall that $\{d \in \mathbb{N} \mid \ell^-(d) \in C\}$ is cofinite. Pick some datum $d \in \mathbb{N} \backslash \{c\}$ such that $\ell^-(d) \in C$. We distinguish two cases.

- Assume $\ell(c) \in C \backslash C_b^+$. Since C is essentially coverable, the set $\{\ell^-(d), \ell(c)\}$ is coverable. Hence there must exist some data word u such that $\{\ell^-(d), \ell(c)\} \subseteq \mathrm{Succ}_{\mathcal{B}}(\ell_{\mathrm{init}}(\bot), u)$. Let π' be a partial isomorphism satisfying $\pi'(d) = a$, $\pi'(a) = d$, and $\pi'(e) = e$ for all $e \in \mathrm{data}(u) \cup \{c\}$. Then $\{\ell^-(a), \ell(c)\} \subseteq \mathrm{Succ}_{\mathcal{B}}(\ell_{\mathrm{init}}(\bot), \pi'(u))$, hence $\{\ell^-(a), \ell(c)\}$ is coverable.
- Second suppose $\ell(c) \in C_b^-$, i.e., $c = b$. This implies that $\{e \in \mathbb{N} \mid \ell(e) \in C\}$ is cofinite. Pick some datum $e \in \mathbb{N} \backslash \{d\}$ such that $\ell(e) \in C$. Since C is essentially coverable, the set $\{\ell^-(d), \ell(e)\}$ is coverable. Hence there must exist some data word u such that $\{\ell^-(d), \ell(e)\} \subseteq \mathrm{Succ}_{\mathcal{B}}(\ell_{\mathrm{init}}(\bot), u)$. Let π' be a partial isomorphism satisfying $\pi'(d) = a$, $\pi'(a) = d$, $\pi'(b) = e$, $\pi'(e) = b$, and $\pi'(f) = f$ for all $f \in \mathrm{data}(u)$. Then $\{\ell(b), \ell^-(a)\} \subseteq \mathrm{Succ}_{\mathcal{B}}(\ell_{\mathrm{init}}(\bot), \pi'(u))$, hence $\{\ell(c), \ell^-(a)\}$ is coverable.

Altogether we have proved that $\mathrm{Succ}_{\mathcal{B}}((C \backslash C_b^+) \cup C_b^-, \pi(w))$ is non-accepting, while there is an accepting run of $\mathcal{A}$ on $\pi(w)$ starting in $\ell^{\mathcal{A}}(\boldsymbol{d})$. This finishes the proof for the ($\Rightarrow$)-direction, and thus the proof of the Proposition. ◀

References

1. Corentin Barloy and Lorenzo Clemente. Bidimensional linear recursive sequences and universality of unambiguous register automata. In *Proceedings of STACS'21*, 2021. To appear.
2. Mikolaj Bojanczyk, Bartek Klin, and Joshua Moerman. Orbit-finite-dimensional vector spaces and weighted register automata. In *Proceedings of LICS'21*, 2021. To appear. `arXiv:2104.02438`.
3. Alin Bostan, Arnaud Carayol, Florent Koechlin, and Cyril Nicaud. Weakly-Unambiguous Parikh Automata and Their Link to Holonomic Series. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 114:1–114:16, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.ICALP.2020.114`.
4. Nicolas Bousquet and Christof Löding. Equivalence and inclusion problem for strongly unambiguous Büchi automata. In *Language and Automata Theory and Applications, 4th International Conference, LATA 2010, Trier, Germany, May 24-28, 2010. Proceedings*, pages 118–129, 2010. `doi:10.1007/978-3-642-13089-2_10`.
5. Thomas Colcombet. Unambiguity in automata theory. In *Proceedings of DCFS 2015*, pages 3–18, 2015.
6. Wojciech Czerwinski, Diego Figueira, and Piotr Hofman. Universality problem for unambiguous VASS. In *Proceedings of CONCUR 2020*, pages 36:1–36:15, 2020.
7. Laure Daviaud, Marcin Jurdzinski, Ranko Lazic, Filip Mazowiecki, Guillermo A. Pérez, and James Worrell. When is containment decidable for probabilistic automata? In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 121:1–121:14, 2018. `doi:10.4230/LIPIcs.ICALP.2018.121`.
8. 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`.
9. Stéphane Demri, Ranko Lazic, and Arnaud Sangnier. Model checking freeze LTL over one-counter automata. In *Proceedings of FOSSACS 2008*, pages 490–504, 2008.
10. Diego Figueira, Santiago Figueira, Sylvain Schmitz, and Philippe Schnoebelen. Ackermannian and primitive-recursive bounds with dickson's lemma. In *Proceedings of LICS 2011*, pages 269–278, 2011.
11. Dimitri Isaak and Christof Löding. Efficient inclusion testing for simple classes of unambiguous ω-automata. *Inf. Process. Lett.*, 112(14-15):578–582, 2012. `doi:10.1016/j.ipl.2012.04.010`.

12 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

13 Michael Kaminski and Daniel Zeitlin. Finite-memory automata with non-deterministic reassignment. *International Journal of Foundations of Computer Science*, Volume 21, Issue 05, 2010.

14 Antoine Mottet and Karin Quaas. The containment problem for unambiguous register automata. In *Proceedings of STACS 2019*, pages 53:1–53:15, 2019.

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

16 Erik Paul. Finite Sequentiality of Finitely Ambiguous Max-Plus Tree Automata. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 137:1–137:15, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ICALP.2020.137.

17 Mikhail Raskin. A superpolynomial lower bound for the size of non-deterministic complement of an unambiguous automaton. In *Proceedings of ICALP 2018*, pages 138:1–138:11, 2018.

18 Richard Edwin Stearns and Harry B. Hunt III. On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM J. Comput.*, 14(3):598–611, 1985.

19 Wen-Guey Tzeng. On path equivalence of nondeterministic finite automata. *Inf. Process. Lett.*, 58(1):43–46, 1996.