Universality and Forall-Exactness of Cost Register Automata with Few Registers

Laure Daviaud[1] and Andrew Ryzhikov[2]

[1]School of Computing Sciences, University of East Anglia, UK
[2]Department of Computer Science, University of Oxford, UK

Abstract

The universality problem asks whether a given finite state automaton accepts all the input words. For quantitative models of automata, where input words are mapped to real values, this is naturally extended to ask whether all the words are mapped to values above (or below) a given threshold. This is known to be undecidable for commonly studied examples such as weighted automata over the positive rational (plus-times) or the integer tropical (min-plus) semirings, or equivalently cost register automata (CRAs) over these semirings. In this paper, we prove that when restricted to CRAs with only three registers, the universality problem is still undecidable, even with additional restrictions for the CRAs to be copyless linear with resets.

In contrast, we show that, assuming the unary encoding of updates, the ∀-exact problem (does the CRA output zero on all the words?) for integer min-plus linear CRAs can be decided in polynomial time if the number of registers is constant. Without the restriction on the number of registers this problem is known to be PSPACE-complete.

This work has been supported by the EPSRC grant EP/T018313/1. The second author was also partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 852769, ARiAT).

1 Introduction

Cost register automata (CRAs), introduced by Alur et al. [3], are an extension of finite state automata. Instead of just accepting or rejecting words, they assign each word a value, usually from some semiring. This allows to reason about quantitative properties of systems, such as costs, probabilities, or durations. A CRA is a deterministic finite automaton equipped with a finite set of write-only registers which store values from a semiring, and which are combined using the operations of this semiring. The transitions that a CRA takes thus depend only on the input word, and not on the values of the registers, which makes them different to automata with counters (such as Minsky machines), and allows more of their properties to be decidable.

CRAs are tightly related to weighted automata (WAs), a classical computational model which maps words to values from a fixed semiring. In general, WAs are less expressive than CRAs [3]. However, WAs are equally expressive to linear CRAs, which are CRAs where the updates of the registers are restricted to linear transformations. Hence linear CRAs can be seen as a deterministic model for WAs. Transforming a linear CRA into an equivalent WA and vice versa can be done in polynomial time.

WAs extend automata to a quantitative setting and have been extensively studied since their introduction by Schützenberger in [21], see also surveys [11, 12]. Two widely studied models are WAs over the tropical (or min-plus) semiring and probabilistic WAs, a restricted case of WAs over the semiring of rational numbers with usual addition and multiplication (plus-times semiring). They find their applications in such areas as language and speech processing [19], verification [6], image processing [7], and analysis of on-line algorithms [4] and probabilistic systems [23].

Most applications are algorithmic in their nature, that is, require checking some property of the automaton which models a given system. Often these are classical automata-theoretic properties such as universality, emptiness, boundedness, equivalence, etc., generalised to the quantitative setting. For example, the universality problem for non-deterministic finite state automata asks whether all the input words are accepted. It is PSPACE-complete and is solvable in polynomial time for deterministic finite automata [22]. A natural way to generalise this problem to the quantitative setting is to ask whether all the input words are mapped to a value above (or below, depending on the semiring) a certain threshold. For the min-plus semiring $(\mathbb{Z} \cup \{+\infty\}, \min, +)$ (are all the values strictly below 0?), and the plus-times semiring $(\mathbb{Q}, +, \times)$ (are all the values strictly above 1?) this problem is undecidable, see [16, 1] and [20] respectively.

It is thus important to try to find subclasses where this problem becomes decidable. A WA is called linearly ambiguous (respectively, finitely ambiguous) if there is a constant C such that for every word w the number of accepting runs labelled by w is bounded by $C|w|$ (respectively, by C). In both cases (min-plus and plus-times), undecidability is retained even for linearly ambiguous WAs, see [16, 1] and [9] respectively. For finitely ambiguous min-plus WAs universality becomes decidable [24, 13].

Some syntactic restrictions on CRAs allow to introduce subclasses whose expressiveness is incomparable to known classes of WAs. One natural restriction is to bound the number of registers: there exist CRAs with only two registers that compute functions which cannot be computed by a finitely ambiguous WA (Example 2 provides one such CRA). This means that if universality is decidable for CRAs with only two registers, that would allow to decide it for more WAs than it was possible before. We show that for three registers this is not the case, even when restricted further to copyless linear CRAs with resets (see next section for a formal definition), leaving the two-register case as the only remaining option. Undecidability for five registers in the min-plus case follows from a construction in [2], which we use as a starting idea in Subsection 3.1. We note that no characterisation of CRAs with bounded number of registers (or any of its subclasses) in terms of WAs is known, so our results do not follow from any results about WAs.

Informally, a CRA is called copyless if the value of each its registers can only be used once for each transition. In [3], Alur et al. conjectured that universality is decidable for the class of copyless CRAs over the min-plus semiring. In [2], Almagor et al. disproved this conjecture and showed that universality is still undecidable for them. It is natural to ask the same question for the plus-times semiring, but no such results have been known so far. In this paper we show that this conjecture is not true for the plus-times case as well.

CRAs with a bounded number of registers were also studied in the context of register complexity. The register complexity of a function is the minimum number of registers of a CRA that computes it. The problem of computing the register complexity is known to be decidable for unambiguous WA [10], but is open and highly challenging in general. This problem can be seen as a generalisation of the classical determinisation problem, asking if for a given WA there exists an equivalent deterministic one, which amounts to ask whether there exists an equivalent linear CRA with one register [15, 14, 8].

Our contributions. In this paper we prove that for CRAs which are linear with resets, copyless and have only 3 registers universality is still undecidable, both for the min-plus and the plus-times semirings. Our approach gives in fact a more general result, encompassing both semirings at the same time, and proves undecidability of universality under some specific conditions on the semiring. This is an additional advantage of our technique, since usually the proofs for the two mentioned semirings are very different.

Another natural decision problem we consider is the ∀-exact problem, which asks for a given CRA or WA if it outputs zero on all the words. It is known to be PSPACE-complete for WAs (and hence for linear CRAs) over the min-plus semiring [1]. It was also investigated for polynomial automata, which can be seen as a generalisation of WAs over a field [5]. We prove that for this problem bounding the number of registers does help: namely, the ∀-exact problem is solvable in polynomial time for a linear CRA over the min-plus semiring when the number of its registers is a constant, and the updates of registers are given in the unary encoding.

Organisation of the paper. In Section 2, we introduce the model under consideration, namely copyless linear cost-register automata with resets, and the decision problems we study, the universality and the ∀-exact problems. In Sections 3 and 4, we give the proof of undecidability for the universality problem. To make the content more understandable, we do it in two steps: first we explain the main ideas on a specific sub-problem and on a particular semiring in Section 3, and then extend these ideas to give the general proof in Section 4.

2 Cost register automata and decision problems

2.1 Cost register automata

Cost register automata (CRAs) are defined in a general way as deterministic finite automata equipped with so-called registers that can store values (numbers, words...) and be combined with operations (addition, multiplication, minimum, discounted sum, concatenation...). In this paper, we consider a quite restrictive class of CRAs. The undecidability results we obtain for this specific class are then applicable to larger classes and CRAs in general.

Definition 1. A **linear cost register automaton with resets** with k registers over a semiring $(\mathbb{K}, \oplus, \otimes)$ is a deterministic finite automaton (Q, Σ, δ, s) equipped with registers $r_1, \ldots, r_k$ taking values in $\mathbb{K}$, where Q is a finite set of states and s the unique initial state, Σ the alphabet and δ the transition function. The initial values of the registers are specified by a vector $\lambda \in \mathbb{K}^k$ and the output function at a state q is specified as (a linear transformation) $\oplus_{j=1}^{k} (m_j^{(q)} \otimes r_j)$ or (a constant) $m^{(q)}$, where $m_1^{(q)}, \ldots, m_k^{(q)}, m^{(q)}$ are elements of $\mathbb{K}$. Finally, each transition of δ is equipped with k transformations, one for every register, each of one of the two forms:

$$r_i \leftarrow \bigoplus_{j=1}^{k} m_{i,j} \otimes r_j \text{ (a linear transformation)},$$

$$r_i \leftarrow m_{i,k+1} \text{ (a reset to a constant)}$$

with $m_{i,j} \in \mathbb{K}$ for all $1 \le i \le k$, $1 \le j \le k+1$.

The semantics of a CRA is defined by means of valuations of the registers. A valuation σ of the registers is a function $\{r_1, \ldots, r_k\} \to \mathbb{K}$. A run on a word $a_1 a_2 \ldots a_n$, where $a_i \in \Sigma$ for all i, is a sequence: $\rho = (q_1, \sigma_1) \xrightarrow{a_1} (q_2, \sigma_2) \xrightarrow{a_2} \ldots \xrightarrow{a_n} (q_{n+1}, \sigma_{n+1})$ where $q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \ldots \xrightarrow{a_n} q_{n+1}$ is a run in the underlying deterministic finite automaton, and the valuations of the registers $\sigma_1, \ldots, \sigma_{n+1}$ are updated according to the transitions: for all $1 \le \ell \le n$, $\sigma_{\ell+1}(r_i) = \bigoplus_{j=1}^{k} m_{i,j} \otimes \sigma_\ell(r_j)$ if the transition update is given by a linear transformation and $\sigma_{\ell+1}(r_i) = m_{i,k+1}$ if it is given by a reset to a constant.

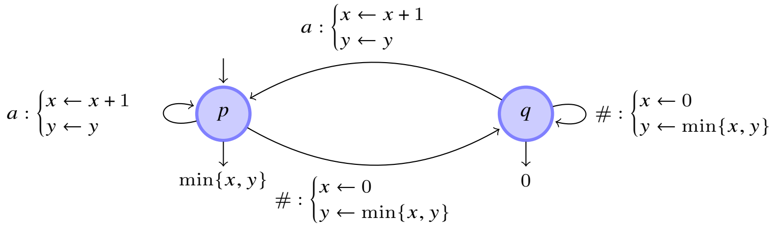
The run is accepting if additionally $q_1 = s$ and $\sigma_1(r_i) = \lambda_i$ for all i, and the output on $a_1 a_2 \ldots a_n$ is given by the output function at state q_{n+1}, i.e. $\oplus_{j=1}^{k} m_j^{(q_{n+1})} \otimes \sigma_{n+1}(r_j)$ if this is given by a linear transformation and $m^{(q_{n+1})}$ if this is given by a constant. Finally, a CRA is called **copyless** if for each transition, every register appears at most once on the right hand-side of the updates: for all j, $m_{i,j}$ is the zero of $\mathbb{K}$, except for at most one i.

All CRAs considered in this paper are from this restrictive class of copyless linear CRAs with resets, so unless specified otherwise by CRAs we mean CRAs from this class. Given a CRA $\mathcal{A}$, we will use $\mathcal{A}$ to denote both the CRA itself and the function $\Sigma^* \to \mathbb{K}$ it computes.

Example 2. In Figure 1, we give an example of a CRA over the semiring $(\mathbb{Z} \cup \{+\infty\}, \min, +)$ and alphabet $\{a, \#\}$. It has two states p (which is initial) and q, and two registers x and y. The output function at state p (words ending with a and the empty word) is $\min\{x, y\}$ and at state q (words ending with #) is 0. Both registers are initialised with value 0.

If the input word ends with # or is empty, this CRA outputs 0. If it ends with a, the CRA outputs the length of the shortest maximal blocks of consecutive a's. Register x stores the number of a's read in the current block, and y stores the minimum length of the blocks read so far. This CRA is copyless linear with resets.

For any semiring, copyless linear CRAs with resets are at most as expressive as linearly ambiguous WAs, strictly less expressive than polynomially ambiguous WAs, and are incomparable to unambiguous WAs [8]. For the min-plus semiring specifically, they are strictly less expressive than linearly ambiguous WAs [2], incomparable to unambiguous WAs [2] and, at the same time, there exists a copyless CRA (but not linear with resets) with 3 registers which is not equivalent to any polynomially ambiguous WA [17].



$a : \begin{cases} x \leftarrow x+1 \\ y \leftarrow y \end{cases}$
$a : \begin{cases} x \leftarrow x+1 \\ y \leftarrow y \end{cases}$
p
q
$\# : \begin{cases} x \leftarrow 0 \\ y \leftarrow \min\{x,y\} \end{cases}$
$\min\{x,y\}$
$\# : \begin{cases} x \leftarrow 0 \\ y \leftarrow \min\{x,y\} \end{cases}$
0

Figure 1: An example of CRA on the semiring $(\mathbb{Z} \cup \{+\infty\}, \min, +)$.

2.2 Decision problems

We are mainly interested in two classes of CRAs: CRAs over $(\mathbb{Z} \cup \{+\infty\}, \min, +)$, called min-plus CRAs, and over $(\mathbb{Q}, +, \times)$, called plus-times CRAs in this paper. Our main result is that the universality problem for these two classes, where the number of registers is restricted to 3, is undecidable. We define this problem as follows:

Universality problem for 3-register CRAs
Input: A min-plus (respectively, plus-times) CRA $\mathcal{A}$ with 3 registers.
Output: Yes if and only if for all words w, we have $\mathcal{A}(w) < 0$ (respectively, $\mathcal{A}(w) > 1$).

Theorem 3. *The universality problem for* 3*-register CRAs is undecidable both for min-plus CRAs and for plus-times CRAs.*

To prove this result we give a reduction from the halting problem for Minsky machines with 2 counters, which is known to be undecidable. This leads us to prove a slightly more general result, encompassing both the min-plus and plus-times cases at the same time.

To simulate the behaviour of a Minsky machine with a CRA, we encode the run of the Minsky machine into a specific word and use the CRA to check that a given word corresponds to a correct encoding of the halting run. This boils down to checking some regular properties and verifying that the counters are updated accordingly to the transitions. To explain our approach of this later part, we start by looking at a simpler problem in Section 3: checking that $n_1 = n_2 = \ldots = n_p$ for a word of the shape $a^{n_1}\sharp a^{n_2}\sharp \ldots \sharp a^{n_{p-1}}\sharp\sharp a^{n_p}$. We explain how to do this with only 3 registers in the min-plus case. In Section 4, we give the full proof of Theorem 3 in a more general setting, allowing us to apply it to both min-plus and plus-times.

On the other hand, we also give a more positive result in the min-plus case: the $\forall$-exact problem, as defined below, is known to be PSPACE-complete for linear CRAs with a non-restricted number of registers [1, Theorem 6.13]. We show that it becomes solvable in polynomial time when the number of registers is fixed (and is not a part of the input). For a fixed number of registers, we assume that the size of a CRA is given by its number of states plus the largest absolute value of an integer appearing in an update of a transition or in the output function at a state. Hence, we assume the unary encoding of the numbers in the input. Without the restriction on the number of registers the $\forall$-exact problem remains PSPACE-complete in this case [1]. Note that for this results, and for this result only, we consider the class of linear CRAs instead of the class of copyless linear CRAs with resets.

The $\forall$-exact problem for k-register min-plus linear CRAs
Input: A min-plus linear CRA $\mathcal{A}$ with k registers.
Output: Yes if and only if for all words w, we have $\mathcal{A}(w) = 0$.

Theorem 4. *For a fixed k, the $\forall$-exact problem is decidable in polynomial time for k-register min-plus linear CRAs, assuming that the numbers in the transformations are given in the unary encoding.*

This result comes from a variation of a pumping argument, showing that for all words to have value 0, the (useful) values of the registers have to be bounded (below and above) by a constant that is polynomial in the size of the CRA. One can then just keep track of these values. The full proof of this result can be found in the full version of the paper.

Variants. The proofs given in this paper can be easily adapted to obtain the undecidability of other variants of the universality problem: for plus-times, whether $\mathcal{A}(w) > c$ or $\mathcal{A}(w) < c$, and for min-plus, whether $\mathcal{A}(w) < c$ for any constant c, and the polynomial-time complexity of variants of the $\forall$-exact problem: for min-plus, whether $\mathcal{A}(w) = c$ for any given constant c. Note that in some cases (for min-plus), a direct translation between these problems is possible and preserves the number of registers. In others (plus-times), the natural translation between these problems would increase the number of registers by 1, but an adaptation of the proofs given in this paper would maintain this number to 3.

3 Recognising equal-length blocks

To explain our approach, we first look at a simpler problem: given a word

$$w = a^{n_1}\sharp a^{n_2}\sharp \dots \sharp a^{n_{p-1}}\sharp a^{n_p},$$

check with a min-plus CRA if all n_i are equal. We assume that the CRA knows that it is going to read the last block of a's. For example, that can be done by duplicating the last $\sharp$ symbol in the word, but we omit this technical detail for now. More precisely:

Proposition 5. *There exist a min-plus CRA $\mathcal{A}$ over the alphabet $\{a, \sharp\}$ with 3 registers such that $\mathcal{A}(w) \le 0$ for all words w, and $\mathcal{A}(w) = 0$ if and only if $w = (a^n\sharp)^m\sharp a^n$ for some integers n, m.*

We describe the solution to this problem in an incremental way, starting from a CRA with 5 registers and then using more and more complex ideas to get to 4 and then finally to only 3 registers. We call a maximal subword of consecutive a's a *block*. For $1 \le i \le p-1$, define $w_i = a^{n_i}\sharp a^{n_{i+1}}$.

3.1 Five registers

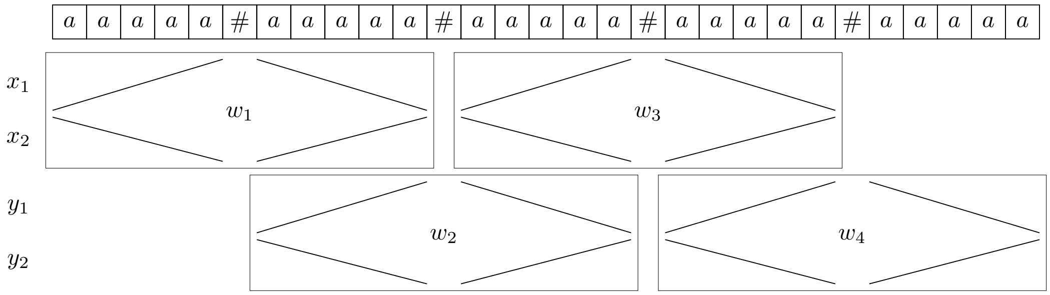


Figure 2: The computations of two pairs of registers processing w_i, $1 \le i \le 4$. An increasing line means that the register is incremented by 1 for each letter and a decreasing line that the register is decremented by 1 for each letter.

The case of five registers is easy and uses the idea described in [2]. To test if for a word $w_i = a^{n_i}\sharp a^{n_{i+1}}$ we have $n_i = n_{i+1}$, we use two registers, call them x_1 and x_2, initialised to zero. While reading the first of two blocks, x_1 is incremented by one, and x_2 is decremented by one. While reading the second block, they do the opposite. Thus, after reading w_i, the value of x_1 is $n_i - n_{i+1}$, and the value of x_2 is $-n_i + n_{i+1}$. Observe that $n_i = n_{i+1}$ if and only if $\min\{n_i - n_{i+1}, -n_i + n_{i+1}\} = 0$. Moreover, $\min\{n_i - n_{i+1}, -n_i + n_{i+1}\}$ is always non-positive. We introduce a new special register z. After reading w_i, we set $z \longleftarrow \min\{z, x_1, x_2\}$ and reset x_1 and x_2 to zero. Hence the value of z is then the minimum of its previous value and $\min\{n_i - n_{i+1}, -n_i + n_{i+1}\}$.

After doing that, we have already read the second block of w_i, so we cannot do the same for w_{i+1} using the same registers. However, we can cover all words w_i by using two pairs of registers: one pair x_1, x_2 for w_i with i odd, and another pair y_1, y_2 for i even, because each pair is reset to zero at the end of reading w_{i+1}, and hence can be used for processing w_{i+2}. Register z is shared between these two pairs. Figure 2 illustrates the computations performed by the pairs x_1, x_2 and y_1, y_2.

The register z is initialised with zero at the beginning, and changes its value as described above. The output of the CRA is then defined as $\min\{z, x_1, x_2\}$ if i is odd, and $\min\{z, y_1, y_2\}$ if i is even. This value is the minimum of the values $\{n_i - n_{i+1}, -n_i + n_{i+1} \mid 1 \le i \le p-1\}$. It is equal to zero if and only if all n_i are equal, for $1 \le i \le p$, otherwise it is strictly negative.

3.2 Four registers

The case of four registers is handled similarly, but now we want to get rid of the register z, and accumulate the non-positive value $\min\{n_i - n_{i+1}, -n_i + n_{i+1}\}$ in one of the registers that we use for its computation.

Once again, we use two pairs x_1, x_2 and y_1, y_2 of registers to separately process w_i for odd and even values of i, and before reading the input we initialise them all with zeros. Assume that i is odd, and hence we use x_1, x_2 to process the word $w_i = a^{n_i}\sharp a^{n_{i+1}}$. When processing w_i, the registers x_1 and x_2 perform the same computations as in the case of five registers, thus computing $n_i - n_{i+1}$ and $-n_i + n_{i+1}$ respectively. However, after that, instead of sending these values to z and resetting both registers to zero, we reset to zero only x_1, and set $x_2 \leftarrow \min\{x_1, x_2\}$. Let m_i be the value of x_2 after processing w_i if $i \geq 1$, and zero otherwise. We can show by induction that m_i is always non-positive and 0 if and only if $n_j = n_{j+1}$ for all odd $j < i$. Indeed, after processing w_i the value of x_2 is $\min\{n_i - n_{i+1}, -n_i + n_{i+1} + m_{i-2}\}$. By the induction hypothesis, it is always non-positive, and is 0 if and only if $n_i - n_{i+1} = -n_i + n_{i+1} = 0$ and $m_{i-2} = 0$, which concludes the argument. For even i, we do similarly for y_1, y_2.

After reading w, the CRA then outputs the value $\min\{x_1, x_2, y_1, y_2\}$. As explained above, this value is zero if and only if for all i we have $n_i = n_{i+1}$, which means that all blocks have the same length. Otherwise this value is strictly negative.

3.3 Three registers

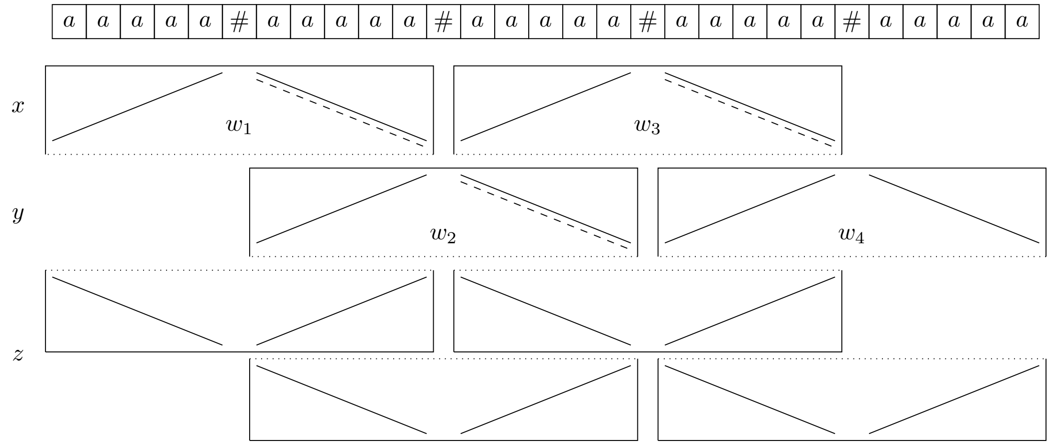


Figure 3: The computations of the registers processing w_i, $1 \leq i \leq 4$. An increasing line means that the register is incremented by 1 for each letter and a decreasing line – whether plain or dash – that the register is decremented by 1 for each letter. When both a plain and a dashed line are present it means that the increment/decrement is by 2. The two last lines are for register z, and hence z is only incremented/decremented in the first and last block – the increasing and decreasing lines cancel each other but we show them to explain the idea used for the general construction. Note that additionally after reading each block the value of the register z is changed (not shown in the picture). The dotted sides of the boxes represent that the corresponding boxes from Figure 2 were cut in halves and reassigned to different registers.

Idea. For the case of three registers we use the idea described for four registers, but now we only leave the first register of each pair (call these registers x and y), and use the third register (call it z) to act simultaneously as the second register from both pairs. Since in the case of four registers the computations performed by two pairs overlap, this will require some adjustments for the behaviour of x and y to deal with the overlapping parts.

We use register x to process $w_i = a^{n_i}\sharp a^{n_{i+1}}$ for odd values of i, and y for even values of i. Assume for example that i is odd. The idea remains the same: we want x to compute $n_i - n_{i+1}$, and z to compute $-n_i + n_{i+1}$. However, if we want z to perform such computations for all w_i, then z is also involved first in processing w_{i-1} and then in processing w_{i+1}. The solution is to consider this unwanted change for z and to make the same change to x, which then allows to meaningfully compare the values of x and z despite the overlaps. Refer to Figure 3 for the symbolic depiction of these computations. The dashed lines illustrate the additional decrements made by x and y to adjust for overlap with y and x respectively. After processing each w_i, we set $z \leftarrow \min\{z, x\}$ if i is odd, and $z \leftarrow \min\{z, y\}$ if i is even.

The output of the CRA after reading w is $\min\{z, x\}$ if i is odd, and $\min\{z, y\}$ if i is even. Below we show how to construct the CRA implementing this idea.

Implementation. As mentioned above, we separate the cases of odd and even i, and process each w_i with only one register. At the beginning all registers are initialised with zeros. When processing w_i, $1 \leq i \leq p-2$, we compute the value $n_i - 2n_{i+1}$ by incrementing the corresponding register by one when reading each letter of a^{n_i}, and decrementing it by two when reading each letter of $a^{n_{i+1}}$. After processing w_i, we set $z \leftarrow \min\{z, x\}$ if i is odd, and $z \leftarrow \min\{z, y\}$ if i is even. We reset to zero the corresponding used register x or y. When processing w_{p-1} by x or y, we compute the value $n_{p-1} - n_p$ instead.

The description of the computation is almost done. While reading a^{n_1}, we decrement z by one for each letter a, and while reading a^{n_p} we increment it by one for each reading of a (we use here the fact that we know in advance when we are going to read the last block). The output of the CRA is defined as the minimum of the values of the register processing w_{p-1} (which has the value $n_{p-1} - n_p$ at the end) and z. Figure 4 illustrates the computations of thus constructed CRA.

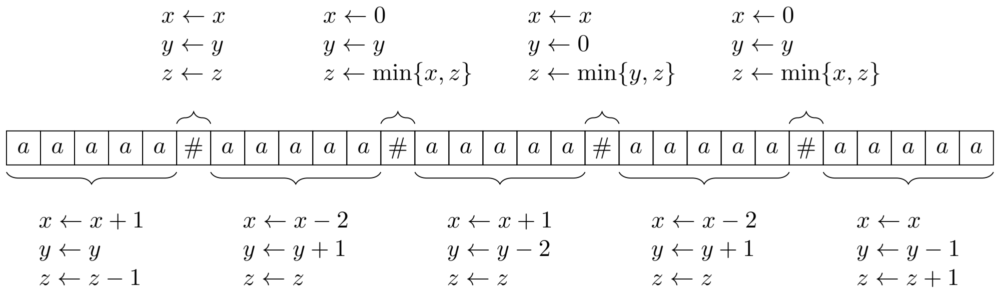


Figure 4: The output value is $\min\{y, z\}$.

Lemma 6. *The output of thus constructed CRA is zero if and only if all blocks of w have the same length, otherwise it is strictly negative.*

Proof. After processing w_1, register x has value $n_1 - 2n_2$, and register z has value $-n_1$. Then x is reset to 0, and its value is passed to z, hence the value of z is $\min\{-n_1, n_1 - 2n_2\} = -n_2 + \min\{-n_1 + n_2, n_1 - n_2\} = -n_2 + C_1$, where we define $C_1 = \min\{n_1 - n_2, -n_1 + n_2\}$.

For $2 \leq i \leq p-2$, define $C_i = \min\{n_i - n_{i+1}, -n_i + n_{i+1} + C_{i-1}\}$. Then inductively after reading $a^{n_{i+1}}$ for $2 \leq i \leq p-2$ the corresponding register finishes processing w_i and passes its value to z, which then has value

$$\min\{n_i - 2n_{i+1}, -n_i + C_{i-1}\} = -n_{i+1} + \min\{n_i - n_{i+1}, -n_i + n_{i+1} + C_{i-1}\} = -n_{i+1} + C_i.$$

Moreover, after reading a^{n_p} the output of the CRA is $\min\{n_{p-1} - n_p, -n_{p-1} + n_p + C_{p-2}\}$, which we accordingly denote by C_{p-1}. Note that even though the shape of the formula for C_{p-1} is the same as for C_i with $i \leq p-2$, the way this value is computed by the CRA is different, since while reading a^{n_p} both z and the register processing w_{p-1} behave differently than before.

To show that the constructed CRA satisfies the requirements, we inductively analyse the values $C_1, \ldots, C_{p-1}$. As noted above, $C_1 = 0$ if and only if $n_1 = n_2$, otherwise $C_1 < 0$. Moreover, $C_i = 0$ if and only if $n_i = n_{i+1}$ and $C_{i-1} = 0$, otherwise $C_i < 0$. Hence the output $C_{p-1} = 0$ if and only if all blocks have the same length, and $C_{p-1} < 0$ otherwise. □

With a similar proof, we can show that the same result applies to plus-times CRA (see end of Section 4 for a general scheme).

4 Simulating a Minsky machine with a CRA with 3 registers

In this section, we show how to, given a Minsky machine $\mathcal{M}$ with two counters, construct a CRA $\mathcal{A}$ with three registers which simulates $\mathcal{M}$. For min-plus, this will mean that $\mathcal{A}$ outputs 0 on the (unique) word encoding the halting run of $\mathcal{M}$ if it exists, and outputs a strictly negative value for all other words. For plus-times, $\mathcal{A}$ outputs 1 on the (unique) word encoding the halting run of $\mathcal{M}$ if it exists, and outputs a value strictly greater than 1 for all other words. This will prove Theorem 3 as the halting problem for Minsky machine with two counters is undecidable. We will in fact prove a more general result, extending both the min-plus and the plus-times cases. This result is given in Theorem 8.

4.1 Minsky machines

Let $P = \cup_{1 \le i \le 2}\{\mathrm{inc}_i, \mathrm{dec}_i, \mathrm{test}_i\}$ be a set of operations (increments, decrements and tests for zero of the ith counter) on two counters. A Minsky machine $\mathcal{M}$ with 2 counters is a deterministic finite automaton over the alphabet P, such that there is a designated *initial* state, and the transitions satisfy the following restrictions: for each state q, exactly one of the following holds

- q has exactly one outgoing transition, which is then labelled by inc_i for some $i \in \{1, 2\}$,
- q has exactly two outgoing transitions, which are then labelled respectively by $\mathrm{dec}_i, \mathrm{test}_i$ for the same $i \in \{1, 2\}$,
- q has no outgoing transition, in which case it is a unique state called the *halting state.*

Let $\mathcal{M}$ be a Minsky machine with 2 counters. Consider an alternating sequence

$$\rho = (q_1, \mathbf{v}_1) \xrightarrow{t_1} (q_2, \mathbf{v}_2) \xrightarrow{t_2} \dots \xrightarrow{t_{p-1}} (q_p, \mathbf{v}_p)$$

of pairs $(q_i, \mathbf{v}_i)$, where each q_i is a state and each $\mathbf{v}_i$ is a pair of non-negative integers, and of operations $t_i \in P$. The pairs $\mathbf{v}_i$ represent the values of the two counters, and we denote by $\mathbf{v}_i[j]$ its jth component. Such sequence ρ is called a *halting run* of $\mathcal{M}$ if $q_1 \xrightarrow{t_1} q_2 \xrightarrow{t_2} \dots \xrightarrow{t_{p-1}} q_p$ is a run in the underlying DFA of $\mathcal{M}$, q_1 is the initial state, q_p is the halting state, $\mathbf{v}_1 = (0, 0)$, and for each i, $1 \le i \le p - 1$, we have the following (where $\mathbf{e}_1 = (1, 0)$ and $\mathbf{e}_2 = (0, 1)$):

- if $t_i = \mathrm{inc}_j$, then $\mathbf{v}_{i+1} = \mathbf{v}_i + \mathbf{e}_j$;
- if $t_i = \mathrm{test}_j$, then $\mathbf{v}_i[j] = 0$ and $\mathbf{v}_{i+1} = \mathbf{v}_i$;
- if $t_i = \mathrm{dec}_j$, then $\mathbf{v}_i[j]$ is positive and $\mathbf{v}_{i+1} = \mathbf{v}_i - \mathbf{e}_j$.

Theorem 7 ([18], Theorem 14.1-1)**.** *The problem whether a given Minsky machine with two counters has a halting run is undecidable.*

4.2 Encoding a run of a Minsky machine

To construct a CRA simulating a given Minsky machine with two counters, we first specify how to represent a run of a Minsky machine by a finite word. For now on, fix a Minsky machine $\mathcal{M}$.

Encoding the values of two counters as one number. If $\mathbf{v}$ is a vector of two components which are non-negative integer numbers, define $\nu(\mathbf{v}) = 2^{\mathbf{v}[1]} \times 3^{\mathbf{v}[2]}$. We use $\nu(\mathbf{v})$ to encode the values of two counters of $\mathcal{M}$ in a given moment of time. Note that since 2 and 3 are coprime, there is at most one vector $\nu^{-1}(x)$ for a natural number x, so this encoding is injective.

Encoding the runs. Given a halting run $\rho = (q_1, \mathbf{v}_1) \xrightarrow{t_1} \dots \xrightarrow{t_{p-1}} (q_p, \mathbf{v}_p)$ of $\mathcal{M}$, we construct the word $W(\rho)$ over the alphabet $\Sigma = P \cup \{a, \sharp\}$, where

$$W(\rho) = t_1 a^{\nu(\mathbf{v}_1)} t_2 a^{\nu(\mathbf{v}_2)} t_3 \dots t_{p-1} a^{\nu(\mathbf{v}_{p-1})} \sharp a^{\nu(\mathbf{v}_p)}$$

We say that the word $W(\rho)$ *encodes* the halting run ρ. Note that each letter t_i appears before the blocks encoding the values of $\mathbf{v}_i$ (the argument of t_i) and $\mathbf{v}_{i+1}$ (the result of t_i), so that the CRA knows which operation to expect before starting to check that this operation was applied correctly. Note also that the new symbol $\sharp$ announces the last block of a's. We call the word w over Σ encoding the halting run of $\mathcal{M}$ the *run word.*

Regular tests for run words. To test if a given word w over Σ is a run word, we first check if this word has shape

$$t_1 a^{n_1} t_2 a^{n_2} t_3 \dots t_{p-1} a^{n_{p-1}} \sharp a^{n_p}$$

for some positive integers $p, n_1, \dots, n_p$, and for some $t_1, t_2, \dots, t_{p-1}$ in P. We then check that there exist states $q_1, q_2, \dots, q_p$ of $\mathcal{M}$ such that for each $1 \leq i \leq p-1$ there is a transition $q_i \xrightarrow{t_i} q_{i+1}$ in $\mathcal{M}$, and also that q_1 is the initial and q_p is the halting state of $\mathcal{M}$.

Next, we check that if $t_i = \mathrm{test}_\ell$ for $\ell \in \{1, 2\}$, we have that the ℓth component of $\nu^{-1}(a^{n_i})$ is 0. To do so, we simply check that n_i, the length of a^{n_i}, is not divisible by 2 (respectively, 3) if $\ell = 1$ (respectively, $\ell = 2$). Finally, to test that the values of the counters in the beginning are 0, we simply check that $n_1 = 1$.

It is easy to see all these checks are regular, hence they can be performed by the underlying deterministic finite automaton of a CRA, that is, without using any registers. We call a word satisfying these regular properties a *pre-run word.*

What remains to check is that the values of the counters (encoded by ν) change according to the corresponding operations of $\mathcal{M}$. This check is the main challenge if we want to perform it by a CRA with a small number of registers. The next section describes how to deal with it.

4.3 Handling operations

Let $\mathcal{M}$ be a Minsky machine, and w be a pre-run word. We now describe how to construct a CRA $\mathscr{A}$ with three registers which checks that w is a run word. The idea of this construction is similar to the idea described in Section 3. The main difference is that now instead of testing that all blocks of a's have the same length, we need to test that these lengths are changed according to the corresponding operations.

We give a general construction, that we will be able to apply later on to min-plus and plus-times CRAs and prove the following result, where all the notions will be introduced in due course:

Theorem 8. *Let $\mathcal{M}$ be a Minsky machine and let $(\mathbb{K}, \oplus, \otimes, \preccurlyeq)$ be an ordered semiring with a multiplicative group, such that:*

- *there exists a linear transformation which is a $\mathbf{1}$-peak (where $\mathbf{1}$ is the identity element of $\otimes$), and*
- *there exists an element with no finite order.*

Then one can construct a CRA $\mathscr{A}$ over $(\mathbb{K}, \oplus, \otimes)$ such that:

- *$\mathscr{A}(w) \preccurlyeq \mathbf{1}$ for all words w, and*
- *$\mathscr{A}(w) = \mathbf{1}$ if and only if w encodes the halting run of $\mathcal{M}$.*

We consider a semiring $(\mathbb{K}, \oplus, \otimes)$. It is said to have a multiplicative group if $(\mathbb{K} - \{\mathbf{0}\}, \otimes)$ is a group, where $\mathbf{0}$ is the identity element of $\oplus$. We denote by $\mathbf{1}$ the identity element of this group. We also denote, for an element d of the group and some positive integer ℓ, by d^ℓ the product of d by itself ℓ times, and by $d^{-\ell}$ its inverse. By convention $d^0 = \mathbf{1}$. We also fix a linear transformation $\varphi : \mathbb{K}^2 \to \mathbb{K}$, i.e. $\varphi(x, y) = (c \otimes x) \oplus (d \otimes y)$ for some c, d in $\mathbb{K}$, and we fix an element α of $\mathbb{K}$.

We will now construct a CRA over the semiring $(\mathbb{K}, \oplus, \otimes)$ using α and φ.

Let $w = t_1 a^{n_1} t_2 a^{n_2} t_3 \dots t_{p-1} a^{n_{p-1}} \sharp a^{n_p}$ be a pre-run input word. For convenience of the presentation, we denote $t_p = \sharp$. As before, we call a maximal continuous subword of w consisting of a's a *block.* Denote $w_i = t_i a^{n_i} t_{i+1} a^{n_{i+1}}$. Call x, y, z the three registers of $\mathscr{A}$. We process w_i with x for odd values of i, and with y for even values of i, and send the results to z.

First we describe what $\mathscr{A}$ does when reading a^{n_i} for $1 < i < p$. The word a^{n_i} is the second block in w_{i-1} and the first block in w_i. Assume without loss of generality that i is odd (otherwise switch x and y in the further description). Then w_i is processed by x, and w_{i-1} is processed by y. The computations performed by x depend only on t_i: for each occurrence of a, we set

- $x \leftarrow x \otimes \alpha$ if $t_i \in \{\mathrm{test}_\ell, \mathrm{dec}_\ell\}$ for $\ell \in \{1, 2\}$,
- $x \leftarrow x \otimes \alpha^2$ if $t_i = \mathrm{inc}_1$,
- $x \leftarrow x \otimes \alpha^3$ if $t_i = \mathrm{inc}_2$.

The computations performed by y depend on both t_{i-1} and t_i. Let c be the value (1, 2 or 3) used above in the transformation $x \leftarrow x \otimes \alpha^c$ depending on t_i. Then for each occurrence of a we set:

- $y \leftarrow y \otimes \alpha^{-1} \otimes \alpha^{-c}$ if $t_{i-1} \in \{\mathrm{test}_\ell, \mathrm{inc}_\ell\}$ for $\ell \in \{1, 2\}$,
- $y \leftarrow y \otimes \alpha^{-2} \otimes \alpha^{-c}$ if $t_{i-1} = \mathrm{dec}_1$,
- $y \leftarrow y \otimes \alpha^{-3} \otimes \alpha^{-c}$ if $t_{i-1} = \mathrm{dec}_2$.

Furthermore, if, for each letter of a^{n_i}, register x is multiplied by α^c and y is multiplied by $\alpha^{-d} \otimes \alpha^{-c}$, then we set $z \leftarrow z \otimes \alpha^{-c} \otimes \alpha^d$ for each letter of a^{n_i}. After reading t_{i+2}, the next letter after a^{n_i}, we set $z \leftarrow \varphi(y, z)$, and reset y to $\mathbf{1}$.

It remains to describe the transformations of the registers for a^{n_1} and a^{n_p}. For each letter of a^{n_1}, x performs the same update as described above, and z is updated by the inverse of that value. Assume without loss of generality that $p - 1$ is even (otherwise use x instead of y in the further description). Then w_{p-1} is processed by y. For each letter in a^{n_p} we set

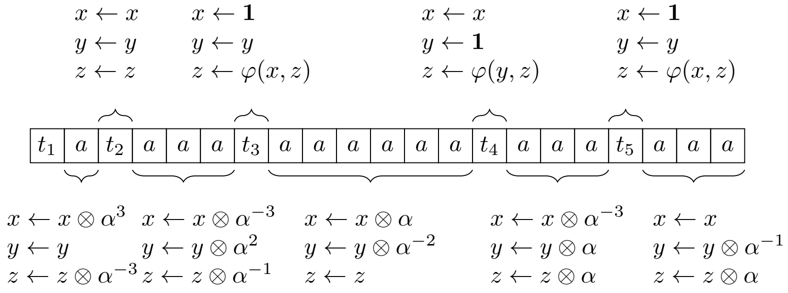
- $y \leftarrow y \otimes \alpha^{-1}$, $z \leftarrow z \otimes \alpha$ if $t_{p-1} \in \{\mathrm{test}_\ell, \mathrm{inc}_\ell\}$ for $\ell \in \{1, 2\}$,
- $y \leftarrow y \otimes \alpha^{-2}$, $z \leftarrow z \otimes \alpha^2$ if $t_{p-1} = \mathrm{dec}_1$,
- $y \leftarrow y \otimes \alpha^{-3}$, $z \leftarrow z \otimes \alpha^3$ if $t_{p-1} = \mathrm{dec}_2$.

Finally, we set the output of $\mathcal{A}$ to be $\varphi(y, z)$. Note that in this construction, the CRA is linear with resets and copyless.

Example 9. Consider the following halting run:

$$\rho = (q_1, (0, 0)) \xrightarrow{\mathrm{inc}_2} (q_1, (0, 1)) \xrightarrow{\mathrm{inc}_1} (q_1, (1, 1)) \xrightarrow{\mathrm{dec}_1} (q_1, (0, 1)) \xrightarrow{\mathrm{test}_1} (q_1, (0, 1))$$

We abbreviate by t_1, t_2, t_3, t_4 the operations along this run and by t_5 the $\sharp$ symbol indicating that we are going to read the last block. Note again that in w the operation is written before its argument, hence for example $t_1 = \mathrm{inc}_2$ is applied to the first block, and its result is the second block. The computations of $\mathcal{A}$ for the word encoding ρ are shown in Figure 5. The CRA outputs $\varphi(y, z)$ in this example.



$x \leftarrow x$
$y \leftarrow y$
$z \leftarrow z$
$x \leftarrow \mathbf{1}$
$y \leftarrow y$
$z \leftarrow \varphi(x, z)$
$x \leftarrow x$
$y \leftarrow \mathbf{1}$
$z \leftarrow \varphi(y, z)$
$x \leftarrow \mathbf{1}$
$y \leftarrow y$
$z \leftarrow \varphi(x, z)$
t_1 a t_2 a a a t_3 a a a a a a t_4 a a a t_5 a a a
$x \leftarrow x \otimes \alpha^3$
$y \leftarrow y$
$z \leftarrow z \otimes \alpha^{-3}$
$x \leftarrow x \otimes \alpha^{-3}$
$y \leftarrow y \otimes \alpha^2$
$z \leftarrow z \otimes \alpha^{-1}$
$x \leftarrow x \otimes \alpha$
$y \leftarrow y \otimes \alpha^{-2}$
$z \leftarrow z$
$x \leftarrow x \otimes \alpha^{-3}$
$y \leftarrow y \otimes \alpha$
$z \leftarrow z \otimes \alpha$
$x \leftarrow x$
$y \leftarrow y \otimes \alpha^{-1}$
$z \leftarrow z \otimes \alpha$

Figure 5: The computations of $\mathcal{A}$ on the word encoding the halting run ρ.

Suppose now that $\mathbb{K}$ is equipped with a linear order $\preccurlyeq$, and denote by $\prec$ the associated strict order (we call such $\mathbb{K}$ an *ordered semiring*). We say that α has no finite order if $\alpha^\ell = \mathbf{1}$ implies $\ell = 0$ and that φ is a ***1**-peak* if, for all d, e in $\mathbb{K}$ such that $d \otimes e \preccurlyeq \mathbf{1}$, we have $\varphi(d, e) = \mathbf{1}$ if and only if $d = e = \mathbf{1}$, and otherwise $\varphi(d, e) \prec \mathbf{1}$.

Lemma 10. *Suppose that φ is a $\mathbf{1}$-peak and that α has no finite order. Then, for a pre-run word w, we have $\mathcal{A}(w) = \mathbf{1}$ if and only if w is a run word, otherwise $\mathcal{A}(w) \prec \mathbf{1}$.*

Proof. We adapt the proof of Lemma 6. Define $C_0 = \mathbf{1}$ and for $1 \le i \le p - 1$ define

$$C_i = \begin{cases} \varphi(\alpha^{n_i - n_{i+1}}, \alpha^{-n_i + n_{i+1}} \otimes C_{i-1}\} & \text{if } t_i = \mathrm{test}_\ell \text{ for } \ell = 1, 2 \\ \varphi(\alpha^{2n_i - n_{i+1}}, \alpha^{-2n_i + n_{i+1}} \otimes C_{i-1}\} & \text{if } t_i = \mathrm{inc}_1 \\ \varphi(\alpha^{3n_i - n_{i+1}}, \alpha^{-3n_i + n_{i+1}} \otimes C_{i-1}\} & \text{if } t_i = \mathrm{inc}_2 \\ \varphi(\alpha^{n_i - 2n_{i+1}}, \alpha^{-n_i + 2n_{i+1}} \otimes C_{i-1}\} & \text{if } t_i = \mathrm{dec}_1 \\ \varphi(\alpha^{n_i - 3n_{i+1}}, \alpha^{-n_i + 3n_{i+1}} \otimes C_{i-1}\} & \text{if } t_i = \mathrm{dec}_2 \end{cases}$$

Observe that by construction after a register (x or y) processes w_i, $1 \leq i \leq p-2$, and passes its value to z, the value of z is

$$\begin{cases} \alpha^{-n_{i+1}} \otimes C_i & \text{if } t_{i+1} = \mathrm{test}_\ell \text{ or } t_{i+1} = \mathrm{inc}_\ell \text{ for } \ell \in \{1, 2\} \\ \alpha^{-2n_{i+1}} \otimes C_i & \text{if } t_{i+1} = \mathrm{dec}_1 \\ \alpha^{-3n_{i+1}} \otimes C_i & \text{if } t_{i+1} = \mathrm{dec}_2 \end{cases}$$

Moreover, the output of $\mathcal{A}$ equals C_{p-1}.

Observe that using the definition of C_i and by induction, one can prove that $C_i = \mathbf{1}$ if and only if t_i is performed correctly (that is, the result of t_i on the pair $\nu^{-1}(a^{n_i})$ is $\nu^{-1}(a^{n_{i+1}})$) and $C_{i-1} = \mathbf{1}$, otherwise $C_i \prec \mathbf{1}$, since φ is $\mathbf{1}$-peak and α has no finite order. Hence the output C_{p-1} equals $\mathbf{1}$ if and only if the input is a run word, otherwise it is $\prec \mathbf{1}$. □

By combining this proposition with the checks for pre-run words described in Section 4.2, we get a CRA which outputs $\mathbf{1}$ if and only if the input encodes a halting run of $\mathcal{M}$. Indeed, if a word is not a pre-run word (which is checked by the underlying DFA of the CRA), the run labelled by it will end in a state which outputs a constant value $f \prec \mathbf{1}$. All runs ending in other states are thus labelled by pre-run words, and for each such word the computations described in this section output the desired value depending on whether this word encodes a halting run or not. This means that all runs in the CRA are considered, concluding the proof of Theorem 8.

Min-plus and plus-times cases. To finish the proof of Theorem 3, it is enough to instantiate $(\mathbb{K}, \oplus, \otimes, \preccurlyeq)$, α and φ to suitable elements. For min-plus, we take $(\mathbb{K}, \oplus, \otimes, \preccurlyeq) = (\mathbb{Z} \cup \{+\infty\}, \min, +, \leq)$, $\alpha = 1$ and $\varphi = \min$. For plus-times, we take $(\mathbb{K}, \oplus, \otimes, \preccurlyeq) = (\mathbb{Q}_+, +, \times, \geq)$, $\alpha = 2$ and $\varphi(c, d) = 2^{-1}(c + d)$, where $\mathbb{Q}_+$ is the set of positive rational numbers. It is easy to check that in both cases, φ is $\mathbf{1}$-peak and α has no finite order.

5 Conclusions

In this paper, we prove the undecidability of the universality problem for models of CRAs where the number of registers is limited to 3. Our main result holds for min-plus and plus-times CRAs, but we give a slightly more general construction and it would be interesting to see if our techniques can be applied to other cases, and in particular to see its link to infinitary groups that have already been studied in conjunction to the register complexity [10].

The main open question that remains is whether this is still true when considering CRAs with only 2 registers. Our proof cannot be adapted easily to the 2-register case. One approach is to understand whether with only 2 registers, one can recognise the language with equal length blocks as defined in Section 3. Even this is difficult.

Finally, we proved that the ∀-exact problem is solvable in polynomial time when the number of registers of a min-plus linear CRA is fixed. The same question can be asked for the boundedness problem over the $(\mathbb{N} \cup \{+\infty\}, \min, +)$ semiring, which is known to be PSPACE-complete for WAs (and hence linear CRAs) [1].

References

[1] Shaull Almagor, Udi Boker, and Orna Kupferman. What's decidable about weighted automata? *Information and Computation*, 282:104651, 2022. doi:10.1016/j.ic.2020.104651.

[2] Shaull Almagor, Michaël Cadilhac, Filip Mazowiecki, and Guillermo A. Pérez. Weak cost register automata are still powerful. *International Journal of Foundations of Computer Science*, 31(6):689–709, 2020. doi:10.1142/S0129054120410026.

[3] Rajeev Alur, Loris D'Antoni, Jyotirmoy Deshmukh, Mukund Raghothaman, and Yifei Yuan. Regular functions and cost register automata. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2013)*, pages 13–22, 2013. `doi:10.1109/LICS.2013.65`.

[4] Benjamin Aminof, Orna Kupferman, and Robby Lampert. Reasoning about online algorithms with weighted automata. *ACM Transactions on Algorithms*, 6(2):28:1–28:36, 2010. `doi:10.1145/1721837.1721844`.

[5] Michael Benedikt, Timothy Duff, Aditya Sharad, and James Worrell. Polynomial automata: Zeroness and applications. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2017)*, pages 1–12, 2017. `doi:10.1109/LICS.2017.8005101`.

[6] Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Transactions on Computational Logic*, 11(4):23:1–23:38, 2010. `doi:10.1145/1805950.1805953`.

[7] Karel Culík and Jarkko Kari. Digital images and formal languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages, Volume 3: Beyond Words*, pages 599–616. Springer, 1997. `doi:10.1007/978-3-642-59126-6\_10`.

[8] Laure Daviaud. Register complexity and determinisation of max-plus automata. *ACM SIGLOG News*, 7(2):4–14, 2020. `doi:10.1145/3397619.3397621`.

[9] Laure Daviaud, Marcin Jurdzinski, Ranko Lazic, Filip Mazowiecki, Guillermo A. Pérez, and James Worrell. When are emptiness and containment decidable for probabilistic automata? *Journal of Computer and System Sciences*, 119:78–96, 2021. `doi:10.1016/j.jcss.2021.01.006`.

[10] Laure Daviaud, Pierre-Alain Reynier, and Jean-Marc Talbot. A generalised twinning property for minimisation of cost register automata. In *31st Annual ACM/IEEE Symposium on Logic in Computer Science, (LICS 2016)*, pages 857–866, 2016. `doi:10.1145/2933575.2934549`.

[11] Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer Berlin, Heidelberg, 1st edition, 2009. `doi:10.1007/978-3-642-01492-5`.

[12] Manfred Droste and Dietrich Kuske. Weighted automata. In Jean-Éric Pin, editor, *Handbook of Automata Theory*, pages 113–150. European Mathematical Society Publishing House, Zürich, Switzerland, 2021. `doi:10.4171/Automata-1/4`.

[13] Kosaburo Hashiguchi, Kenichi Ishiguro, and Shuji Jimbo. Decisability of the equivalence problem for finitely ambiguous automata. *International Journal of Algebra and Computation*, 12(03):445–461, 2002. `doi:10.1142/S0218196702000845`.

[14] Daniel Kirsten and Sylvain Lombardy. Deciding Unambiguity and Sequentiality of Polynomially Ambiguous Min-Plus Automata. In *26th International Symposium on Theoretical Aspects of Computer Science (STACS 2009)*, volume 3 of *LIPIcs*, pages 589–600, 2009. `doi:10.4230/LIPIcs.STACS.2009.1850`.

[15] Ines Klimann, Sylvain Lombardy, Jean Mairesse, and Christophe Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theoretical Computer Science*, 327(3):349–373, 2004. `doi:10.1016/j.tcs.2004.02.049`.

[16] Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *International Journal of Algebra and Computation*, 4(3):405–426, 1994. `doi:10.1142/S0218196794000063`.

[17] Filip Mazowiecki and Cristian Riveros. Copyless cost-register automata: Structure, expressiveness, and closure properties. *Journal of Computer and System Sciences*, 100:1–29, 2019. `doi:10.1016/j.jcss.2018.07.002`.

[18] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, USA, 1967.

[19] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997. URL: `https://aclanthology.org/J97-2003`.

[20] Azaria Paz. *Introduction to probabilistic automata*. Academic Press, 1971.

[21] Marcel-Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2):245–270, 1961. `doi:10.1016/S0019-9958(61)80020-X`.

[22] Larry J. Stockmeyer and Albert R. Meyer. Word problems requiring exponential time: Preliminary report. In *5th Annual ACM Symposium on Theory of Computing (STOC 1973)*, pages 1–9, 1973. `doi:10.1145/800125.804029`.

[23] Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *26th Annual Symposium on Foundations of Computer Science (FOCS 1985)*, pages 327–338, 1985. `doi:10.1109/SFCS.1985.12`.

[24] Andreas Weber. Finite-valued distance automata. *Theoretical Computer Science*, 134(1):225–251, 1994. `doi:10.1016/0304-3975(94)90287-9`.