# Weak Cost Register Automata are Still Powerful

Shaull Almagor
*Computer Science Department, Technion*
*Haifa 3200003, Israel*
*shaull@cs.technion.ac.il*

Michaël Cadilhac
*Department of Computer Science, University of Oxford, Parks Road*
*Oxford, OX1 3QD, United Kingdom*
*michael@cadilhac.name*

Filip Mazowiecki
*Laboratoire Bordelais de Recherche en Informatique (UMR 5800) 351 cours de la Libération*
*33405 Talence CEDEX, France*
*filip.mazowiecki@u-bordeaux.fr*

Guillermo A. Pérez
*Universiteit Antwerpen, Campus Middelheim, Middelheimlaan 1*
*2020 Antwerpen, België*
*guillermoalberto.perez@uantwerpen.be*

Received (Day Month Year)
Accepted (Day Month Year)
Communicated by (xxxxxxxxxx)

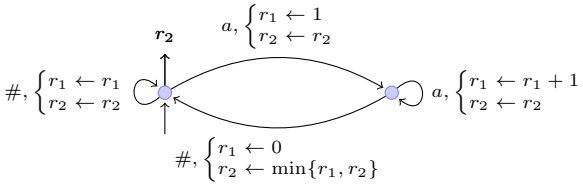
We consider one of the weakest variants of cost register automata over a tropical semiring, namely copyless cost register automata over $\mathbb{N}$ with updates using min and increments. We show that this model can simulate, in some sense, the runs of counter machines with zero-tests. We deduce that a number of problems pertaining to that model are undecidable, namely equivalence, upperboundedness, and semilinearity. In particular, the undecidability of equivalence disproves a conjecture of Alur et al. from 2012. To emphasize how weak these machines are, we also show that they can be expressed as a restricted form of linearly-ambiguous weighted automata.

*Keywords*: Cost register automata; semilinearity; equivalence.

## 1. Introduction

Cost register automata (CRA) [3] encompass a wealth of computation models for functions from words to values (herein, integers). In their full generality, a CRA is simply a DFA equipped with registers that are updated upon taking transitions. The updates are expressions built using a prescribed set of operations (e.g., $+, \times, \min, \ldots$), constants, and the registers themselves.

In this work, we will focus on CRA computing integer values, where the updates may only use "$+c$", for any constant $c$, and min. For instance:




With $r_1$ initialized to 0 and $r_2$ to $\infty$, this CRA computes the length of the minimal nonempty block of $a$'s between two #'s. This model has the same expressive power as weighted automata (WA) over the structure $(\mathbb{Z}, \min, +)$, but the use of registers can simplify the design of functions.

The example above enjoys an extra property that can be used to restrain the model (since a lot of interesting problems are undecidable on WA [1]). Indeed, *no register is used twice in any update function*; this property is called *copylessness*. This syntactic restriction, introduced by Alur et al. [3] and studied by Mazowiecki and Riveros [12], provably weakens the model. It was the hope of Alur et al. that this would provide a model for which equivalence is decidable.

**Semilinearity and decidability of equivalence.** Recall that a set $R \subseteq \mathbb{Z}^k$ is semilinear if it is expressible in first-order logic with addition: FO$[<, +]$. This latter logic being decidable [15], semilinearity is a useful tool to show decidability results. For instance, let $f, g \colon A^* \to \mathbb{Z}$ be expressible in some model for which the images of functions are effectively semilinear. Suppose further that the function $h \colon w \mapsto \min\{2 \times f(w), 2 \times g(w) + 1\}$ is also in that model. Since the image $h(A^*)$ is effectively semilinear, one can check whether it is always even: this would show that $f(w) \leq g(w)$ for all $w$. In this work we investigate, among other things, whether copyless CRA (CCRA) are such a model as well. We show that, somewhat surprisingly, they are not.

**Iterating min breaks semilinearity.** Deterministic automata equipped with copyless registers with only "$+c$" updates are quite well-behaved [7] ; in particular, the set

$$R = \{\overline{r} \mid \overline{r} \text{ are the values of the registers at the end of an accepting run}\}$$

is semilinear. Naturally, $\min\{x, y\}$ is expressible in FO$[<, +]$, hence FO$[<, +]$ = FO$[<, +, \min]$ (even, and this is not immediate, when the extra value $\infty$ is added [8]). This entails that if we were to give to these automata the ability to do a *constant* number of min, we would still have that $R$ is semilinear. In this paper, it is shown that if the number of min is unbounded along runs, then the set is not semilinear (see the proof of Theorem 18 for a simple and self-contained construction), and that it is undecidable to check whether $R$ is semilinear.

**Related work.** The model of cost-register automata was originally introduced as a model providing a different representation of weighted automata [3]. This followed a similar presentation of transducers with a streaming string transducer model [2]. In both cases the authors provided a way to give a characterisation of standard models (weighted automata and transducers) via a deterministic model at the cost of introducing registers that store partial computations. Similar presentations of weighted automata, like CRA, have also been given in [16, 5]. Registers were already used in many models of automata (both deterministic and nondeterministic). The content of registers can be: data [10]; time [4]; or processes [14]. The latter model of Petri nets can be interpreted as vector addition systems, which are particularly interesting in comparison with CRA. When both CRA and vector addition systems are considered over the integers, there are cases when the two models coincide (see the Remark on page 6 in [6]). Namely, vector addition systems do not label their transitions with letters and the decision problem is usually the reachability problem: is there a sequence of transitions between two given configurations? When transitions are labelled, this model can be seen to be equivalent to CRA over integers (with only the + operation), where instead of reachability, the decision problem is nonemptiness: is there an accepting word? As these decision problems are not the topic of this paper, we refer the interested reader to [5] (see e.g. Example 3 and Theorem 6).

**Contributions.** Beyond considerations on semilinearity, we show that CCRA over $\mathbb{N}$ can simulate the runs of counter machines with zero-tests (Theorem 15). Intuitively, the only words mapped by the CCRA to an even value are the correct executions of the counter machine. This construction is then used to show that equivalence is undecidable for CCRA over $\mathbb{N}$ and that upper-boundedness is undecidable for WA. To better gauge the expressiveness of CCRA, we show that they are a weak form of *linearly-ambiguous* WA, that is, WA for which no word $w$ has more than $k \times |w|$ accepting runs, for some constant $k$ (see drawing on page 7). Since the problems we tackle are decidable for *finitely-ambiguous* WA, CCRA are arguably the simplest generalization of deterministic WA for which equivalence is undecidable.

**Paper Structure.** In Section 2 we define the models we use, and other fundamental definition and results. In Section 3 we compare the expressive power of CCRA and weighted automata. In Sections 4 and 5 we show how to simulate VASS using CCRA, over $\mathbb{Z}_\infty$ and $\mathbb{N}_\infty$, respectively. Then, in Section 6, we derive undecidability results. We present some concluding remarks in Section 7.

## 2. Preliminaries

We assume familiarity with automata theory, for which we settle some notations. We write $\mathbb{N}$ for $\{0, 1, 2, \ldots\}$, $\mathbb{Z}$ for the integers, and define $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$ and $\mathbb{Z}_\infty = \mathbb{Z} \cup \{\infty\}$. Naturally, $\min\{\ldots, \infty, \ldots\}$ stays the same when removing the $\infty$

value, and we set $\min \emptyset = \infty$. For any $k \geq 1$, we write $[k]$ for $\{1, 2, \ldots, k\}$. We write $\varepsilon$ for the empty word.

**Automata.** An automaton (NFA) is a tuple $(Q, A, \delta, q_0, F)$, where $Q$ is the set of states, $A$ the alphabet, $\delta \subseteq Q \times (A \cup \{\varepsilon\}) \times Q$ the transition relation, $q_0$ the initial state, and $F \subseteq Q$ the set of final states. We rely on the usual vocabulary pertaining to automata: a *run* is a word in $\delta^*$ starting in $q_0$, and such that each transition is consistent with the next; it is accepting if the last reached state is in $F$; a word $w \in A^*$ is accepted if there is an accepting run labeled by $w$.

If $\delta$ is a function from $Q \times A$ to $Q$, the automaton is *deterministic* (DFA). If there is a $k \in \mathbb{N}$ such that each accepted word $w$ is the label of at most $k \times |w|$ accepting runs, the automaton is *linearly-ambiguous*.

**Tropical Semirings.** A *semiring* is a tuple $(S, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ where $S$ is a set, $\oplus$ and $\otimes$ are associative binary operations, with respective identity elements $\mathbf{0}$ and $\mathbf{1}$, $\oplus$ is commutative, and $\otimes$ distributes over $\oplus$.

Unless explicitly mentioned otherwise, the only semirings we consider in this work are $(\mathbb{Z}_\infty, \min, +, \infty, 0)$ and $(\mathbb{N}_\infty, \min, +, \infty, 0)$, often dubbed "tropical semirings".

When the discussion is not specific to one of the two semirings, we simply write $\mathbb{K}$ for both. As with rings, matrix multiplication is well-defined (and associative) in semirings; e.g., if $(b_{ij})$ and $(c_{ij})$ are $2 \times 2$ matrices and $(a_{ij}) = (b_{ij}) \cdot (c_{ij})$, then:

$$a_{2,1} = \min\{b_{2,1} + c_{1,1}, b_{2,2} + c_{2,1}\} \ .$$

**Weighted automata.** Weighted automata will only be used in Section 3 and Theorem 19. A weighted automaton [a] $\mathcal{W}$ over $\mathbb{K}$ ($\mathbb{K}$-WA) is a tuple $(\mathcal{A}, \lambda, \mu, \nu)$ where $\mathcal{A} = (Q, A, \delta, q_0, F)$ is an NFA, and $\lambda \in \mathbb{K}, \mu \colon \delta \to \mathbb{K}$, and $\nu \colon F \to \mathbb{K}$. Given a run $t_1 \cdot t_2 \cdots t_n \in \delta^*$ ending in a state $q \in F$ in $\mathcal{A}$, its *weight* is $\lambda + \mu(t_1) + \mu(t_2) + \cdots + \mu(t_n) + \nu(q)$. The weight $\mathcal{W}(w)$ of a word $w \in A^*$ is the minimum weight for all accepting runs over $w$ in the NFA (hence it is $\infty$ if the word is not accepted). The $\mathbb{K}$-WA is *deterministic* (resp. *linearly-ambiguous*) if $\mathcal{A}$ is. We use $\mathbb{K}$-DetWA and $\mathbb{K}$-LinWA for these restrictions.

**Registers and counters.** A central goal of this work is to present a simulation of some *counter* machine with zero-tests by a *register* machine without zero-test but with more complicated update functions. To avoid confusion, we will stick to that vocabulary, and use $c_i$ for counters and $r_i$ for registers.

**Cost register automata.** In this work, we only consider cost register automata over $\mathbb{K} \in \{\mathbb{Z}_\infty, \mathbb{N}_\infty\}$ where the registers are updated using expressions that use min and "$+c$" for $c \in \mathbb{K}$.

[a] In general, weighted automata can be defined over any semiring. The definition we bring here applies for tropical semirings.

*Intuitive definition.* A $\mathbb{K}$-cost register automaton $\mathcal{C}$ of dimension $k$ ($\mathbb{K}$-CRA, for short) is a DFA equipped with $k$ registers $r_1, r_2, \ldots, r_k$ taking values in $\mathbb{K}$. The initial values of the registers are specified by a vector $\overline{\lambda}$ in $\mathbb{K}^k$, and each transition is equipped with a transformation $\mu$, referred to as the *update function*, of the form:

$$(\forall i \in [k]) \quad r_i \leftarrow \min\{r_1 + m_{1,i},\ r_2 + m_{2,i},\ \ldots,\ r_k + m_{k,i},\ m_{k+1,i}\}\ ,$$

where each $m_{i,j}$ is in $\mathbb{K}$ (hence it can be $\infty$, making the subexpression irrelevant). Each final state is paired with an *output function* $\nu$ of the shape:

$$\min\{r_1 + m_1,\ r_2 + m_2,\ \ldots,\ r_k + m_k,\ m_{k+1}\}\ ,$$

where again the $m_i$'s are in $\mathbb{K}$.

Given a word $w \in A^*$, the *value* of $\mathcal{C}$ on $w$, written $\mathcal{C}(w)$, is $\infty$ if $w$ is not accepted by the underlying DFA, and otherwise computed in the obvious way: the registers are initialized, then updated along the (single) run in the DFA, and the output is determined by the output function at the final state.

*Formal definition.* The $\mathbb{K}$-CRA $\mathcal{C}$ is a tuple $(\mathcal{A}, \overline{\lambda}, \mu, \nu)$ where $\mathcal{A} = (Q, A, \delta, q_0, F)$ is a DFA, $\overline{\lambda} \in \mathbb{K}^{1\times k}$ is the initial value of the $k$ registers, $\nu\colon F \to \mathbb{K}^{(k+1)\times 1}$ gives the output function for each final state, and $\mu\colon Q \times A \to \mathbb{K}^{(k+1)\times(k+1)}$ provides the update functions. To compare with the intuition given above, and using the notation therein, $\mu(q, a)$ is:

$$\begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,k} & \infty \\ m_{2,1} & m_{2,2} & \cdots & m_{2,k} & \infty \\ \vdots & \vdots & \ddots & \vdots & \\ m_{k,1} & m_{k,2} & \cdots & m_{k,k} & \infty \\ m_{k+1,1} & m_{k+1,2} & \cdots & m_{k+1,k} & 0 \end{pmatrix}$$

It can be readily checked that $(\overline{r}', 0) = (\overline{r}, 0) \cdot \mu(q, a)$ indeed satisfies, for all $i \in [k]$ that:

$$r'_i = \min\{r_1 + m_{1,i},\ r_2 + m_{2,i},\ \ldots,\ r_k + m_{k,i},\ m_{k+1,i}\}\ .$$

(Recalling that the multiplication is made in the semiring $(\mathbb{K}, \min, +)$.) Note that the $(k+1)$-th component is a virtual register that will be maintained to 0. Given an accepting run $(q_0, w_0, q_1) \cdot (q_1, w_1, q_2) \cdots (q_n, w_n, q_{n+1}) \in (Q \times A)^*$ in $\mathcal{A}$, the output value is then defined as:

$$\mathcal{C}(w_0 w_1 \cdots w_n) = \overline{\lambda} \cdot \mu(q_0, w_0) \cdot \mu(q_1, w_1) \cdots \mu(q_n, w_n) \cdot \nu(q_{n+1})\ .$$

The $\mathbb{K}$-CRA is said to be *copyless* ($\mathbb{K}$-CCRA) if all the update functions satisfy, using the notations above, that for all $i \in [k]$, $|\{j \mid m_{i,j} \neq \infty\}| \leq 1$; in words, for each $i$, at most one of the subexpressions "$r_i + m_{i,j}$" will evaluate to a non-$\infty$ value: the value of $r_i$ impacts at most one register.

**Vector addition systems with states and zero-tests.** The main construction of this paper focuses on simulating counters with zero-tests. The precise formalism for our counter machines is a variant of vector addition systems with states (VASS) over $\mathbb{Z}^k$, equipped with transitions that can only be fired if a designated counter is zero. For any $k$, we define the *update alphabet* $C_k$ as:

$$C_k = \bigcup_{i \in [k]} \{\mathbf{inc}_i, \mathbf{dec}_i, \mathbf{chk}_i\} \ ,$$

the intended meaning being that $\mathbf{inc}_i$ will increment the $i$-th counter, $\mathbf{dec}_i$ will decrement it, and $\mathbf{chk}_i$ will check that it is zero.

A $\mathbb{Z}$-VASS$^{\mathbf{z}}$ $\mathcal{V}$ of dimension $k$ is a DFA $(Q, C_k, \delta, q_0, F)$. Consider a *configuration* $K = (q, \overline{c}) \in Q \times \mathbb{Z}^k$; writing $(\overline{e_i}) \in \mathbb{Z}^k$ for the standard basis:

- If $\delta(q, \mathbf{inc}_i) = q'$, then $K$ can reach the configuration $(q', \overline{c} + \overline{e_i})$;
- If $\delta(q, \mathbf{dec}_i) = q'$, then $K$ can reach the configuration $(q', \overline{c} - \overline{e_i})$;
- If $\delta(q, \mathbf{chk}_i) = q'$, then $K$ can reach the configuration $(q', \overline{c})$ if and only if $c_i = 0$.

We say that the $\mathbb{Z}$-VASS$^{\mathbf{z}}$ *reaches* a state $q$ if $(q_0, \overline{0})$ reaches, by a sequence of configurations, $(q, \overline{c})$ for some $\overline{c}$. We write $L_{\mathcal{V},q} \subseteq (C_k)^*$ for the *reachability language* of $q$, that is, the language of updates along the runs reaching $q$.

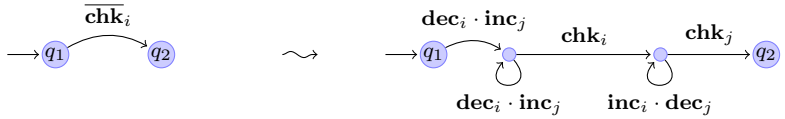
**Proposition 1.** *The following problem is undecidable:*

| | |
|---|---|
| ***Given:*** | *A $\mathbb{Z}$-VASS$^{\mathbf{z}}$ $\mathcal{V}$ and a state $q$* |
| ***Question:*** | *Is $L_{\mathcal{V},q}$ empty?* |

*The problem stays undecidable even if $|L_{\mathcal{V},q}| \leq 1$ is guaranteed.*

**Proof.** We define an extension of $\mathbb{Z}$-VASS$^{\mathbf{z}}$ that can implement classical Minsky machines to streamline the reduction. Define $C'_k = C_k \cup \bigcup_{i \in [k]} \{\overline{\mathbf{chk}}_i\}$. A $k$-counter machine is an automaton over $C'_k$, with the $\mathbb{Z}$-VASS$^{\mathbf{z}}$ semantics, augmented with the property that a transition labeled $\overline{\mathbf{chk}}_i$ can only be taken if the $i$-th counter is *strictly greater* than zero.

Minsky [13] showed that the emptiness of reachability languages is undecidable for these machines—in particular, even if it is assumed that there is at most one run reaching the given state. To show the same for $\mathbb{Z}$-VASS$^{\mathbf{z}}$, we need only remove the transitions labeled $\overline{\mathbf{chk}}_i$, while preserving the reachability languages. To do so, it suffices to replace them with the following gadget, where $j$ is a new counter and some states are omitted:



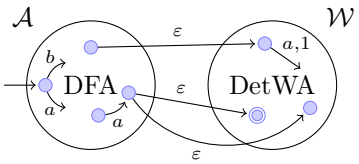
It is easily checked that upon reaching state $q_2$, the $i$-th counter is restored to its value in $q_1$, the $j$-th is 0, and the state can only be reached if the $i$-th counter were strictly positive. □

## 3. CCRA and weighted automata

With the plethora of models computing functions from words to values in modern literature, it is imperative to justify studying the seemingly artificial CCRA. In this section, we provide a normal form that will demonstrate that these machines are but deterministic weighted automata with a small dose of nondeterminism. More precisely, we show in Proposition 2 and Corollary 3 that the functions computed by *kccra* can be captured by a specific form of linearly-ambiguous weighted automata. In particular, all the problems we show to be undecidable in Section 6 turn out to be decidable for deterministic (or even finitely-ambiguous) weighted automata; this gives credence to the assertion that $\mathbb{N}_\infty$-CCRA is one of the weakest models for which equivalence, for instance, is undecidable.

We conclude the section by showing, in Proposition 5, that the class of linearly-ambiguous weighted automata strictly contains the class of functions computed by $\mathbb{K}$-CCRA .

In the following proposition, it is shown that any $\mathbb{K}$-CCRA can be expressed as a DFA making nondeterministic jumps into a $\mathbb{K}$-DetWA; graphically, every $\mathbb{K}$-CCRA is equivalent to:




**Proposition 2.** *Let* $\mathcal{C} = (\langle Q, A, \delta, q_0, F\rangle, \overline{\lambda}, \mu, \nu)$ *be a* $\mathbb{K}$-CCRA *with* $k$ *registers. There are a DFA* $\mathcal{A}$ *with state set* $Q \times \mathcal{P}([k])$ *and initial state* $q_0$*, a* $\mathbb{K}$-DetWA $\mathcal{W}$ *with state set* $Q'$*, and a function* $\eta\colon Q \times \mathcal{P}([k]) \to \mathcal{P}(Q')$ *such that:*

$$(\forall w \in A^*) \qquad \mathcal{C}(w) = \min\{\mathcal{W}^q(v) \mid w = uv \wedge q \in \eta(q_0.u)\} \ ,$$

*where* $\mathcal{W}^q$ *is* $\mathcal{W}$ *with the initial state set to* $q$*, and* $q_0.u$ *is the state reached by reading* $u$ *in* $\mathcal{A}$*.*

**Proof.** We first sketch the intuition behind the proof.

Consider the output of $\mathcal{C}$ on some word $w$. This value is of the form $r_i + m$ for some register $r_i$, or a constant $m$ (indeed, it is the minimum of several such expressions, and thus equals one of them). This value, in turn, was propagated by a sequence of similar expressions, up to a certain initial point where it was reset to a constant.

Viewing this in the order that $w$ is read, we see that at some point along the run, the value of $w$ is "created" by a reset of the form $r_i \leftarrow m$, and then the value is

propagated by adding constants to it (and possibly changing the register in which it is stored), until it is outputted.

Thus, in order to simulate $\mathcal{C}$ using a $\mathbb{K}$-DetWA, we can nondeterministically guess when the value is created, and follow the (deterministic) sequence of additions. In our construction, the DFA $\mathcal{A}$ keeps track of the recently-reset variables, and makes nondeterministic jumps to components of $\mathcal{W}$, that track the deterministic value updates.

We now turn to formalize this idea.

Consider a nondeterministic variant of a given $\mathbb{K}$-CCRA $\mathcal{C}$ where updates of the form $r_1 \leftarrow \min\{r_2, r_3\}$ become nondeterministic jumps between the updates $r_1 \leftarrow r_2$ and $r_1 \leftarrow r_3$. The final value of this variant is set to be the minimum output of any run. Then this variant has the same output value as the original CRA, by distributivity of addition over min.

We will assume that $\overline{\lambda} \in \{0, \infty\}^k$ and that the updates are in one of two possible forms:

- $r_i \leftarrow \min\{r_1 + m_{1,i}, r_2 + m_{2,_i}, \ldots, r_k + m_{k,i}\}$, that is, no constant term appears;
- $r_i \leftarrow 0$.

In symbols, this means that if $\mu(q, a) = (m_{i,j})$, then for any $i \in [k]$, either $m_{k+1,i}$ is $\infty$ or all $m_{j,i}$, for $j \in [k]$, are $\infty$. Any $\mathbb{K}$-CCRA can be put under that form using standard techniques. For instance, one can add fresh registers to replace the virtual 0-register in $\mu$. (Indeed, more than one may be needed so as to preserve the copyless restriction.)

The automaton $\mathcal{A}$ is the underlying automaton of $\mathcal{C}$, augmented with the information of *which registers* were reset by the previous transition. More precisely, $\mathcal{A} = (Q \times \mathcal{P}([k]), A, \delta_{\mathcal{A}}, q'_0, \emptyset)$ where $q'_0 = (q_0, \{i \mid \lambda_i = 0\})$; note that the final states are irrelevant. The transition function $\delta_{\mathcal{A}}$ is defined by:

$$\delta_{\mathcal{A}}((q, \cdot), a) = (\delta(q, a), E) \quad \text{where } E = \{i \mid \mu(q, a)_{k+1,i} = 0\} \ .$$

The $\mathbb{K}$-DetWA $\mathcal{W}$ consists of $k$ copies of $\mathcal{C}$, one for each register. Formally, $\mathcal{W} = (\mathcal{B}, \overline{0}, \mu_{\mathcal{W}}, \nu_{\mathcal{W}})$ with $\mathcal{B} = (Q \times [k], A, \delta_{\mathcal{B}}, (q_0, 1), F \times [k])$; here, the initial valuation is irrelevant. We now define the transition function $\delta_{\mathcal{B}}$ and the weight function $\mu_{\mathcal{W}}$. Let $(q, x)$ be a state of $\mathcal{B}$ and $a \in A$. By copylessness, there is at most one $y$ such that $\mu(q, a)_{x,y}$ is not $\infty$. If one such $y$ exists, then:

$$\delta_{\mathcal{B}}((q, x), a) = (\delta(q, a), y)$$
$$\mu_{\mathcal{W}}((q, x), a) = \mu(q, a)_{x,y} \ .$$

The output function of $\mathcal{W}$ is then, for any $q \in Q, i \in [k]$, $\nu_{\mathcal{W}}(q, i) = \nu(q)_i$.

Finally, $\eta \colon Q \times \mathcal{P}([k]) \to \mathcal{P}(Q \times [k])$, whose existence is claimed in the Proposition, is defined as $\eta(q, E) = \{(q, i) \mid i \in E\}$.

Consider a word $w \in A^*$, and a factorization $w = uv$. The word $u$ reaches a state $q$ in $\mathcal{C}$, and a state $(q, E)$ in $\mathcal{A}$. The last transition taken in $\mathcal{C}$ reading $u$ updated

all the registers $r_i, i \in E$, with the value 0. For each of these $i$'s, there will be a run over $v$ in $\mathcal{W}$, starting at $(q, i)$, which follows the updates applied to $r_i$. This process thus simulates the nondeterministic variant of $\mathcal{C}$ described above, showing the Proposition. □

This normal form is akin to those which can be extracted from the work of Weber and Seidl on degrees of ambiguity (Equation 6 in [17]). We have:

**Corollary 3.** $\mathbb{K}$-CCRA $\subseteq \mathbb{K}$-LinWA.

**Proof.** With the notations of Proposition 2, let us see $\mathcal{A}, \eta$, and $\mathcal{W}$ as a single $\mathbb{K}$-WA, where the weights in the $\mathcal{A}$ part are set to 0. For any word $w$, each run on $w$ consists of a run over a prefix $u$ within $\mathcal{A}$, and a run over the leftover suffix $v$ within $\mathcal{W}$ starting in some state $q \in \eta(q_0.u)$. Thus there are at most $|w| \times |Q'|$ runs, hence the WA is linearly ambiguous. □

**Remark 4.** *Note that Proposition 2 and Corollary 3 hold for arbitrary semirings – indeed, the proof proceeds without any modification, apart from replacing* $+$ *and* min *with the corresponding semiring operations. Since the focus of this work are tropical semirings, the proofs were given in the corresponding notation.*

As an application of this specific form, it is not hard to show that some specific functions are not expressible using a $\mathbb{Z}_\infty$-CCRA. Let $\mathsf{minblock}$ (resp. $\mathsf{lastblock}$) be the function from $\{a, \#\}^*$ to $\mathbb{N}$ which, given $w = \#a^{n_1}\#a^{n_2}\#\cdots\#a^{n_k}\#$ returns $\min\{n_i\}_{i\in[k]}$ (resp. $n_k$):

**Proposition 5.** *The following functions are not expressible by a* $\mathbb{Z}_\infty$-CCRA*:*

- $c^i \cdot w \mapsto i + \mathsf{minblock}(w)$*, with* $w \in \{a, \#\}^*$*;*
- $u \cdot \$ \cdot v \mapsto \mathsf{lastblock}(u) + \mathsf{lastblock}(v)$*, with* $u, v \in \{a, \#\}^*$*.*

**Proof.** In both cases, one has to reason about when the nondeterministic jump, given by $\eta$ in Proposition 2, is made in the minimal run, bearing in mind that neither $\mathsf{minblock}$ nor $\mathsf{lastblock}$ are computable by a DetWA. For the first example, the jump has to be made at the beginning of the minimal block of $a$'s, after reading a #; thus the number of $c$'s cannot be taken into account. For the second example, if the jump is made just before the last block of $a$'s in $v$, then the value of the last block in $u$ is disregarded. If it is made just before the last block in $u$, then the DetWA part has to compute $\mathsf{lastblock}$ on $v$, which is not possible.

For a formal proof of the claim, we use Proposition 2 and assume that the DetWA $\mathcal{W}$ has $n$ states, its largest absolute weight is $W$. Let us first focus on the first function. Consider a word $v = \#a^{nW+n}\#a^{(n-1)W+n-1}\#\ldots\#a^{W+1}\#a\#$. It is easy to see that when reading $c^{nW+1} \cdot v$, there must be a run which transitions from $\mathcal{A}$ to $\mathcal{W}$ after the last $c$ has been read. Indeed, after every read block, a run in $\mathcal{W}$ has too high a weight to be relevant for the computation of the value of the next

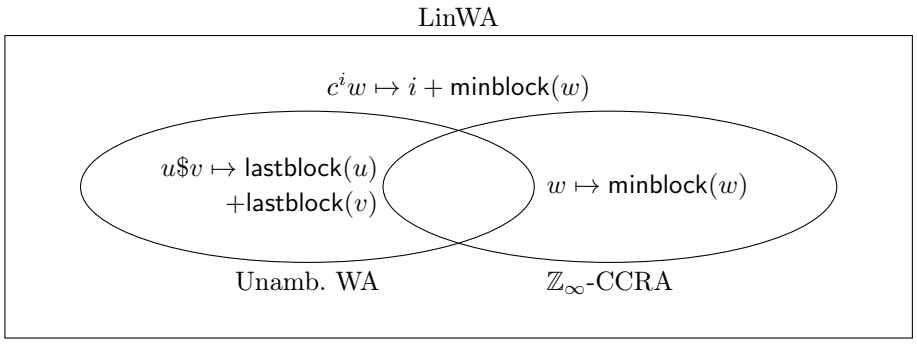



Figure 1. Graphical depiction of the inclusions outlined in Remark 6 between function realized by $\mathbb{Z}_\infty$-CCRA, LinWA, and unambiguous WA.

block. (Note that the blocks of $a$'s are sorted from longest to shortest.) As there are one too many blocks, a new run with a smaller weight – and which can only come from $\mathcal{A}$ – is needed. Let $v'$ be the prefix of $v$ after which such a run with weight 0 moves into state $q$ from $\mathcal{W}$. Since this new run can be extended to a minimal-weight run ending at a final state, by our previous argument, we know that there is some path from $q$ to $F$. A shortest such path has length $n$ and so $c^{nW+1}v'y$, for some $y$ of length at most $n$, is in the language of the automaton. Note that its value should be at least $nW + 1$ and yet the value of the run from $q$ cannot be larger than $nW$, leading to a contradiction.

Because the blocks in $v$ are sorted by length, an almost identical argument can be used to prove the second part of the claim. It suffices to replace the prefix of $c$'s by $\#a^{nW+1}\#\$$. □

**Remark 6.** *Note that the first function of Proposition 5 is expressible by a LinWA, and the second by an unambiguous WA (i.e., at most one run per accepted word). Moreover, since* minblock *is not expressible by an unambiguous WA but is by a* CCRA *(see Section 1), the classes of functions expressed by the two models are incomparable. (See Figure 1.)*

## 4. Simulation of $\mathbb{Z}$-VASS$^{\mathbf{z}}$ using $\mathbb{Z}_\infty$-CCRA

In this section we show how to simulate $\mathbb{Z}$-VASS$^{\mathbf{z}}$ using $\mathbb{Z}_\infty$-CCRA. We start by considering the case of a single counter, and then extend it to multiple counters. Finally, we show how to encode the result of the simulation in one of the registers.

Let $\mathcal{V}$ be a $\mathbb{Z}$-VASS$^{\mathbf{z}}$ and $q$ a state of $\mathcal{V}$. Recall that $C_k$ is the update alphabet of symbols $\mathbf{inc}_i, \mathbf{dec}_i$, and $\mathbf{chk}_i$, for $i \in [k]$, and that $L_{\mathcal{V},q} \subseteq (C_k)^*$ is the reachability language of $q$ (i.e., the update sequences along runs that reach $q$ in $\mathcal{V}$). We devise a simulation of $\mathcal{V}$ using $\mathbb{Z}_\infty$-CCRA in the following sense: Given a word $w \in (C_k)^*$, the $\mathbb{Z}_\infty$-CCRA will output 0 if and only if $w \in L_{\mathcal{V},q}$.

Compared to the simulation by $\mathbb{N}_\infty$-CCRA of the forthcoming Section 5, the $\mathbb{Z}$ case is quite straightforward, and reminiscent of the methodology of [1]; it however provides some intuition for the construction for $\mathbb{N}$.

We present how the counter increments (**inc**), decrements (**dec**), and zero-tests (**chk**) are implemented for a single counter before showing how multiple counters can be handled. The automaton structure of the source $\mathbb{Z}$-VASS$^{\mathbf{z}}$, with accepting state $q$, can then be followed by the CRA while simulating the counters.

### 4.1. *Simulation of a single counter*

Since we are working with a single counter, we drop the indices of the letters in $C_1$. A single counter $c$ will be simulated with 3 registers: $r^+$ and $r^-$, carrying the values of $c$ and $-c$, respectively, and $r^{\mathbf{z}}$ which shall be 0 if each time the letter **chk** was read, $c$ was 0. If at any time **chk** was read while $c$ was nonzero, then $r^{\mathbf{z}}$ will be strictly smaller than 0. This is implemented as follows:

$$\textbf{inc}\colon \begin{cases} r^+ \leftarrow r^+ + 1 \\ r^- \leftarrow r^- - 1 \\ r^{\mathbf{z}} \leftarrow r^{\mathbf{z}} \end{cases} \quad \textbf{dec}\colon \begin{cases} r^+ \leftarrow r^+ - 1 \\ r^- \leftarrow r^- + 1 \\ r^{\mathbf{z}} \leftarrow r^{\mathbf{z}} \end{cases} \quad \textbf{chk}\colon \begin{cases} r^+ \leftarrow 0 \\ r^- \leftarrow 0 \\ r^{\mathbf{z}} \leftarrow \min\{r^{\mathbf{z}}, r^+, r^-\} \end{cases}$$

**Assertion 7.** *If $r^{\mathbf{z}}$ becomes strictly smaller than 0, it will stay so after reading any word in $(C_1)^*$.*

**Assertion 8.** *Assume $r^+ = r^- = r^{\mathbf{z}} = 0$. After reading $i$ letters* **inc** *and $j$ letters* **dec***, in any order, and a final* **chk***, the new values of the registers satisfy:*

*(1) If $i = j$, then $r^+ = r^- = r^{\mathbf{z}} = 0$;*

*(2) Otherwise $r^{\mathbf{z}} < 0$.*

This simulates the original counter in the following sense:

**Proposition 9.** *Let $\mathcal{V}$ be a $\mathbb{Z}$-VASS$^{\mathbf{z}}$ of dimension 1 and $q$ a state of $\mathcal{V}$. There is a $\mathbb{Z}_\infty$-CCRA $\mathcal{C}$ with $\mathcal{C}(w) \leq 0$ for any $w$ and such that:*

$$(\forall w \in (C_1)^*) \quad w \in L_{\mathcal{V},q} \Leftrightarrow \mathcal{C}(w) = 0 \ .$$

**Proof.** Let $\mathcal{V} = (Q, C_1, \delta, q_0, F)$ and $q \in Q$. The $\mathbb{Z}_\infty$-CCRA $\mathcal{C}$ with 3 registers is defined as having $(Q, C_1, \delta, q_0, \{q\})$ as the automaton structure, and the updates are dictated by the letter being read, as above. On state $q$, $\mathcal{C}$ outputs $r^{\mathbf{z}}$. By Assertions 7 and 8, we conclude the proposition. □

### 4.2. *Simulation of multiple counters*

It is quite straightforward to combine multiple $r^{\mathbf{z}}$ registers into one. Indeed, if $k$ counters are simulated using registers $r_i^+, r_i^-$, and $r_i^{\mathbf{z}}$, $i \in [k]$, then at the end of the simulation, one can set:

$$\mathit{flag} \leftarrow \min\{r_1^{\mathbf{z}}, r_2^{\mathbf{z}}, \ldots, r_k^{\mathbf{z}}\} \ ,$$

so that *flag* is 0 if and only if the execution saw no illegal zero-tests (and negative otherwise).

**Proposition 10.** *Let $\mathcal{V}$ be a $\mathbb{Z}$-VASS$^{\mathbf{z}}$ of dimension $k$ and $q$ a state of $\mathcal{V}$. There is a $\mathbb{Z}_\infty$-CCRA $\mathcal{C}$ with $\mathcal{C}(w) \leq 0$ for any $w$ and such that:*

$$(\forall w \in (C_k)^*) \qquad w \in L_{\mathcal{V},q} \Leftrightarrow \mathcal{C}(w) = 0 \ .$$

### 4.3. *Outputting on correct executions*

So far, we were interested in having a specific output if the simulated execution was correct. If we wanted, by contrast, to output *one of the registers* on correct executions, we would need one more idea; we note that the same idea will be reused in Section 5 for the simulation using $\mathbb{N}_\infty$-CCRA.

Formally, we show the following.

**Proposition 11.** *Let $\mathcal{V}$ be a $\mathbb{Z}$-VASS$^{\mathbf{z}}$ of dimension $k$ and $q$ a state of $\mathcal{V}$. There is a $\mathbb{Z}_\infty$-CCRA $\mathcal{C}$ over the alphabet $C_k \cup \{z\}$ (where $z \notin C_k$), whose output is the value of a designated register, and the following hold:*

*(1) For every $w \in L_{\mathcal{V},q}$ there exists a unique $n \in \mathbb{N}$ such that $\mathcal{C}(wz^n)$ is even,*
*(2) For every $w' \in (C_k \cup \{z\})^*$, if $\mathcal{C}(w')$ is even, then $w' = wz^n$ for some $w \in L_{\mathcal{V},q}$ and $n \in \mathbb{N}$.*

In the remainder of the section we illustrate the proof of Proposition 11.

Suppose that we wish to output the register $r$ if and only if *flag* is 0; recall that *flag* may only be 0 or negative. We will do so by repeatedly reading a new letter $z$, and having $r$ be the only possible *even* output value, provided *flag* is 0—no even value is produced if *flag* is negative.

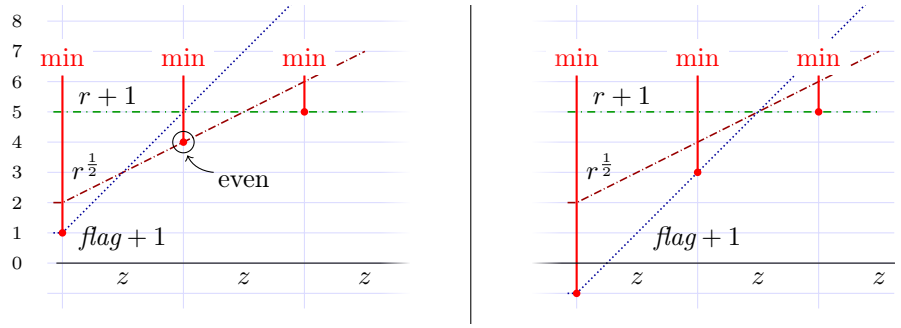
We may assume that, by construction, *flag* is even and $r$ is a multiple of 4; we further assume that we have a register $r^{\frac{1}{2}}$ that contains *half* of $r$'s value. We add the letter $z$ to our alphabet, to be read at the end of the simulation; reading $z$ increases $r^{\frac{1}{2}}$ by 2 and *flag* by 4. The output value is then set to:

$$\min\{r + 1, \mathit{flag} + 1, r^{\frac{1}{2}}\} \ .$$

Write $s$ for the value of $r$ before reading the $z$'s, and $f$ for the value of *flag*. After reading $i$ letters $z$, the new values of the registers read:

$$r = s, \quad \mathit{flag} = f + 4 \times i, \quad r^{\frac{1}{2}} = \frac{s}{2} + 2 \times i \ .$$

For an even output to be produced, $r^{\frac{1}{2}}$ has to be minimal. If $f$ is 0, this happens only when $i = \frac{s}{4}$, and the output is then $s$. If $f$ is negative, then $\mathit{flag} < r^{\frac{1}{2}}$ for $i \leq \frac{s}{4}$ and $r < r^{\frac{1}{2}}$ for larger values of $i$; in that case, no even output value is produced. This is illustrated in the following graphics, where $s = 4$, and the left-hand side depicts the case $f = 0$, while, in the right-hand side, $f = -2$.




## 5. Simulation of $\mathbb{Z}$-VASS$^{\mathbf{z}}$ using $\mathbb{N}_\infty$-CCRA

In this section, we devise a simulation of $\mathcal{V}$ using $\mathbb{N}_\infty$-CCRA in the following sense: Let $\mathcal{V}$ be a $\mathbb{Z}$-VASS$^{\mathbf{z}}$ and $q$ a state of $\mathcal{V}$. Given a word $w \in (C_k)^*$, the $\mathbb{N}_\infty$-CCRA will output an *even value* if and only if $w \in L_{\mathcal{V},q}$.

Translating the simulation strategy for $\mathbb{Z}$, described in Section 4, to the $\mathbb{N}$ setting turns out to be a nontrivial matter. Indeed, one might expect that it would be enough to increase the updates so that no negative number appears therein. This would contribute a linear blowup to the values, but does not seem to change the overall behavior. However, the resets made while reading $\mathbf{chk}_i$ would have to be equal to that blowup, and this would require copying.

The simulation will thus follow two phases. First, one that corresponds to the strategy for $\mathbb{Z}$ with the updates tweaked to be positive; second, after reading a $\mathbf{chk}_i$, a *climb-back* phase that puts the registers back in a manageable state (called "ready" later on). For this latter phase, the $\mathbb{N}_\infty$-CCRA will read a word in $\mathbf{cb}_i^* \cdot \mathbf{chkcb}_i$—the letter $\mathbf{cb}$ standing for *climb-back*. Further, combining the acceptance conditions of multiple counters will also require some new letters; the alphabet of the automaton is thus:

$$C'_k = C_k \cup \bigcup_{i \in [k]} \{\mathbf{cb}_i, \mathbf{chkcb}_i, z_i\} \ .$$

### 5.1. *Simulation of a single counter*

Again, since we are working with a single counter, we drop the indices of the letters in $C'_1$. A single counter in the $\mathbb{Z}$-VASS$^{\mathbf{z}}$ will be simulated by 7 different registers, each with a simple intended meaning:

- $r^+$ and $r^-$ should respectively count the number of increments and decrements of the counter;
- $r^{\mathrm{u}}$ increases each time the counter is either incremented or decremented; it counts the number of *updates* to the counter;
- The register $r^{\frac{\mathrm{u}}{2}}$ should be half [b] of $r^{\mathrm{u}}$;

[b]The concerned reader may wonder how we encode "half" in integers. This is done by effectively

- $r^{\mathbf{z}}$ will be a witness that the **chk** letter has always been read when the simulated counter was zero and that the climb-back phases were done correctly (this notion is formalized in Lemma 13 below);
- Finally, we will need two internal registers $r^{\mathrm{cb}}$ and $r^{2\mathrm{cb}}$, used solely in the climb-back phase.

To simplify the discussion, we give names to some register configurations:

- They are *ready* if $r^+ = r^- = r^{\mathbf{z}} = r^{\frac{\mathrm{u}}{2}} = \frac{1}{2} \times r^{\mathrm{u}}$;
- They are *to-climb* if $r^+ = r^- = 0$ and $r^{\mathbf{z}} = r^{\frac{\mathrm{u}}{2}} = \frac{1}{2} \times r^{\mathrm{u}}$;
- They are *dead* if $r^{\mathbf{z}} < r^{\frac{\mathrm{u}}{2}}$.

In the first two configurations, we also assume that $r^{\mathrm{cb}} = r^{2\mathrm{cb}} = 0$.

**Goal of the construction.** We will show that if the registers are ready and we read an equal number of **inc**'s and **dec**'s followed by a **chk**, then the registers become to-climb. There is then a precise number $i$ such that reading $\mathbf{cb}^i \cdot \mathbf{chkcb}$ will put the registers back in ready mode. Crucially, if the numbers of **inc**'s and **dec**'s are not equal, or an incorrect number of **cb**'s is read, then the registers become dead.

The updates are as follows, where the registers not shown are simply preserved. As we saw in Section 4.3, we will require that the values of the registers be divisible by some values, hence rather than incrementing with 1, we increment by $e \in \mathbb{N}$, a value we shall determine later. Note that these are indeed copyless updates.

$$\mathbf{inc}\colon \begin{cases} r^+ \leftarrow r^+ + e \\ r^{\mathrm{u}} \leftarrow r^{\mathrm{u}} + e \\ r^{\frac{\mathrm{u}}{2}} \leftarrow r^{\frac{\mathrm{u}}{2}} + \frac{e}{2} \\ r^{\mathbf{z}} \leftarrow r^{\mathbf{z}} + \frac{e}{2} \end{cases} \quad \mathbf{dec}\colon \begin{cases} r^- \leftarrow r^- + e \\ r^{\mathrm{u}} \leftarrow r^{\mathrm{u}} + e \\ r^{\frac{\mathrm{u}}{2}} \leftarrow r^{\frac{\mathrm{u}}{2}} + \frac{e}{2} \\ r^{\mathbf{z}} \leftarrow r^{\mathbf{z}} + \frac{e}{2} \end{cases} \quad \mathbf{chk}\colon \begin{cases} r^+ \leftarrow 0 \\ r^- \leftarrow 0 \\ r^{\mathbf{z}} \leftarrow \min\{r^{\mathbf{z}}, r^+, r^-\} \end{cases}$$

$$\mathbf{cb}\colon \begin{cases} r^+ \leftarrow r^+ + e \\ r^- \leftarrow r^- + e \\ r^{\frac{\mathrm{u}}{2}} \leftarrow r^{\frac{\mathrm{u}}{2}} + \frac{e}{2} \\ r^{\mathbf{z}} \leftarrow r^{\mathbf{z}} + \frac{e}{2} \\ r^{\mathrm{cb}} \leftarrow r^{\mathrm{cb}} + e \\ r^{2\mathrm{cb}} \leftarrow r^{2\mathrm{cb}} + 2 \times e \end{cases} \quad \mathbf{chkcb}\colon \begin{cases} r^{\mathrm{cb}} \leftarrow 0 \\ r^{2\mathrm{cb}} \leftarrow 0 \\ r^{\mathrm{u}} \leftarrow r^{2\mathrm{cb}} \\ r^{\mathbf{z}} \leftarrow \min\{r^{\mathbf{z}}, r^{\mathrm{cb}}, r^{\mathrm{u}}\} \end{cases}$$

**Assertion 12.** *If the registers are dead, they will stay so after reading any word in* $(C_1')^*$.

**Lemma 13.** *Assume the registers are ready. After reading* $i$ *letters* **inc** *and* $j$ *letters* **dec***, in any order, and a final* **chk***, the new values of the registers satisfy:*

*(1) If* $i = j$*, then they are to-climb;*

multiplying all registers by some large enough constant.

*(2) Otherwise, they are dead.*

**Proof.** Suppose $r^+ = r^- = r^{\mathbf{z}} = r^{\frac{\mathrm{u}}{2}} = \frac{1}{2} \times r^{\mathrm{u}}$, and let us name that value $s$. After reading $i$ letters **inc** and $j$ letters **dec**, the new values are:

$$r^+ = s + e \times i, \quad r^- = s + e \times j, \quad r^{\mathbf{z}} = r^{\frac{\mathrm{u}}{2}} = \frac{1}{2} \times r^{\mathrm{u}} = s + e \times \frac{i+j}{2} \ .$$

Now, if $i = j$ then $r^+ = r^- = r^{\mathbf{z}} = r^{\frac{\mathrm{u}}{2}} = \frac{1}{2} \times r^{\mathrm{u}}$, thus reading **chk** will indeed make the registers to-climb. Otherwise, one of $r^+$ or $r^-$ is smaller than $r^{\mathbf{z}}$, and reading **chk** will make the registers dead. □

**Lemma 14.** *Assume the registers are to-climb. After reading* $\mathbf{cb}^i \cdot \mathbf{chkcb}$*, the new values of the registers satisfy:*

*(1) If $i$ is equal to the starting value of $r^{\mathbf{z}}$ multiplied by $\frac{2}{e}$, then they are ready;*
*(2) Otherwise, they are dead.*

**Proof.** Suppose $r^+ = r^- = 0$ and $r^{\mathbf{z}} = r^{\frac{\mathrm{u}}{2}} = \frac{1}{2} \times r^{\mathrm{u}}$; we name that latter value $s$. After reading $i$ letters **cb**, the new values are:

$$r^+ = r^- = r^{\mathrm{cb}} = \frac{1}{2} \times r^{2\mathrm{cb}} = e \times i, \quad r^{\mathbf{z}} = r^{\frac{\mathrm{u}}{2}} = s + e \times \frac{i}{2}, \quad r^{\mathrm{u}} = 2 \times s \ .$$

Now if $i = \frac{2 \times s}{e}$, then $r^+ = r^- = r^{\mathrm{cb}} = \frac{1}{2} \times r^{2\mathrm{cb}} = r^{\mathbf{z}} = r^{\frac{\mathrm{u}}{2}} = 2 \times s$. Reading **chkcb** thus makes the registers ready. If $i$ is smaller than $\frac{2 \times s}{e}$ then $r^{\mathrm{cb}} < r^{\mathbf{z}}$; if it is greater, then $r^{\mathrm{u}} < r^{\mathbf{z}}$: reading **chkcb** thus makes the registers dead. □

As we describe in the goal of the construction above, Lemma 13 ensures that a correct number of increases and decreases is read before reading $r^{\mathbf{z}}$, Lemma 14 ensures that the correct number of $r^{\mathrm{cb}}$ occurrences is read, and Assertion 12 ensures that any violation cannot be recovered later.

### 5.2. *Simulation of multiple counters*

We just saw how to simulate a single counter in the sense that the registers are not dead if and only if the input word describes a correct run (i.e., one in which **chk** is only read if the counter is 0). Let us now exhibit a method that combines multiple such simulations, and outputs an even value if and only if none of the simulations is dead. To do so, we will repeatedly read new letters $z_1, z_2, \ldots, z_k$ at the very end of the execution, in a similar fashion as done in Section 4.3.

Let us suppose we have $k$ simulated counters, hence $k$ sets of 7 registers. For this phase, we will only use $r_i^{\mathbf{z}}$, for each $i$, but we will have *one* more register in our $\mathbb{N}_\infty$-CCRA, named $r^{\mathrm{avg}}$. The purpose of $r^{\mathrm{avg}}$ is to hold the average of all the $r_i^{\frac{\mathrm{u}}{2}}$; this is easily achieved by adding to the above updates:

$$r^{\mathrm{avg}} \leftarrow r^{\mathrm{avg}} + \frac{e}{2 \times k}$$

whenever a $r_i^{\frac{u}{2}}$ is incremented (always by $\frac{e}{2}$). Now for each $i$, the new letter $z_i$ will update the registers with:

$$\begin{cases} r_i^{\mathbf{z}} & \leftarrow r_i^{\mathbf{z}} + \frac{e}{2} \\ r^{\text{avg}} & \leftarrow r^{\text{avg}} + \frac{e}{2\times k} \end{cases}$$

The output value of the $\mathbb{N}_\infty$-CCRA is then set to

$$\min\{r^{\text{avg}}, r_1^{\mathbf{z}} + 1, r_2^{\mathbf{z}} + 1, \dots, r_k^{\mathbf{z}} + 1\} \ . \tag{1}$$

We let $e$ be such that all the registers are even (e.g., $e = 4k$).

If $r^{\text{avg}}$ was the average of the $r_i^{\mathbf{z}}$'s before reading the $z_i$'s—and this only happens if none of the register set was dead—it will stay so reading $z_i$'s. Consequently, there is a number of each letter $z_i$ that can be read so that all the $r_i^{\mathbf{z}}$'s are equal, making $r^{\text{avg}}$ the output value of the $\mathbb{N}_\infty$-CCRA.

If $r^{\text{avg}}$ was greater than the average of the $r_i^{\mathbf{z}}$'s—implying that at least one set of registers was dead—then $r^{\text{avg}}$ will never be the output of the CCRA after reading $z_i$'s.

**Theorem 15 (Simulation)** *Let $\mathcal{V}$ be a $\mathbb{Z}$-VASS$^{\mathbf{z}}$ of dimension $k$ and $q$ a state of $\mathcal{V}$. Write $h\colon (C'_k)^* \to (C_k)^*$ for the function that erases the letters $\mathbf{cb}_i, \mathbf{chkcb}_i$, and $z_i$. There is an $\mathbb{N}_\infty$-CCRA $\mathcal{C}$ such that for all $w \in (C_k)^*$:*

$$w \in L_{\mathcal{V},q} \Leftrightarrow (\exists w' \in h^{-1}(w))[\mathcal{C}(w') \text{ is even}] \ .$$

*Moreover, if there exists such $w'$, then it is unique.*

**Proof.** The only detail left to deal with is the uniqueness of the $w'$. We can certainly make sure that $\mathcal{C}$ outputs a value if and only if the input is of the form:

$$(\mathbf{inc}_i + \mathbf{dec}_i + \mathbf{chk}_i \cdot \mathbf{cb}_i^* \cdot \mathbf{chkcb}_i)^* \cdot (z_i)_i^* \ ,$$

but even if the first half (without the $z_i$'s) is indeed unique, as per Lemma 14, the $z_i$'s need not be so. To preserve uniqueness, this latter part is replaced by:

$$\bigcup_{j\in[k]} \prod_{\substack{i=1,\dots,k \\ i\neq j}} (z_i)^* \ .$$

This serves two purposes: first, the order on the $z_i$'s is fixed; second, one of the $z_j$ will *not* be used, hence the condition that all the $r_i^{\mathbf{z}}$ be equal will only be satisfied when they all evaluate to $r_j^{\mathbf{z}}$ for some $j$. Naturally, such a $j$ exists, it is simply the index of a maximal $r_i^{\mathbf{z}}$, making $\prod_{i=1,\dots,k} (z_i)^{r_j^{\mathbf{z}} - r_i^{\mathbf{z}}}$ the only possible suffix leading to an even value. □

## 6. Applications

In this section we draw a number of undecidability results as consequences of the simulations presented in Sections 4 and 5.

**Theorem 16 (Equivalence)** *The following problem is undecidable:*

***Given:*** *Two* $\mathbb{N}_\infty$-CCRA $\mathcal{C}$ *and* $\mathcal{C}'$ *over* $A^*$

***Question:*** $(\forall w \in A^*)[\mathcal{C}(w) = \mathcal{C}'(w)]$

**Proof.** Let $\mathcal{V}$ be a $\mathbb{Z}$-VASS$^{\mathbf{z}}$ and $q$ a state of $\mathcal{V}$, and consider the $\mathbb{N}_\infty$-CCRA $\mathcal{C}$ that simulates $L_{\mathcal{V},q}$. We reduce deciding if that language is empty (which is undecidable by Proposition 1) to the problem at hand. Equation (1), defining the output of $\mathcal{C}$, is such that $r^{\text{avg}}$ is the minimum if and only if the execution was correct. Thus replacing this output function by:

$$\min\{r_1^{\mathbf{z}} + 1, r_2^{\mathbf{z}} + 1, \ldots, r_k^{\mathbf{z}} + 1\}$$

changes the output value of a word if and only if it was a correct run. Calling $\mathcal{C}'$ this modified version, it holds that $(\forall w \in A^*)[\mathcal{C}(w) = \mathcal{C}'(w)]$ if and only if $L_{\mathcal{V},q} = \emptyset$. □

Since equality can be reduced to two-way inequality, we immediately obtain the following.

**Corollary 17 (Inequality)** *The following problem is undecidable:*

***Given:*** *Two* $\mathbb{N}_\infty$-CCRA $\mathcal{C}$ *and* $\mathcal{C}'$ *over* $A^*$

***Question:*** $(\forall w \in A^*)[\mathcal{C}(w) \leq \mathcal{C}'(w)]$

A direct consequence of Theorem 15 and Proposition 1, it is undecidable whether the image of an $\mathbb{N}_\infty$-CCRA is always odd. Furthermore, that image may be non-semilinear (see the following proof), and:

**Theorem 18 (Semilinearity)** *The following problem is undecidable:*

***Given:*** *An* $\mathbb{N}_\infty$-CCRA $\mathcal{C}$ *over* $A^*$

***Question:*** *Is* $\mathcal{C}(A^*)$ *semilinear, i.e., an eventually periodic set?*

**Proof.** We provide an independent construction which bears some similarities to the "climb-back" method. It doubles a register $r$ in the following sense: if $r$ is a register with starting value $s$, then reading $\mathbf{cb}^{s/2} \cdot \mathbf{chkcb}$ *doubles* the value of $r$; if any other number of **cb**'s is read (which happens in particular when $s$ is odd), the new value of $r$ will be some odd number.

Consider a register $r$ with initial value $s$, and suppose we have an additional register $r'$ holding $2 \times s$. We introduce two new registers, $r^{\text{cb}}$ and $r^{\text{2cb}}$ initialized with 0. Upon reading a word $\mathbf{cb}^i \cdot \mathbf{chkcb}$, we apply the updates:

$$\mathbf{cb}\colon \begin{cases} r & \leftarrow r + 2 \\ r^{\text{cb}} & \leftarrow r^{\text{cb}} + 4 \\ r^{\text{2cb}} & \leftarrow r^{\text{2cb}} + 8 \end{cases} \qquad \mathbf{chkcb}\colon \begin{cases} r^{\text{cb}} & \leftarrow 0 \\ r^{\text{2cb}} & \leftarrow 0 \\ r' & \leftarrow r^{\text{2cb}} \\ r & \leftarrow \min\{r, r' + 1, r^{\text{cb}} + 1\} \end{cases}$$

After reading $\mathbf{cb}^i$, it holds that $r = s + 2 \times i$, $r^{\mathrm{cb}} = 4 \times i$, and $r^{2\mathrm{cb}} = 8 \times i$.

If $i = \frac{s}{2}$, then $r = r^{\mathrm{cb}} = r' = 2 \times s$, hence after reading **chkcb**, we have indeed $r = \frac{r'}{2} = 2 \times s$, and the extra registers are reset: we are back to our starting hypothesis.

If $i \neq \frac{s}{2}$, then either $r' < r$ (when $i > \frac{s}{2}$) or $r^{\mathrm{cb}} < r$ (when $i < \frac{s}{2}$). In both cases, after reading **chkcb**, $r$ becomes odd, and will stay so after reading any other word.

(As a side note, consider the $\mathbb{N}_\infty$-CCRA with the above updates and $r$ initialized to 2, that reads words in $(\mathbf{cb}^* \cdot \mathbf{chkcb})^*$. Then the only even outputs of this machine are the powers of two, a nonsemilinear set.)

This concludes the construction. We now present a reduction from the problem of Proposition 1.

Let $\mathcal{V}$ be a $\mathbb{Z}$-VASS$^{\mathbf{z}}$ and $q$ a state of $\mathcal{V}$, and consider the $\mathbb{N}_\infty$-CCRA $\mathcal{C}$ that simulates $L_{\mathcal{V},q}$. We assume that $|L_{\mathcal{V},q}| \leq 1$, and again reduce deciding $L_{\mathcal{V},q} = \emptyset$ to the problem at hand.

First we note that we may assume that $\mathcal{C}$ outputs all the odd numbers, for instance by adding a letter $\ell$ and, upon reading $\ell^n$, outputting $2 \times n + 1$. Also recall that if $L_{\mathcal{V},q}$ is nonempty, then there is a unique $w$ such that $\mathcal{C}(w)$ is even.

We now modify $\mathcal{C}$ into $\mathcal{C}'$ to incorporate the above machinery. We simply store in a new register $r$ the output value of $\mathcal{C}$, and proceed by reading words of the form $\mathbf{cb}^i \cdot \mathbf{chkcb}$ with the updates as above. If $L_{\mathcal{V},q} = \emptyset$, then $\mathcal{C}'((C'_k)^*)$ is all the odd numbers, a semilinear set. Otherwise, there is one (and only one) even value $s$ in the image of $\mathcal{C}$, and it holds that:

$$\mathcal{C}'((C'_k)^*) = (2\mathbb{N} + 1) \cup \{2^i \times s \mid i \geq 0\} \ ,$$

a nonsemilinear set. □

Finally, we show the undecidability of upperboundedness of $\mathbb{Z}_\infty$-WA. We remark that this result is known as folklore. However, to the best of our knowledge, it is not published.

**Theorem 19 (Upperboundedness)** *The following problem is undecidable:*

| | |
|---|---|
| ***Given:*** | *A $\mathbb{Z}_\infty$-WA $\mathcal{A}$ over $A^*$* |
| ***Question:*** | $(\exists c \in \mathbb{Z})(\forall w \in A^*)[\mathcal{A}(w) \leq c]$ |

**Proof.** Let $\mathcal{V}$ be a $\mathbb{Z}$-VASS$^{\mathbf{z}}$ and $q$ a state of $\mathcal{V}$, and consider the $\mathbb{Z}_\infty$-CCRA $\mathcal{C}$ that simulates $L_{\mathcal{V},q}$. Relying on Proposition 2, let $\mathcal{W}$ be a $\mathbb{Z}_\infty$-WA equivalent to $\mathcal{C}$. Tweak $\mathcal{W}$ to output the same as $\mathcal{C}$ *plus one*, hence $\mathcal{W}(w)$ is 1 if and only if $w \in L_{\mathcal{V},q}$. Now let $\mathcal{W}'$ be $\mathcal{W}$ with an added letter # that jumps from the final states of $\mathcal{W}$ to its initial state; formally, let $\mathcal{W} = (\mathcal{A}, \lambda, \mu, \nu)$ with $\mathcal{A} = (Q, A, \delta, q_0, F)$, then $\mathcal{W}'$ is $(\mathcal{A}', \lambda, \mu', \nu)$ where $\mathcal{A}' = (Q, A \uplus \{\#\}, \delta \cup \{(q, \#, q_0) \mid q \in F\}, q_0, F)$, and $\mu'$ agrees with $\mu$ on $\delta$ and is extended by $\mu(q, \#, q_0) = \nu(q) + \lambda$.

In essence, $\mathcal{W}'$ is iterating $\mathcal{W}$:

$$\mathcal{W}'(w_1\#w_2\#\cdots\#w_k) = \sum_{i\in[k]} \mathcal{W}(w_i) \ .$$

From this, we see that if $\mathcal{W}$ is always negative or zero, $\mathcal{W}'$ is bounded, otherwise, if $\mathcal{W}(w) = 1$, then $\mathcal{W}'((w\#)^c \cdot w) = c + 1$, hence $\mathcal{W}'$ is unbounded. □

## 7. Conclusion

Deceptively powerful, copyless cost register automata with increments and min operations were shown to be able to simulate and check runs of counter machines. The constructions show that the repeated use of min enables behaviors that *appear* outside the scope of *copylessness*, e.g., an $\mathbb{N}_\infty$-CCRA can double the value of a register (or, more precisely, can attempt to do so while knowing when it failed). As a main consequence, equivalence of $\mathbb{N}_\infty$-CCRA is undecidable.

We wish to highlight three open questions:

(1) Theorem 19 comes short of telling us anything about the decidability of upper-boundedness for $\mathbb{Z}_\infty$-CCRA (the same being decidable for $\mathbb{N}_\infty$-CCRA and $\mathbb{N}$-WA in general [9]). Note that Proposition 10 shows that it cannot be decided whether a $\mathbb{Z}_\infty$-CCRA is upper-bounded by a *given* constant.
(2) Mazowiecki and Riveros [12] studied a restriction of CCRA in which, in particular, each register is either always updated with increments or with minimums. Our constructions rely heavily on the ability to alternate these two operations; we conjecture that equivalence is decidable for this restriction of CCRA over $\mathbb{N}$.
(3) Since CCRA are effectively expressible as linearly-ambiguous weighted automata (Corollary 3), one can decide, using a result of Kirsten and Lombardy [11], whether a CCRA is expressible as a *deterministic* weighted automaton. The precise complexity of that problem is however open.

**Acknowledgments.** We would like to thank Ismaël Jecker, Andreas Krebs, and James Worrell for stimulating discussions. The first author has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 837327. The third author is supported by the French National Research Agency (ANR) within the "Investments for the future" Programme IdEx Bordeaux (ANR-10-IDEX-03-02).

## Bibliography

[1] S. Almagor, U. Boker and O. Kupferman, What's decidable about weighted automata?, *ATVA 2011*, (2011), pp. 482–491.
[2] R. Alur and P. Cerný, Expressiveness of streaming string transducers, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, (2010), pp. 1–12.

[3] R. Alur, L. D'Antoni, J. V. Deshmukh, M. Raghothaman and Y. Yuan, Regular functions and cost register automata, *LICS 2013*, (2013), pp. 13–22.
[4] R. Alur and D. L. Dill, A theory of timed automata, *Theor. Comput. Sci.* **126**(2) (1994) 183–235.
[5] M. Benedikt, T. Duff, A. Sharad and J. Worrell, Polynomial automata: Zeroness and applications, *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, (2017), pp. 1–12.
[6] M. Blondin, C. Haase and F. Mazowiecki, Affine extensions of integer vector addition systems with states, *29th International Conference on Concurrency Theory, CONCUR 2018, September 4-7, 2018, Beijing, China*, (2018), pp. 14:1–14:17.
[7] M. Cadilhac, A. Finkel and P. McKenzie, Unambiguous constrained automata, *Int. J. Found. Comput. Sci.* **24**(7) (2013) 1099–1116.
[8] S. Gaubert and R. Katz, Rational semimodules over the max-plus semiring and geometric approach to discrete event systems, *Kybernetika* **40**(2) (2004) 153–180.
[9] K. Hashiguchi, Limitedness theorem on finite automata with distance functions, *Journal of Computer and System Sciences* **24**(2) (1982) 233 – 244.
[10] M. Kaminski and N. Francez, Finite-memory automata, *Theor. Comput. Sci.* **134**(2) (1994) 329–363.
[11] D. Kirsten and S. Lombardy, Deciding unambiguity and sequentiality of polynomially ambiguous min-plus automata, *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, (2009), pp. 589–600.
[12] F. Mazowiecki and C. Riveros, Copyless cost-register automata: Structure, expressiveness, and closure properties, *STACS 2016*, (2016), pp. 53:1–53:13.
[13] M. L. Minsky, Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines, *Annals of Mathematics* **74**(3) (1961) pp. 437–455.
[14] C. A. Petri, Kommunikation mit automaten, PhD thesis, Universität Hamburg (1962).
[15] M. Presburger, Über de vollständigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchen, die addition als einzige operation hervortritt, *Comptes Rendus du Premier Congrès des Mathématiciens des Pays Slaves*, Warsaw (1927), pp. 92–101.
[16] G. Sénizergues, Sequences of level 1, 2, 3, ..., $k$ , .., *Computer Science - Theory and Applications, Second International Symposium on Computer Science in Russia, CSR 2007, Ekaterinburg, Russia, September 3-7, 2007, Proceedings*, (2007), pp. 24–32.
[17] A. Weber and H. Seidl, On the degree of ambiguity of finite automata, *Theoretical Computer Science* **88**(2) (1991) 325 – 349.