

# On the Containment Problem for Unambiguous Single-Register Automata with Guessing

Antoine Mottet  
Charles University Prague

Karin Quaas  
Universität Leipzig

## Abstract

Register automata extend classical finite automata with a finite set of registers that can store data from an infinite data domain for later equality comparisons with data from an input data word. While the registers in the original model of register automata, introduced in 1994 by Kaminski and Francez, can only store data occurring in the data word processed so far, we study here the more expressive class of **register automata with guessing**, where registers can nondeterministically take any value from the infinite data domain, even if this data does not occur in the input data word. It is well known that the containment problem, i.e., the problem of deciding for two given register automata with guessing  $\mathcal{A}$  and  $\mathcal{B}$ , whether the language  $L(\mathcal{A})$  accepted by  $\mathcal{A}$  is contained in the language  $L(\mathcal{B})$  accepted by  $\mathcal{B}$ , is undecidable, even if  $\mathcal{B}$  only uses a single register. We prove that the problem is decidable if  $\mathcal{B}$  is unambiguous and uses a single register.

## 1 Introduction

Register automata [3, 4] are a widely studied computational model that extend classical finite automata with finitely many *registers* that can take values from an infinite set and perform equality comparisons with data from the input word. Register automata accept *data languages*, that is sets of *data words* over  $\Sigma \times \mathbb{D}$ , where  $\Sigma$  is a finite alphabet, and  $\mathbb{D}$  is an infinite set called the *data domain*.

As an example, consider the register automaton in Figure 1 using a single register  $r$  ( $\dot{r}$  refers to the future value of  $r$ ). This automaton processes finite data words over  $\Sigma \times \mathbb{D}$ . We assume that  $\Sigma = \{\sigma\}$  is a singleton, so that we omit the letter  $\sigma$  from all transitions and input words, and  $\mathbb{D} = \mathbb{N}$ . Let us study the behaviour of the automaton: starting in the initial location  $\ell_0$  and processing the first input letter  $d$ , the automaton can only move to  $\ell'$  if it satisfies the register constraint  $\dot{r} \neq$ . This constraint requires the register  $r$ , when reaching  $\ell_1$ , to store a data value  $d' \in \mathbb{N}$  such that  $d' \neq d$ . The automaton can nondeterministically *guess* such a datum  $d'$ . Being in  $\ell_1$  with the register holding the value  $d'$ , by the constraint  $= r$ , it can only move to the accepting location  $\ell_2$  if it reads the input letter  $d'$ ; for every other input letter, satisfying the constraint  $\neq r$ , the automaton stays in  $\ell_1$ , keeping the register value (indicated by the constraint  $\dot{r} = r$ ). For instance, for the input data word  $w_{ad} = 1\,2\,2\,3$ , there are infinitely many distinct runs (one for each guessed datum different from 1), but only one accepting run, namely

$$(\ell_0, \perp) \xrightarrow{1} (\ell_1, 3) \xrightarrow{2} (\ell_1, 3) \xrightarrow{2} (\ell_1, 3) \xrightarrow{3} (\ell_2, 3).$$

We write  $L_{ad} = \{d_1 \dots d_k \mid \forall k \geq 2 \, \forall 1 \leq i < k. d_i \neq d_k\}$  to denote the set of data words that is accepted by the automaton in Figure 1 (*ad* standing for *all different*).

We remark that the nondeterministic *guessing* of data values to store them into registers for future comparisons is not allowed in the original model of register automata, introduced by Kaminski and Francez [3], and studied *e.g.* in [2, 5, 8]. In fact, the model that we study here is strictly more expressive (with respect to acceptance of data languages) than the classical model. In order to distinguish the model with guessing from the classical model without guessing, we explicitly refer to the former by *register automata with guessing*, GRA for short.

The GRA in Figure 1 is not *deterministic*: being in  $\ell_0$  and processing the first input datum  $d$ , it can nondeterministically *guess* any datum  $d'$  such that  $d' \neq d$  for storage in  $r$ . However, one can

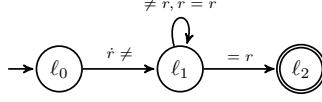


Figure 1: A GURA with a single register  $r$  and over a singleton alphabet (we omit the labels at the edges).

easily see that for every input data word there is *at most one accepting run*, uniquely determined by the single guessed datum  $d'$ . We call automata that have, for every input word, at most one accepting run, *unambiguous*. One of the main open problems concerning unambiguous register automata with guessing (GURA, for short) is whether the class of data languages accepted by GURA is closed under complementation<sup>1</sup>. (In contrast, it is known that data languages accepted by unambiguous register automata (without guessing) are *not* closed under complementation; for instance, the complement of  $L_{ad}$  can be accepted by an unambiguous register automata with a single register, but  $L_{ad}$  cannot even be accepted by any nondeterministic register automaton (without guessing) [4].)

In this paper, we study the *containment problem*: given two GRA  $\mathcal{A}$  and  $\mathcal{B}$ , does  $L(\mathcal{A}) \subseteq L(\mathcal{B})$  hold? Here,  $L(\mathcal{A})$  and  $L(\mathcal{B})$ , respectively, denote the set of data words accepted by  $\mathcal{A}$  and  $\mathcal{B}$ , respectively. This problem, playing a central role in formal verification, has been studied a lot for register automata, see *e.g.* [6, 2, 5]. For GRA, it is well known that the problem is undecidable [2]. More detailed, the special case of deciding whether a single given GRA  $\mathcal{B}$  over  $\Sigma$  and  $\mathbb{D}$  accepts the set  $(\Sigma \times \mathbb{D})^*$  of all data words, is undecidable, even if  $\mathcal{B}$  only uses a single register<sup>2</sup>.

In this short note, we prove that the containment problem  $L(\mathcal{A}) \subseteq L(\mathcal{B})$  is decidable in EXPSPACE if  $\mathcal{B}$  is unambiguous and uses a single register (and no restriction on  $\mathcal{A}$ ).

## 2 Main Definitions

In this section, we define *register automata with guessing* as introduced by Kaminski and Zeitlin [4].

We start with some preliminary notions. We use  $\Sigma$  to denote a finite alphabet, and  $\mathbb{D}$  to denote an infinite data domain. A *data word* is a finite sequence  $(\sigma_1, d_1) \dots (\sigma_k, d_k) \in (\Sigma \times \mathbb{D})^*$ . We use  $\varepsilon$  to denote the empty data word. A *data language* is a set of data words. We use  $\text{data}(w)$  to denote the set  $\{d_1, \dots, d_k\}$  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  $d \in \mathbb{D}$ . We use boldface lower-case letters like  $\mathbf{a}, \mathbf{b}, \dots$  to denote tuples in  $\mathbb{D}_\perp^n$ , where  $n \in \mathbb{N}$ . Given a tuple  $\mathbf{a} \in \mathbb{D}_\perp^n$ , we write  $a_i$  for its  $i$ -th component, and  $\text{data}(\mathbf{a})$  denotes the set  $\{a_1, \dots, a_n\} \subseteq \mathbb{D}_\perp$  of all data occurring in  $\mathbf{a}$ .

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

$$\phi ::= \text{true} \mid = r_i \mid \dot{r}_i = r_j \mid \dot{r}_i = \mid \neg \phi \mid \phi \wedge \phi$$

where  $r_i, r_j \in R$ . Intuitively,  $r_i$  refers to the current value of the register  $r_i$ , while  $\dot{r}_i$  refers to the future value of the register  $r_i$ . We use  $\Phi(R)$  to denote the set of all register operations over  $R$ . The satisfaction relation  $\models$  on  $\mathbb{D}_\perp^R \times \mathbb{D} \times \mathbb{D}_\perp^R$  is defined by structural induction as follows. We only give the atomic cases; the cases for the Boolean formulas are as usual. We have  $(\mathbf{u}, d, \mathbf{v}) \models \phi$  if

- $\phi$  is of the form **true**,
- $\phi$  is of the form  $= r_i$  and  $u_i = d$ ,

<sup>1</sup>In Theorem 12 in [1], it is claimed that the class of data languages accepted by GURA is effectively closed under complement; however, to the best of our knowledge, this claim remains unproved.

<sup>2</sup>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 [2]. The nondeterministic guessing can be used to express that there exists some decrement for which there is no matching preceding increment.

- $\phi$  is of the form  $\dot{r}_i = r_j$  and  $v_i = u_j$ ,
- $\phi$  is of the form  $\dot{r}_i =$  and  $v_i = d$ .

For instance,  $(1, 2, 1) \models (\neq r) \wedge (\dot{r} = r)$ , while  $(1, 2, 3)$  does not. Note that only register constraints of the form  $\dot{r}_i = r_j$  and  $\dot{r} =$  uniquely determine the new value of  $r_i$ . In absence of such a register constraint, the new value of  $r_i$  can be equal to (almost<sup>3</sup>) any of the infinitely many data values in  $\mathbb{D}$ . Register automata that allow for such nondeterministic *guessings* of future register values are called *register automaton with guessing*. Formally, a register automaton with guessing (GRA) over  $\Sigma$  is a tuple  $\mathcal{A} = (R, \mathcal{L}, \ell_{\text{in}}, \mathcal{L}_{\text{acc}}, E)$ , where

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

A *state* of  $\mathcal{A}$  is a pair  $(\ell, \mathbf{u}) \in \mathcal{L} \times \mathbb{D}_{\perp}^R$ , where  $\ell$  is the current location and  $\mathbf{u}$  is the current register valuation. Given two states  $(\ell, \mathbf{u})$  and  $(\ell', \mathbf{u}')$  and some input letter  $(\sigma, d) \in \Sigma \times \mathbb{D}$ , we postulate a transition  $(\ell, \mathbf{u}) \xrightarrow{\sigma, d}_{\mathcal{A}} (\ell', \mathbf{u}')$  if there exists some edge  $(\ell, \sigma, \phi, \ell') \in E$  such that  $(\mathbf{u}, d, \mathbf{u}') \models \phi$ . A *run* of  $\mathcal{A}$  on the data word  $(\sigma_1, d_1) \dots (\sigma_k, d_k)$  is a sequence  $(\ell_0, \mathbf{u}^0) \xrightarrow{\sigma_1, d_1}_{\mathcal{A}} (\ell_1, \mathbf{u}^1) \xrightarrow{\sigma_2, d_2}_{\mathcal{A}} \dots \xrightarrow{\sigma_k, d_k}_{\mathcal{A}} (\ell_n, \mathbf{u}^k)$  of such transitions. We say that a run like above *starts in*  $(\ell, \mathbf{u})$  if  $(\ell_0, \mathbf{u}^0) = (\ell, \mathbf{u})$ . A run is *initialized* if starts in  $(\ell_{\text{in}}, \{\perp\}^k)$ , and a run is *accepting* if  $\ell_k \in \mathcal{L}_{\text{acc}}$ . The data language *accepted* by  $\mathcal{A}$ , denoted by  $L(\mathcal{A})$ , is the set of data words for which there exists an initialized, accepting run of  $\mathcal{A}$ . A GRA is *unambiguous* (GURA) if for every input data word  $w$  there is at most one initialized accepting run. The *containment problem* is the following decision problem: given two GRA  $\mathcal{A}$  and  $\mathcal{B}$ , does  $L(\mathcal{A}) \subseteq L(\mathcal{B})$  hold?

### 3 Some Facts about Register Automata

#### 3.1 Unambiguous Register Automata with Guessing

Fix a GURA  $\mathcal{B} = (R, \mathcal{L}, \ell_{\text{in}}, \mathcal{L}_{\text{acc}}, E)$  with a single register  $r$ . Let  $C \subseteq (\mathcal{L} \times \mathbb{D}_{\perp})$  be a set of states of  $\mathcal{B}$ , and let  $(\sigma, d) \in (\Sigma \times \mathbb{D})$ . We use  $\text{Succ}_{\mathcal{B}}(C, (\sigma, d))$  to denote the *successor of  $C$  on the input  $(\sigma, d)$* , formally defined by

$$\text{Succ}_{\mathcal{B}}(C, (\sigma, d)) := \{(\ell, u) \in (\mathcal{L} \times \mathbb{D}_{\perp}) \mid \exists (\ell', u') \in C. (\ell', u') \xrightarrow{\sigma, d}_{\mathcal{B}} (\ell, u)\}.$$

In order to extend this definition to data words, we define inductively  $\text{Succ}_{\mathcal{B}}(C, \varepsilon) := C$  and  $\text{Succ}_{\mathcal{B}}(C, w \cdot (\sigma, d)) := \text{Succ}_{\mathcal{B}}(\text{Succ}_{\mathcal{B}}(C, w), (\sigma, d))$ . We say that a set  $C \subseteq (\mathcal{L} \times \mathbb{D})$  of states is *reachable in  $\mathcal{B}$*  if there exists some data word  $w$  such that  $C = \text{Succ}_{\mathcal{B}}(C_{\text{in}}, w)$ , where  $C_{\text{in}} = \{(\ell_{\text{in}}, \perp)\}$ .

A *configuration of  $\mathcal{B}$*  is a finite union of finite or cofinite subsets of  $\mathcal{L} \times \mathbb{D}_{\perp}$ . Hence the set  $C_{\text{in}} := \{(\ell_{\text{in}}, \perp)\}$  is a configuration, henceforth called the *initial configuration*. Note that for all configurations  $C$  and data words  $w$ , the successor  $\text{Succ}_{\mathcal{B}}(C, w)$  is a configuration, too. This implies that every reachable set  $C \subseteq (\mathcal{L} \times \mathbb{D})$  of states is a configuration. Given a configuration  $C$ , we use  $\text{data}(C)$  to denote the set  $\{d \in \mathbb{D}_{\perp} \mid \exists \ell \in \mathcal{L}. (\ell, d) \in C\}$  of data occurring in  $C$ .

The *support* of a configuration  $C$  is the set  $\text{supp}(C)$  of data  $d$  such that at least one of the following holds:

---

<sup>3</sup>The register constraint  $\dot{r} \neq$  requires that the new value of  $r$  is different from the current input datum, so that  $r$  may take any of the infinitely data in  $\mathbb{D}$  except for the input datum. Likewise, the register constraint  $\dot{r}_i \neq r_j$  requires that  $r_i$  takes any of the infinitely data in  $\mathbb{D}$  except for the current value of  $r_j$ .

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

We say that a configuration  $C$  is *coverable* if there exists some configuration  $C' \supseteq C$  such that  $C'$  is reachable in  $\mathcal{B}$ . We say that a configuration  $C$  is *accepting* if there exists  $(\ell, u) \in C$  such that  $\ell \in \mathcal{L}_{\text{acc}}$ ; otherwise we say that  $C$  is *non-accepting*. The following proposition follows immediately from the definition of GURA.

**Proposition 1.** *If  $C, C'$  are two configurations of  $\mathcal{B}$  such that  $C \cap C' = \emptyset$  and  $C \cup C'$  is coverable, then for every data word  $w$  the following holds: if  $\text{Succ}_{\mathcal{B}}(C, w)$  is accepting, then  $\text{Succ}_{\mathcal{B}}(C', w)$  is non-accepting.*

A *partial isomorphism of  $\mathbb{D}_{\perp}$*  is an injective mapping  $\pi : D \rightarrow \mathbb{D}_{\perp}$  with domain  $\text{dom}(\pi) := D \subseteq \mathbb{D}$  such that if  $\perp \in D$  then  $\pi(\perp) = \perp$ . Let  $\pi$  be a partial isomorphism of  $\mathbb{D}_{\perp}$  and let  $C$  be a configuration such that  $\text{data}(C) \subseteq \text{dom}(\pi)$ . We define the configuration  $\pi(C) := \{(\ell, \pi(d)) \mid (\ell, d) \in C\}$ ; likewise, if  $\{d_1, \dots, d_k\} \subseteq \text{dom}(\pi)$ , we define the data word  $\pi(w) = (\sigma_1, \pi(d_1)) \dots (\sigma_k, \pi(d_k))$ . We say that  $C, w$  and  $C', w'$  are *equivalent with respect to  $\pi$* , written  $C, w \sim_{\pi} C', w'$ , if  $\pi(C) = C'$  and  $\pi(w) = w'$ . If  $w = w' = \varepsilon$ , we may write  $C \sim_{\pi} C'$ . We write  $C, w \sim C', w'$  if  $C, w \sim_{\pi} C', w'$  for some partial isomorphism  $\pi$  of  $\mathbb{D}_{\perp}$ .

**Proposition 2.** *If  $C, w \sim C', w'$ , then  $\text{Succ}(C, w) \sim \text{Succ}(C', w')$ .*

As an immediate consequence of Proposition 2, we obtain that  $\sim$  preserves the configuration properties of being *accepting* respectively *non-accepting*.

**Corollary 3.** *If  $C, w \sim C', w'$  and  $\text{Succ}_{\mathcal{B}}(C, w)$  is non-accepting (accepting, respectively), then  $\text{Succ}_{\mathcal{B}}(C', w')$  is non-accepting (accepting, respectively).*

Combining the last corollary with Proposition 1, we obtain

**Corollary 4.** *If  $C, C'$  are two configurations such that  $C \cap C' = \emptyset$  and  $C \cup C'$  is coverable in  $\mathcal{B}$ , then for every data word  $w$  such that  $C, w \sim C', w$ , the configurations  $\text{Succ}_{\mathcal{B}}(C, w)$  and  $\text{Succ}_{\mathcal{B}}(C', w)$  are non-accepting.*

### 3.2 The Synchronized State Space

For the rest of this paper, let  $\mathcal{A} = (R^{\mathcal{A}}, \mathcal{L}^{\mathcal{A}}, \ell_{\text{in}}^{\mathcal{A}}, \mathcal{L}_{\text{acc}}^{\mathcal{A}}, E^{\mathcal{A}})$  be a GNRA over  $\Sigma$  with  $R^{\mathcal{A}} = \{r_1, \dots, r_m\}$ , and let  $\mathcal{B} = (R^{\mathcal{B}}, \mathcal{L}^{\mathcal{B}}, \ell_{\text{in}}^{\mathcal{B}}, \mathcal{L}_{\text{acc}}^{\mathcal{B}}, E^{\mathcal{B}})$  be a GURA over  $\Sigma$  with a single register  $r$ . A *synchronized configuration of  $\mathcal{A}$  and  $\mathcal{B}$*  is a pair  $((\ell, \mathbf{d}), C)$ , where  $(\ell, \mathbf{d}) \in (\mathcal{L}^{\mathcal{A}} \times \mathbb{D}_{\perp}^{R^{\mathcal{A}}})$  is a single state of  $\mathcal{A}$ , and  $C \subseteq (\mathcal{L}^{\mathcal{B}} \times \mathbb{D}_{\perp})$  is a configuration of  $\mathcal{B}$ . We define  $S_{\text{in}} := ((\ell_{\text{in}}^{\mathcal{A}}, \{\perp\}^m), C_{\text{in}})$  to be the *initial synchronized configuration of  $\mathcal{A}$  and  $\mathcal{B}$* . We define the *synchronized state space of  $\mathcal{A}$  and  $\mathcal{B}$*  to be the (infinite) state-transition system  $(\mathbb{S}, \Rightarrow)$ , where  $\mathbb{S}$  is the set of all synchronized configurations of  $\mathcal{A}$  and  $\mathcal{B}$ , and  $\Rightarrow$  is defined as follows. If  $S = ((\ell, \mathbf{d}), C)$  and  $S' = ((\ell', \mathbf{d}'), C')$ , then  $S \Rightarrow S'$  if there exists a letter  $(\sigma, d) \in (\Sigma \times \mathbb{D})$  such that  $(\ell, \mathbf{d}) \xrightarrow{\sigma, d}_{\mathcal{A}} (\ell', \mathbf{d}')$ , and  $\text{Succ}_{\mathcal{B}}(C, (\sigma, d)) = C'$ . We say that a synchronized configuration  $S$  *reaches a synchronized configuration  $S'$  in  $(\mathbb{S}, \Rightarrow)$*  if there exists a path in  $(\mathbb{S}, \Rightarrow)$  from  $S$  to  $S'$ . We say that a synchronized configuration  $S$  is *reachable in  $(\mathbb{S}, \Rightarrow)$*  if  $S_{\text{in}}$  reaches  $S$ . We say that a synchronized configuration  $S = ((\ell, \mathbf{d}), C)$  is *coverable in  $(\mathbb{S}, \Rightarrow)$*  if there exists some configuration  $C' \supseteq C$  such that  $((\ell, \mathbf{d}), C')$  is reachable in  $(\mathbb{S}, \Rightarrow)$ .

We aim to reduce the containment problem  $L(\mathcal{A}) \subseteq L(\mathcal{B})$  to a reachability problem in  $(\mathbb{S}, \Rightarrow)$ . For this, call a synchronized configuration  $((\ell, \mathbf{d}), C)$  *bad* if  $\ell \in \mathcal{L}_{\text{acc}}^{\mathcal{A}}$  is an accepting location and  $C$  is non-accepting, i.e.,  $\ell' \notin \mathcal{L}_{\text{acc}}^{\mathcal{B}}$  for all  $(\ell', u) \in C$ . The following proposition is easy to prove, cf. [7].

**Proposition 5.**  *$L(\mathcal{A}) \subseteq L(\mathcal{B})$  does not hold if, and only if, some bad synchronized configuration is reachable in  $(\mathbb{S}, \Rightarrow)$ .*

We extend the equivalence relation  $\sim$  defined above to synchronized configurations in a natural manner, i.e, given a partial isomorphism  $\pi$  of  $\mathbb{D}_{\perp}$  such that  $\text{data}(\mathbf{d}) \cup \text{data}(C) \subseteq \text{dom}(\pi)$ , we define

$((\ell, \mathbf{d}), C) \sim_\pi ((\ell, \mathbf{d}'), C')$  if  $\pi(C) = C'$  and  $\pi(\mathbf{d}) = \mathbf{d}'$ . We shortly write  $S \sim S'$  if there exists a partial isomorphism  $\pi$  of  $\mathbb{D}_\perp$  such that  $S \sim_\pi S'$ . Clearly, an analogon of Proposition 2 holds for this extended relation. In particular, we have the following:

**Proposition 6.** *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  $S'$ .*

Note that  $(\mathbb{S}, \Rightarrow)$  is infinite so that *a priori* it is not clear how to exploit Proposition 5 to solving the containment problem. First of all,  $(\mathbb{S}, \Rightarrow)$  is not finitely branching: for every synchronized configuration  $S = ((\ell, \mathbf{d}), C)$  in  $\mathbb{S}$ , every input datum  $d \in \mathbb{D}$  and every *guessed* new value of the registers may give rise to its own individual synchronized configuration  $S_d$  such that  $S \Rightarrow S_d$ . However, it is well known that, using standard techniques, one can define an *abstract* finite-branching state-transition system that is bisimilar to  $(\mathbb{S}, \Rightarrow)$  with respect to  $\sim$ , cf. [5]. Second, and potentially more harmful, the data needed to define the configuration  $C$  in a synchronized configuration  $((\ell, \mathbf{d}), C)$  can grow unboundedly. As an example, consider the GURA in Figure 1. For every  $k \geq 1$ , the configuration  $\{(\ell_1, d) \mid d \in \mathbb{N} \setminus \{d_1, \dots, d_k\}\} \cup \{(\ell_2, d_k)\}$  with pairwise distinct data values  $d_1, \dots, d_k$  is reachable by inputting the data word  $d_1 \dots d_k$ . In the next section, we prove that one can solve the reachability problem from Proposition 5 by focussing on a subset of configurations of  $\mathcal{B}$  that can be defined by a bounded number of data. The approach follows the ideas presented in [5] for unambiguous register automata (without guessing); however, the main technical proposition in [5] does not apply to GRA and is substituted by Proposition 9 below.

## 4 Decidability of the Containment Problem

### 4.1 Bounding the Size of the Supports

Recall the equivalence relation  $\sim$  on  $k$ -tuples, where for  $\mathbf{a}, \mathbf{b} \in \mathbb{D}_\perp^k$  we have  $\mathbf{a} \sim \mathbf{b}$  if there exists a partial isomorphism  $\pi$  of  $\mathbb{D}_\perp$  such that  $\pi(\mathbf{a}) = \mathbf{b}$ . Note that this equivalence relation has finitely many equivalence classes for all  $k \in \mathbb{N}$ . As an example, for  $k = 3$  the equivalence classes of triples are the classes of  $(d_0, d_0, d_0), (d_0, d_0, d_1), (d_0, d_1, d_0), (d_1, d_0, d_0), (d_0, d_1, d_2)$ , where  $d_0, d_1, d_2$  are pairwise distinct data values.

Let  $S = ((\ell, \mathbf{d}), C)$  be a synchronized configuration, and let  $a, b \in \text{supp}(C)$  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  $a, b \notin \text{data}(\mathbf{d})$  and  $\{\ell \in \mathcal{L} \mid (\ell, a) \in C\} = \{\ell \in \mathcal{L} \mid (\ell, b) \in C\}$ .

Let  $S$  be a synchronized configuration  $((\ell, \mathbf{d}), C)$  and let  $a, b \in \text{supp}(C) \setminus \text{data}(\mathbf{d})$ .

Given a configuration  $C$ , we define for every datum  $d \in \mathbb{D}$  the sets

$$C_d^+ := \{(\ell, d) \in \mathcal{L} \times \{d\} \mid (\ell, d) \in C \text{ and } \text{data}(C \cap (\{\ell\} \times \mathbb{D})) \text{ is finite}\}, \text{ and} \\ C_d^- := \{(\ell, d) \in \mathcal{L} \times \{d\} \mid (\ell, d) \notin C \text{ and } \text{data}(C \cap (\{\ell\} \times \mathbb{D})) \text{ is infinite}\}$$

For later reference, we state the following simple fact.

**Fact 7.**  $C \cap C_d^- = \emptyset$ , for all configurations  $C$  and data  $d \in \mathbb{D}$ .

**Example 1.** Let  $C = \{(\ell_1, 0), (\ell_1, 1)\} \cup \{(\ell_2, d) \mid d \in \mathbb{N} \setminus \{1, 2\}\} \cup \{(\ell_3, d) \mid d \in \mathbb{N} \setminus \{0, 1\}\}$ . 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 8.** *Let  $C$  be an essentially coverable configuration, and let  $b \in \text{supp}(C)$ . Then  $((C \setminus C_b^+) \cup C_b^-)$  is essentially coverable, too.*

*Proof.* Let  $(\ell, c), (\ell', c') \in ((C \setminus C_b^+) \cup C_b^-)$ . If  $(\ell, c), (\ell', c') \in C \setminus C_b^+$ , 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

$e \in \mathbb{D} \setminus \{b\}$  such that  $(\ell, e), (\ell', e) \in C$ . Note that such a value  $e$  must exist, as by definition of  $C_b^-$ , the sets  $\text{data}((\{\ell\} \times \mathbb{D}) \cap C)$  and  $\text{data}((\{\ell'\} \times \mathbb{D}) \cap C)$  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  $\{(\ell, e), (\ell', e)\} \subseteq \text{Succ}((\ell_{\text{in}}, \perp), w)$ . Let  $\pi$  be any partial isomorphism satisfying  $\pi(e) = b$  and whose domain contains  $\text{data}(w)$ . Clearly,  $\{(\ell, b), (\ell', b)\} \subseteq \text{Succ}((\ell_{\text{in}}, \perp), \pi(w))$ , and hence  $\{(\ell, b), (\ell', b)\}$  is coverable. Finally, suppose  $(\ell, c) \in C \setminus C_b^+$  and  $(\ell', c') \in C_b^-$ . The proof that  $\{(\ell, c), (\ell', c')\}$  is coverable is very similar to the proof for the preceding case and left as an exercise.  $\square$

**Proposition 9.** *Let  $S = ((\ell^A, \mathbf{d}), C)$  be a synchronized configuration of  $\mathcal{A}$  and  $\mathcal{B}$  such that  $C$  is essentially coverable, and let  $a \neq b$  be such that  $a, b \in \text{supp}(C)$  and  $a \equiv_S b$ .  $S$  reaches a bad configuration in  $(S, \Rightarrow)$  if, and only if,  $S' := ((\ell^A, \mathbf{d}), (C \setminus C_b^+) \cup C_b^-)$  reaches a bad configuration in  $(S, \Rightarrow)$ .*

*Proof.* ( $\Leftarrow$ ) Suppose there exists some data word  $w$  such that there exists an accepting run of  $\mathcal{A}$  on  $w$  that starts in  $(\ell^A, \mathbf{d})$ , and  $\text{Succ}_{\mathcal{B}}(C \setminus C_b^+ \cup C_b^-, w)$  is non-accepting. We assume in the following that  $\text{Succ}_{\mathcal{B}}(C_b^+, w)$  is accepting; otherwise we are done. Let  $(\ell^+, b) \in C_b^+$  be the unique state such that  $\text{Succ}_{\mathcal{B}}((\ell^+, b), w)$  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  $a' \in \mathbb{D}$  such that  $a' \notin \text{data}(w) \cup \text{supp}(C) \cup \text{data}(\mathbf{d})$ . Let  $\pi$  be the isomorphism defined by  $\pi(a) = a'$ ,  $\pi(a') = a$ , and  $\pi(d) = d$  for all  $d \in \mathbb{D}_{\perp} \setminus \{a, a'\}$ . Then  $(\ell^A, \mathbf{d}), w \sim_{\pi} (\ell^A, \mathbf{d}), \pi(w)$  (as  $a \notin \text{data}(\mathbf{d})$  by  $a \equiv_S b$ ), and  $(\ell^+, b), w \sim_{\pi} (\ell^+, b), \pi(w)$ . By Corollary 3, there exists an accepting run of  $\mathcal{A}$  on  $\pi(w)$  that starts in  $(\ell^A, \mathbf{d})$ , and  $\text{Succ}_{\mathcal{B}}((\ell^+, b), \pi(w))$  is accepting. We prove that  $\text{Succ}_{\mathcal{B}}((\ell, c), \pi(w))$  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 Proposition 1,  $\text{Succ}_{\mathcal{B}}((\ell, c), \pi(w))$  must be non-accepting. Second, let  $(\ell, c) \in C_b^-$ . But then  $c = b$ , and hence  $(\ell, c), w \sim_{\pi} (\ell, c), \pi(w)$ . By assumption,  $\text{Succ}_{\mathcal{B}}((\ell, c), w)$  is non-accepting, so that by Corollary 3,  $\text{Succ}_{\mathcal{B}}((\ell, c), \pi(w))$  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  $\text{supp}(C) \cup \text{data}(w) \cup \text{data}(\mathbf{d})$ , we obtain a data word that guides  $S$  to a bad synchronized configuration. Formally, pick some datum  $b' \notin \text{data}(w) \cup \text{supp}(C) \cup \text{data}(\mathbf{d})$ , and let  $\pi$  be the isomorphism defined by  $\pi(b) = b'$ ,  $\pi(b') = b$ , and  $\pi(d) = d$  for all  $d \in \mathbb{D}_{\perp} \setminus \{b, b'\}$ . Note that  $\pi(w)$  does not contain any  $a$ 's or  $b$ 's. Clearly,  $(\ell^A, \mathbf{d}), w \sim_{\pi} (\ell^A, \mathbf{d}), \pi(w)$ . By Corollary 3, there still exists an accepting run of  $\mathcal{A}$  on  $\pi(w)$  that starts in  $(\ell^A, \mathbf{d})$ . We prove that  $\text{Succ}_{\mathcal{B}}(C, \pi(w))$  is non-accepting. Let  $(\ell, c) \in C$ . We distinguish three cases.

1. Let  $c \notin \{b, b'\}$ . Then  $(\ell, c), w \sim_{\pi} (\ell, c), \pi(w)$ . Since  $\text{Succ}_{\mathcal{B}}((\ell, c), w)$  is non-accepting by assumption, so that by Corollary 3 also  $\text{Succ}_{\mathcal{B}}((\ell, c), \pi(w))$  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 Corollary 4 we obtain that  $\text{Succ}_{\mathcal{B}}((\ell, c), \pi(w))$  is non-accepting.
3. Let  $c = b'$ . Note that  $(\ell, b), w \sim_{\pi} (\ell, b'), \pi(w)$ . Recall that  $b' \notin \text{supp}(C)$ . This implies that  $\text{data}(C \cap (\{\ell\} \times \mathbb{D}_{\perp}))$  is cofinite. We distinguish two cases.
  - $b \in \text{data}(C \cap (\{\ell\} \times \mathbb{D}_{\perp}))$ , i.e.,  $(\ell, b) \in C$ . But note that  $(\ell, b) \notin C_b^+$  by cofiniteness of  $\text{data}(C \cap (\{\ell\} \times \mathbb{D}_{\perp}))$ . Hence  $(\ell, b) \in C \setminus \{(\ell^+, b)\}$ .
  - $b \notin \text{data}(C \cap (\{\ell\} \times \mathbb{D}_{\perp}))$ , i.e.,  $(\ell, b) \in C_b^-$ .

In both cases, we have proved above that  $\text{Succ}_{\mathcal{B}}((\ell, b), w)$  is non-accepting. By  $(\ell, b), w \sim_{\pi} (\ell, b'), \pi(w)$  and Corollary 3,  $\text{Succ}_{\mathcal{B}}((\ell, b'), \pi(w))$  is non-accepting, too.

Altogether we have proved that  $\text{Succ}_{\mathcal{B}}(C, \pi(w))$  is non-accepting, while there exists some accepting run of  $\mathcal{A}$  on  $\pi(w)$  starting in  $(\ell^A, \mathbf{d})$ . 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  $(\ell^A, \mathbf{d})$ , and  $\text{Succ}_{\mathcal{B}}(C, w)$  is non-accepting. We assume in the following that  $\text{Succ}_{\mathcal{B}}(C \setminus C_b^+ \cup C_b^-, w)$  is accepting; otherwise we are done. Let  $(\ell^-, b)$  be a state in  $C_b^-$  such that  $\text{Succ}_{\mathcal{B}}((\ell^-, b), w)$  is accepting. Pick some datum  $a' \in \mathbb{D}_{\perp}$  such that  $a' \notin \text{data}(w) \cup \text{supp}(C) \cup \text{data}(\mathbf{d})$ . Let  $\pi$  be the

isomorphism defined by  $\pi(b) = a$ ,  $\pi(a) = a'$ ,  $\pi(a') = b$ , and  $\pi(d) = d$  for all  $d \in \mathbb{D} \setminus \{a, b, a'\}$ . Clearly,  $(\ell^A, \mathbf{d}), w \sim_\pi (\ell^A, \mathbf{d}), \pi(w)$ , so that by Corollary 3, there exists some accepting run of  $\mathcal{A}$  on  $\pi(w)$  starting in  $(\ell^A, \mathbf{d})$ . We prove that  $\text{Succ}_{\mathcal{B}}(C \setminus C_b^+ \cup C_b^-, \pi(w))$  is non-accepting. Let  $(\ell, c) \in C \setminus C_b^+ \cup C_b^-$ . 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  $(\ell, b), w \sim_\pi (\ell, a), \pi(w)$ . Note that  $(\ell, b) \neq (\ell^-, b)$  by Fact 7. By assumption,  $\text{Succ}_{\mathcal{B}}((\ell, b), w)$  is non-accepting. By Corollary 4,  $\text{Succ}_{\mathcal{B}}((\ell, a), \pi(w))$  is non-accepting, too.
2. Let  $c \neq a$ . Note that also  $(\ell^-, b), w \sim_\pi (\ell^-, a), \pi(w)$ . Recall that  $\text{Succ}_{\mathcal{B}}((\ell^-, b), w)$  is accepting. By Corollary 3,  $\text{Succ}_{\mathcal{B}}((\ell^-, a), \pi(w))$  is accepting. We prove below that  $\{(\ell^-, a), (\ell, c)\}$  is coverable. Proposition 1 then implies that  $\text{Succ}_{\mathcal{B}}((\ell, c), \pi(w))$  is non-accepting.

Recall that  $\text{data}((\{\ell^-\} \times \mathbb{D}) \cap C)$  is cofinite. Pick some datum  $d \in \mathbb{D} \setminus \{c\}$  such that  $(\ell^-, d) \in C$ . We distinguish two cases.

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

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

## 4.2 The Algorithm

When a synchronized configuration  $S'$  is obtained from some essentially coverable synchronized configuration  $S = ((\ell, \mathbf{d}), C)$  by applying Proposition 9 to two distinct data values  $a, b \in \text{supp}(C)$ , we say that  $S$  *collapses to*  $S'$ . We say that  $S$  is *maximally collapsed* if one cannot find two distinct data values  $a, b \in \text{supp}(C)$  that satisfy the assumptions of Proposition 9. Note that, by Proposition 8, the synchronized configuration  $S'$  in Proposition 9 is again essentially coverable. By iterating Proposition 9, one obtains that an essentially coverable synchronized configuration reaches a bad synchronized configuration if, and only if, it collapses in finitely many steps to a maximally collapsed synchronized configuration that also reaches a bad synchronized configuration.

The number of maximally collapsed configurations is asymptotically bounded by  $2^{k \log(k) 2^{|\mathcal{L}|}}$ . Indeed, a maximally collapsed configuration  $((\ell, \mathbf{d}'), C)$  can be recovered up to  $\sim$  by:

- The location  $\ell$  and the equivalence class of  $\mathbf{d}'$ ,
- A list  $L_\perp, L_1, \dots, L_k$  of subsets of  $\mathcal{L}$ ,
- A set  $\{L_{k+1}, \dots, L_p\}$  of subsets of  $\mathcal{L}$ ,
- For each location  $\ell \in \mathcal{L}$ , a bit  $b_\ell \in \{0, 1\}$ .

From this, one can constitute a configuration  $S = ((\ell, \mathbf{d}'), C)$  where:

- $\mathbf{d}'$  is an arbitrary tuple in the equivalence class of  $\mathbf{d}$ , using only data from  $\{\perp, 1, \dots, k\}$ ,
- For every  $i \in \{\perp, 1, \dots, k\}$  and  $\ell' \in L_i$ ,  $C$  contains  $(\ell', d'_i)$ ,

- For every  $i \in \{k+1, \dots, p\}$  and  $\ell' \in L_i$ ,  $C$  contains  $(\ell', i)$ ,
- For each  $d \in \mathbb{D} \setminus \{1, \dots, p\}$ ,  $(\ell', d)$  is in  $C$  iff  $b_{\ell'} = 1$ . That is, the bit  $b_{\ell'}$  is set to 1 to indicate that  $(\{\ell'\} \times \mathbb{D}) \cap C$  is cofinite.

Thus, one can bound the number of maximally covered configurations by  $|\mathcal{L}| \times k^k \times (k+1)2^{|\mathcal{L}|} \times 2^{2^{|\mathcal{L}|}} \times 2^{|\mathcal{L}|}$  which is asymptotically  $2^{k \log(k) 2^{|\mathcal{L}|}}$ . Consider the graph whose vertices are the maximally collapsed synchronized configurations and which contains an edge  $S \rightsquigarrow S'$  iff there exists an  $S''$  such that  $S \Rightarrow S''$  and  $S''$  collapses to  $S'$ . This graph has doubly-exponential size in  $\mathcal{A}$  and  $\mathcal{B}$ , and the relation  $\rightsquigarrow$  can be decided in polynomial space [5]. Thus, one obtains that the reachability problem in this graph can be decided in exponential space, so that the containment problem for 1-register GURA is in EXPSPACE.

**Theorem 10.** *The containment problem  $L(\mathcal{A}) \subseteq L(\mathcal{B})$  is in EXPSPACE, if  $\mathcal{A}$  is a GRA and  $\mathcal{B}$  is an unambiguous GRA with a single register.*

## References

- [1] T. Colcombet. Unambiguity in automata theory. In J. Shallit and A. Okhotin, editors, *Descriptive Complexity of Formal Systems - 17th International Workshop, DCFS 2015, Waterloo, ON, Canada, June 25-27, 2015. Proceedings*, volume 9118 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2015.
- [2] S. Demri and R. Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3), 2009.
- [3] M. Kaminski and N. Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.
- [4] M. Kaminski and D. Zeitlin. Finite-memory automata with non-deterministic reassignment. *International Journal of Foundations of Computer Science*, Volume 21, Issue 05, 2010.
- [5] A. Mottet and K. Quaas. The containment problem for unambiguous register automata. In R. Niedermeier and C. Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPIcs*, pages 53:1–53:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.
- [6] F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004.
- [7] J. Ouaknine and J. 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.
- [8] K. Quaas and M. Shirmohammadi. Synchronizing data words for register automata. *ACM Trans. Comput. Log.*, 20(2):11:1–11:27, 2019.