Copyless Cost-Register Automata: Structure, Expressiveness, and Closure Properties

Filip Mazowiecki[1] **and Cristian Riveros**[2]

1 **University of Warsaw, Poland**
2 **Pontificia Universidad Católica de Chile, Chile**

Abstract

Cost register automata (CRA) and its subclass, copyless CRA, were recently proposed by Alur et al. as a new model for computing functions over strings. We study structural properties, expressiveness, and closure properties of copyless CRA. We show that copyless CRA are strictly less expressive than weighted automata and are not closed under reverse operation. To find a better class we impose restrictions on copyless CRA, which ends successfully with a new robust computational model that is closed under reverse and other extensions.

1998 ACM Subject Classification F.4.1. Computational logic

Keywords and phrases Cost Register Automata, Weighted Automata, Semirings

Digital Object Identifier 10.4230/LIPIcs.STACS.2016.53

1 Introduction

Weighted automata (WA) are an expressible extension of finite state automata for computing functions over strings. They have been extensively studied since Schützenberger [17], and its decidability problems [12, 1], extensions [7], logic characterization [7, 11], and applications [14, 6] have been deeply investigated.

Unfortunately, there exists a lack of research on understanding subclasses of functions below the class of WA. Recently Alur et al. [2, 4] introduced the computational model of cost register automata (CRA), an alternative model for computing functions over strings. The idea of this model is to enhance deterministic finite automata with registers that can be combined by using operations over a fixed semiring. In contrast to previous models with counters, CRA blindly updates its registers on each transition by using values computed on the previous state. In [2], it was shown that CRA are strictly more expressive than WA. Interestingly, it was also shown that a natural subfragment of CRA is equally expressive to WA, which gives a new representation for understanding this class of functions.

A new representation for WA allows to study natural subclasses of functions that could not be proposed from the classical perspective. This is the case for the class of copyless CRA that where proposed in [2]. The idea of the so-called copyless restriction is to use each register at most once in every transition. Intuitively, this automaton model is register-deterministic in the sense that it cannot copy the content of each register, similar to a deterministic finite automaton that cannot make a copy of its current state. The copyless restriction was successfully used in the context of streaming tree transducers [3] for capturing MSO-transductions over trees and it was proposed as a natural restriction over CRA. Furthermore, copyless CRA are an excellent candidate for having good decidability properties. It was stated in [2] that the existing proofs of undecidability in WA rely on the unrestricted non-deterministic nature of the model and, thus, it is believed that copyless CRA might have good decidability properties. Despite that this is a natural and interesting model for computing

functions, research on this line has not been pursued further and not much is known about copyless CRA.

In this paper we study the structure, expressiveness, and closure properties of copyless CRA. We start by developing a toolkit of structural properties for analysing copyless CRA. Towards this goal, we introduce a *normal form* on the registers of copyless CRA. We show that every copyless CRA can be put in this normal form which considerably simplifies the analysis of this model. With this restriction we provide further results that explain the flow and grow of registers content during a run. Specifically, we prove that from its normal form one can identify a subset of registers, called *stable* registers, that cannot be reset and are constantly growing during a run. We show that stable registers lead the behaviour of copyless CRA and that they are crucial to analyse the growing rate of loops.

Then we turn our attention on studying the expressivity of copyless CRA. As a proof of concept, we use the structural properties developed in this paper to compare the expressiveness of copyless CRA with the class of functions defined by WA. We show that copyless CRA are strictly less expressive than WA. Furthermore, we show that copyless CRA are strictly less expressive than regular cost functions, a class of functions introduced in [2]. It is important to stress that it was previously believed that copyless CRA was strictly less expressive than WA, but this is the first paper which proves this statement formally.

In the last sections, we focus on the robustness of copyless CRA in terms of its closure properties. The robustness of a computational model is usually measured in terms of how stable is the model when new operations or extensions are allowed. Deterministic finite automata are a good example for the previous statement: they are closed under several extensions/operations like set operations (e.g union, intersection), different flavours of non-determinism, regular look-ahead, two-directions (in particular, under reverse), etc. These properties are probably one of the reasons behind its fruitful connection with MSO logic or finite monoids [5, 16]. Unfortunately, this measure of robustness put copyless CRA in an undesirable position: our expressiveness result shows that copyless CRA are not closed under reverse and, furthermore, under any extensions regarding directions of its reading head. This implies that the behaviour of copyless CRA is asymmetric with respect to the input, which buried our expectations of a robust class for computing functions.

The lack of good closure properties for copyless CRA fuels our interest in its subclasses. We consider a natural fragment of copyless CRA, called *bounded alternation copyless* CRA (BAC). This class was previously introduced in [13] and characterized in terms of the so-called *Maximal Partition logic*. Interestingly, this fragment of copyless CRA does not lose much in terms of expressivity. In fact, most examples in [2] and in this paper are definable by BAC. Furthermore, all the structural toolkit introduced for copyless CRA also extend for this class. In contrast to copyless CRA, BAC are robust under several and natural extensions previously considered in [2, 3]. Specifically, we show that BAC are closed under unambiguous non-determinism, regular look-ahead and under reverse. These results emphasize that BAC is a promising computational model in the world of quantitative functions and show that there exists a rich theory of functions below the class of WA.

Organization. In Section 2 we introduce copyless CRA and some basic definitions. In Section 3 we introduce the normal form and analyze the content of registers during the runs of copyless CRA. Then we show in Section 4 that the class of copyless CRA is not closed under reverse. In Section 5 we define BAC and show some closure properties of this class. We conclude in Section 6 with possible directions for future research. Due to the page limit all full proofs are moved to the appendix, available online.

2 Preliminaries

In this section, we recall the definitions of cost register automata and the copyless restriction. We start with the definitions of expressions and substitutions over a semiring that are standard in this area.

Semirings and functions. A semiring is a structure $\mathbb{S} = (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$ where $(S, \oplus, \mathbb{0})$ is a commutative monoid, $(S - \{\mathbb{0}\}, \odot, \mathbb{1})$ is a monoid, multiplication distributes over addition, and $\mathbb{0} \odot s = s \odot \mathbb{0} = \mathbb{0}$ for each $s \in S$. If the multiplication is commutative, we say that $\mathbb{S}$ is commutative. In this paper, we always assume that $\mathbb{S}$ is commutative. For the sake of simplicity, we usually denote the set of elements S by the name of the semiring $\mathbb{S}$. As standard examples of semirings we will consider the *semiring of natural numbers* $\mathbb{N}(+,\cdot) = (\mathbb{N}, +, \cdot, 0, 1)$, the *min-plus semiring* $\mathbb{N}_\infty(\min, +) = (\mathbb{N}_\infty, \min, +, \infty, 0)$ and the *max-plus semiring* $\mathbb{N}_{-\infty}(\max, +) = (\mathbb{N}_{-\infty}, \max, +, -\infty, 0)$ which are standard semirings in the field of weighted automata [8].

In this paper, we study the specification of functions from strings to values, namely, from Σ^* to $\mathbb{S}$. We say that a function $f : \Sigma^* \to \mathbb{S}$ is definable by a computational system $\mathcal{A}$ (e.g. weighted automaton, or CRA) if $f(w) = [\![\mathcal{A}]\!](w)$ for any $w \in \Sigma^*$, where $[\![\mathcal{A}]\!]$ is the semantics of $\mathcal{A}$ over strings. For any string w, we denote by w^r the reverse string. We say that a class of functions F is *closed under reverse* [2] if for every $f \in F$ there exists a function $f^r \in F$ such that $f^r(w^r) = f(w)$ for all $w \in \Sigma^*$.

Variables, expressions, and substitutions. Fix a semiring $\mathbb{S} = (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$ and a set of variables $\mathcal{X}$ disjoint from S. We denote by $\mathrm{Expr}(\mathcal{X})$ the set of all syntactical *expressions* that can be defined with $\mathcal{X}$, constants in S, and the binary operations $\oplus, \odot$. For any expression $e \in \mathrm{Expr}(\mathcal{X})$ we denote by $\mathrm{Var}(e)$ the set of variables in e. We call an expression $e \in \mathrm{Expr}(\mathcal{X})$ a *ground expression* if $\mathrm{Var}(e) = \varnothing$. For any ground expression we define $[\![e]\!] \in \mathbb{S}$ to be the evaluation of e with respect to $\mathbb{S}$.

A *substitution* over $\mathcal{X}$ is defined as a mapping $\sigma : \mathcal{X} \to \mathrm{Expr}(\mathcal{X})$. We denote the set of all substitutions over $\mathcal{X}$ by $\mathrm{Subs}(\mathcal{X})$. A *ground substitution* σ is a substitution where the expression $\sigma(x)$ is ground for every $x \in \mathcal{X}$. Any substitution σ can be extended to a mapping $\hat{\sigma} : \mathrm{Expr}(\mathcal{X}) \to \mathrm{Expr}(\mathcal{X})$ such that, for every $e \in \mathrm{Expr}(\mathcal{X})$, $\hat{\sigma}(e)$ is the resulting expression $e[\sigma]$ of substituting each $x \in \mathrm{Var}(e)$ by the expression $\sigma(x)$. For example, if $\sigma(x) = 2x$ and $\sigma(y) = 3y$, and $e = x + y$, then $\hat{\sigma}(e) = 2x + 3y$. Using the extension $\hat{\sigma}$, we can define the composition substitution $\sigma_1 \circ \sigma_2$ of two substitutions σ_1 and σ_2 such that $\sigma_1 \circ \sigma_2(x) = \hat{\sigma}_1(\sigma_2(x))$ for each $x \in \mathcal{X}$.

A valuation is defined as a substitution of the form $\nu : \mathcal{X} \to \mathbb{S}$. We denote the set of all valuations over $\mathcal{X}$ by $\mathrm{Val}(\mathcal{X})$. Clearly, any valuation ν composed with a substitution σ defines a ground substitution, since $\mathrm{Var}(\nu \circ \sigma(x)) = \varnothing$ for all $x \in \mathcal{X}$.

In this paper, we say that two expressions e_1 and e_2 are equal (denoted by $e_1 = e_2$) if they are equal up to evaluation equivalence, that is, $[\![\nu \circ e_1]\!] = [\![\nu \circ e_2]\!]$ for every valuation $\nu \in \mathrm{Val}(\mathcal{X})$. Similarly, we say that two substitutions σ_1 and σ_2 are equal (denoted by $\sigma_1 = \sigma_2$) if $\sigma_1(x) = \sigma_2(x)$ for every $x \in \mathcal{X}$.

Cost register automata. A cost register automaton (CRA) over a semiring $\mathbb{S}$ [2] is a tuple $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \delta, q_0, \nu_0, \mu)$ where Q is a set of states, Σ is the input alphabet, $\mathcal{X}$ is a set of variables (we also call them registers), $\delta : Q \times \Sigma \to Q \times \mathrm{Subs}(\mathcal{X})$ is the transition function, q_0 is the initial state, $\nu_0 : \mathcal{X} \to \mathbb{S}$ is the initial valuation, and $\mu : Q \to \mathrm{Expr}(\mathcal{X})$ is the final

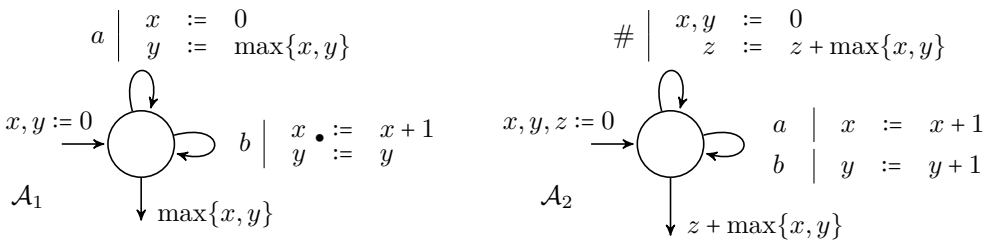


Figure 1 Examples of copyless cost-register automata.

output function. A configuration of $\mathcal{A}$ is a tuple (q, ν) where $q \in Q$ and $\nu \in \mathrm{Val}(\mathcal{X})$ represents the current values in the variables of $\mathcal{A}$. Given a string $w = a_1 \dots a_n \in \Sigma^*$, the run of $\mathcal{A}$ over w is a sequence of configurations: $(q_0, \nu_0) \xrightarrow{a_1} (q_1, \nu_1) \xrightarrow{a_2} \dots \xrightarrow{a_n} (q_n, \nu_n)$ such that, for every $1 \le i \le n$, $\delta(q_{i-1}, a_i) = (q_i, \sigma_i)$ and $\nu_i(x) = [\![\nu_{i-1} \circ \sigma_i(x)]\!]$ for each $x \in \mathcal{X}$. The output of $\mathcal{A}$ over w, denoted by $[\![\mathcal{A}]\!](w)$, is $[\![\nu_n \circ \mu(q_n)]\!]$.

The run of $\mathcal{A}$ over w can be equally defined in terms of ground expressions rather than values. A ground configuration of $\mathcal{A}$ is a tuple (q, ς) where $q \in Q$ and $\varsigma \in \mathrm{Subs}(\mathcal{X})$ is a ground substitution. Given a string $w = a_1 \dots a_n \in \Sigma^*$, the ground run of $\mathcal{A}$ over w is a sequence of ground configurations: $(q_0, \varsigma_0) \xrightarrow{a_1} \dots \xrightarrow{a_n} (q_n, \varsigma_n)$ such that for $1 \le i \le n$, $\delta(q_{i-1}, a_i) = (q_i, \sigma_i)$, $\varsigma_0 = \nu_0$ and $\varsigma_i(x) = \varsigma_{i-1} \circ \sigma_i(x)$ for each $x \in \mathcal{X}$. We denote the output ground expression of $\mathcal{A}$ over a string w by $|\mathcal{A}|(w) = \varsigma_n \circ \mu(q_n)$. Notice that, in contrast to ordinary runs, ground runs keep ground expressions as partial values of the run. It is easy to see that $[\![\mathcal{A}]\!](w) = [\![|\mathcal{A}|(w)]\!]$.

We define the transitive closure of the transition function $\delta^* : Q \times \Sigma^* \to Q \times \mathrm{Subs}(\mathcal{X})$, by induction over the word-length. Formally, $\delta^*(q, \epsilon) = (q, \mathrm{id})$ where ϵ is the empty word and $\mathrm{id}(x) = x$ for all $x \in \mathcal{X}$; and $\delta^*(q_1, w \cdot a) = (q_3, \sigma \circ \sigma')$, whenever $\delta^*(q_1, w) = (q_2, \sigma)$ and $\delta(q_2, a) = (q_3, \sigma')$. For a CRA $\mathcal{A}$ we define the set $\mathrm{Subs}(\mathcal{A})$ of all substitutions in $\mathcal{A}$ such that $\sigma \in \mathrm{Subs}(\mathcal{A})$ if, and only if, $\delta^*(p, w) = (q, \sigma)$ for some $p, q \in Q$ and $w \in \Sigma^*$.

Copyless restriction and copyless CRA. We say that an expression $e \in \mathrm{Expr}(\mathcal{X})$ is *copyless* if e uses every variable from $\mathcal{X}$ at most once. For example, $x \cdot (y + z)$ is copyless but $x \cdot y + x \cdot z$ is not copyless (because x is mentioned twice). Notice that the copyless restriction is a syntactical constraint over expressions. Furthermore, we say that a substitution σ is *copyless* if for every $x \in \mathcal{X}$ the expression $\sigma(x)$ is copyless and $\mathrm{Var}(\sigma(x)) \cap \mathrm{Var}(\sigma(y)) = \varnothing$ for every pair of different registers $x, y \in \mathcal{X}$. Copyless substitutions, similar to copyless expressions, are restricted in such a way that each variable is used at most once in the whole substitution.

A CRA $\mathcal{A}$ is called *copyless* if for every transition $\delta(q_1, a) = (q_2, \sigma)$ the substitution σ is copyless; and for every state $q \in Q$ the expression $\mu(q)$ is copyless, where μ is the output function of $\mathcal{A}$. In other words, every time $\mathcal{A}$ updates registers or outputs a value, each register can be used only once. It is straightforward that if $\mathcal{A}$ is copyless then all substitutions $\sigma \in \mathrm{Subs}(\mathcal{A})$ are also copyless. In the following, we give an example of a copyless CRA.

▶ **Example 1.** Let $\mathbb{S}$ be the max-plus semiring $\mathbb{N}_{-\infty}(\max, +)$ and $\Sigma = \{a, b\}$. Consider the function f_1 that for a given string $w \in \Sigma^*$ computes the longest substring of b's. This can be easily defined by the CRA $\mathcal{A}_1$ in Figure 1. The CRA $\mathcal{A}_1$ stores in the x-register the length of the last suffix of b's and in the y-register the length of the longest substring of b's seen so far. One can easily check that $\mathcal{A}_1$ is a copyless CRA. Indeed, each substitution is copyless and the final output expression $\max\{x, y\}$ is copyless as well.

▶ **Example 2.** Let $\mathbb{S}$ be the max-plus semiring $\mathbb{N}_{-\infty}(\max, +)$ and $\Sigma = \{a, b, \#\}$. Consider the function f_2 such that, for any $w \in \Sigma^*$ of the form $w_0 \# w_1 \# \dots \# w_n$ with $w_i \in \{a, b\}^*$, it

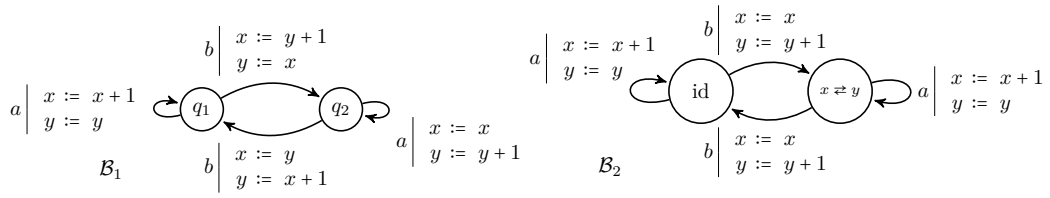


Figure 2 Examples of copyless cost-register automata regarding normal form.

computes the maximum number of a's or b's for each substring w_i (i.e. $\max\{|w_i|_a, |w_i|_b\}$) and then it sums these values over all substrings w_i, that is, $f_2(w) = \sum_{i=0}^{n} \max\{|w_i|_a, |w_i|_b\}$. One can check that the copyless CRA $\mathcal{A}_2$ in Figure 1 computes f_2. The copyless CRA $\mathcal{A}_2$ follows similar ideas to $\mathcal{A}_1$: the registers x and y count the number of a's and b's, respectively, in the longest suffix without # and the register z stores the partial output without considering the last suffix of a's and b's.

In the diagram of $\mathcal{A}_2$, we omit an assignment if a register is not updated (i.e. it keeps its previous value). For example, for the a-transition we omit the assignments $y := y$ and $z := z$ for the sake of presentation of the CRA. One should keep in mind these assignments because of the copyless restriction.

Trim assumption. For technical reasons, in this paper we assume that our finite automata and cost register automata are always *trim*, namely, all their states are reachable from some initial states (i.e., they are accessible) and they can reach some final state (i.e., they are co-accessible). It is worth noticing that verifying if a state is accessible or co-accessible is reduced to a reachability test in the transition graph [15] and this can be done in NLogSpace. Thus, we can assume without lost of generality that all our automata are trimmed.

3 Structural Properties of Copyless CRA

Fix a copyless CRA $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \delta, q_0, \nu_0, \mu)$. In this section we analyze the structure of $\mathcal{A}$ and develop some machinery that will be useful in Section 4. These results help to understand the internal structure of copyless CRA.

Normal form. Let $\mathcal{A}$ be a copyless CRA and let $\preceq$ be a predefined linear order over $\mathcal{X}$. We say that $\sigma \in \text{Subs}(\mathcal{A})$ is in normal form with respect to $\preceq$ if $x \preceq y$ for all $x \in X$ and all $y \in \text{Var}(\sigma(x))$. In other words, all variables mentioned in $\sigma(x)$ are greater or equal to x with respect to $\preceq$. Furthermore, we write that $\mathcal{A}$ is in *normal form* with respect to $\preceq$ if every $\sigma \in \text{Subs}(\mathcal{A})$ is in normal form with respect to $\preceq$. For example, the copyless CRA in Example 1 is in normal form with respect to the order $y \preceq x$. Throughout the paper we assume that every set of registers $\mathcal{X}$ is given with a linear order $\preceq_{\mathcal{X}}$. Instead of writing that $\mathcal{A}$ or $\sigma \in \text{Subs}(\mathcal{A})$ are in normal form with respect to $\preceq_{\mathcal{X}}$ we write in short that $\mathcal{A}$ is in normal form, and that σ is in normal form.

▶ **Example 3.** Consider the set of registers $\mathcal{X} = \{x, y\}$ with the order $x \preceq y$. The copyless CRA $\mathcal{B}_1$ in Figure 2 is not in normal form, because of the b-transitions. On the other hand, the copyless CRA $\mathcal{B}_2$ is in normal form. For both automata we omit the initial states, initial valuations and final output functions because they are not relevant for the discussion.

In the previous example, the automaton $\mathcal{B}_1$ uses the registers x and y to count the number of a's and b's. However, depending on the current state both registers have either

the number of a's or the number of b's. It is clear that one would like to avoid this type of behavior in a theoretical analysis. Intuitively, one register should always contain the number of a's and the other register the number of b's. One can clearly transform $\mathcal{B}_1$ to an automaton in normal form by exchanging the use of x and y in the transitions and encoding this permutation between registers in the states. This is precisely what the automaton $\mathcal{B}_2$ does. In the following result, we generalize this idea for all copyless CRA.

▶ **Proposition 4.** *For every copyless CRA $\mathcal{A}$ there exists a copyless CRA in normal form $\mathcal{A}'$ with the same set of registers such that they output the same ground expressions for all words and thus recognize the same function. The number of states in $\mathcal{A}'$ can be bounded exponentially in the size of the automaton $\mathcal{A}$.*

Proof (sketch). For a copyless CRA $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \delta, q_0, \nu_0, \mu)$, we define the set of states for $\mathcal{A}'$ by $Q' = Q \times \{\rho \mid \rho \text{ is a permutation of the set } \mathcal{X}\}$. Let $\delta(q, a) = (p, \sigma)$ be a transition. The idea is to find a permutation ρ such that the new substitution $\sigma'(x) = \sigma(\rho(x))$ is a substitution in normal form. Such a permutation always exists because $\mathcal{A}$ satisfies the copyless restriction. Intuitively, the substitutions σ' and σ store the same values in the registers, but the corresponding values might be in different registers. To enable $\mathcal{A}'$ to find the proper value of every register, we store the permutation ρ in the state p (i.e., the one defined by the substitution σ) and update the transition of $\mathcal{A}$ accordingly. ◀

Stable registers and reset substitutions. Let now $\mathcal{A}$ be a copyless CRA in normal form. During a run of $\mathcal{A}$ the content of its registers flows from higher to lower registers with respect to the total order $\preceq$. This does not necessarily mean that the content of all registers eventually reaches the $\preceq$-minimum register. For example, if all substitutions in $\mathcal{A}$ are of the form $\sigma(x) = x \oplus k$ for some $k \in \mathbb{S}$, then each register will store just its own content during the whole run. Intuitively, in this example each register is "stable" with respect to the content flow of $\mathcal{A}$, since each register never passes its value to lower registers. This idea motivates the notion of *stable registers* which are essential to understand the behaviour and output of copyless CRA. Let $\sigma \in \mathrm{Subs}(\mathcal{A})$ be a copyless substitution in normal form. We say that a register x is *σ-stable* (or stable on σ) if $x \in \mathrm{Var}(\sigma(x))$. More general, we write that a register x is *stable* in $\mathcal{A}$ if x is σ-stable for all substitutions $\sigma \in \mathrm{Subs}(\mathcal{A})$.

Stable registers play a crucial role in the behavior of copyless CRA. We show that they are the only registers whose value always depends on the whole word. Namely, we can always "reset" the value of non-stable registers to a constant. For instance, the automaton $\mathcal{A}_1$ in Example 1 resets the register x to 0 each time the symbol a is read. On the other side, the register y is stable and it cannot be reset to a constant. In fact its value only grows or remains the same during the run of $\mathcal{A}_1$.

We formalize this idea of reseting the content of registers as follows: a substitution $\sigma \in \mathrm{Subs}(\mathcal{A})$ is a *reset* substitution if $\mathrm{Var}(\sigma(x)) = \varnothing$ for all non σ-stable registers x. We say that a substitution $\sigma \in \mathrm{Subs}(\mathcal{A})$ is an *$\mathcal{A}$-reset substitution* if $\mathrm{Var}(\sigma(x)) = \varnothing$ for all non-stable registers in $\mathcal{A}$.

In the next result, we say that a copyless CRA is strongly connected if all states are mutually reachable in the transition graph of $\mathcal{A}$.

▶ **Proposition 5.** *Let $\mathcal{A}$ be a copyless and strongly connected CRA in normal form. Then for all $q, q' \in Q$ there exists $w^{q,q'} \in \Sigma^*$ and a substitution σ such that $\delta^*(q, w^{q,q'}) = (q', \sigma)$ and σ is an $\mathcal{A}$-reset substitution. Furthermore, there exists $w^{q,q'}$ containing all letters in Σ.*

The strongly connected restriction is a technical assumption to be sure that the result holds for every pair of states. Of course, the result also holds if we restrict to the strongly connected

components of $\mathcal{A}$. For instance in Example 1 it suffices to take the word $w = a$ given that the substitution defined by the a-transition is an $\mathcal{A}$-reset substitution.

Growing rate of stable registers in a cycle. The behavior of cycles in a computation model is always important; most of the decidability results can be derived from a good understanding of its cyclic behavior. Here, we study how the content of stables registers behaves through cycles. We say that a word $w \in \Sigma^*$ is a cycle over a state $q \in Q$ in $\mathcal{A}$ if $\delta^*(q, w) = (q, \sigma)$ for some substitution σ. Of course, the iteration of a cycle w (i.e. w^n for any $n \geq 1$) is also a cycle over q and it satisfies $\delta^*(q, w^n) = (q, \sigma^n)$ for any n. The next result shows that by iterating a cycle one can always "reset" the content of non σ-stable registers.

▶ **Lemma 6.** *Let $\mathcal{A}$ be in normal form and $\sigma \in \mathrm{Subs}(\mathcal{A})$ a copyless substitution in normal form. There exists $N \geq 0$ such that σ^N is a reset substitution.*

In the next proposition, we study the growing behavior of stable registers when a reset substitution is iterated. This will be useful to understand the behavior of copyless CRA inside their cycles. For this result we need an additional assumption that automata do not use $\mathbb{0}$ in their expressions. Moreover the expressions in our semirings that do not use $\mathbb{0}$ do not evaluate to $\mathbb{0}$. This is a technical assumption to avoid having $\mathbb{0}$ in the registers of CRA. Notice that the arctic-semiring $\mathbb{N}_{-\infty}(\max, +)$ satisfies $\mathbb{0} = -\infty$ and that the functions we defined in Section 4 do not output $-\infty$. Also in $\mathbb{N}_{-\infty}(\max, +)$ expressions defined with max, + and $\mathbb{N}$ do not evaluate to $-\infty$. Thus this assumption does not influence the results in Section 4.

▶ **Proposition 7.** *Let $\mathcal{A}$ be in normal form, $\sigma \in \mathrm{Subs}(\mathcal{A})$ a reset substitution and x a σ-stable register. Then there exist $c, d \in \mathbb{S}$ with $c \neq \mathbb{0}$ such that for every $i \geq 0$ we have:*

$$\sigma^{i+1}(x) \quad = \quad (c^i \odot \sigma(x)) \oplus \Big(d \odot \bigoplus_{j=0}^{i-1} c^j\Big).$$

In particular, for the semiring $\mathbb{N}_{-\infty}(\max, +)$ (i.e. $\odot = +$ and $\oplus = \max$) one can check that this is equivalent to $\sigma^{i+1}(x) = \max\{i \cdot c + \sigma(x),\ (i-1) \cdot c + d\}$, which shows that, intuitively, the σ-stable registers grow linearly when the substitution σ is cycling. For instance, consider the copyless CRA in Example 1 and let σ be the substitution defined by the b-transition. One can easily check that the register x is σ-stable since $\sigma(x) = x + 1$. Furthermore, $\sigma^{i+1}(x) = x + i + 1$, which is as in Proposition 7 (take $c = 1, d = 0$). This is precisely the expected behaviour of the copyless CRA on a long sequence of b's.

4 Inexpressibility of Copyless CRA

In this section, we use the techniques discussed previously to show a function that is not definable by any copyless CRA. This function can be defined by a weighted automaton, which will prove that copyless CRA are less expressive than weighted automata. Interestingly, the "reverse" of this function is definable by copyless CRA. From this, we will conclude that copyless CRA are not closed under reverse.

Consider the function $f_{\mathcal{B}}$ given by the copyless CRA $\mathcal{B}$ over $\Sigma = \{a, b, \#\}$ and $\mathbb{N}_{-\infty}(\max, +)$ in Figure 3. To understand $f_{\mathcal{B}}$, let us define the output of $\mathcal{B}$ formally. For any $w \in \Sigma^*$, let k be the number of #-symbols in w. Furthermore, for $0 < i < k$ let n_i and m_i be the number of a's and b's, respectively, between the i-th and $(i+1)$-th occurrence of # in w. Additionally, let n_0, m_0, n_k, m_k be the numbers of a's and b's before and after the first and last # in w.

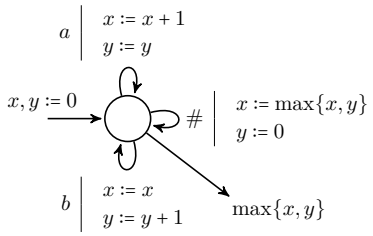


Figure 3 CRA $\mathcal{B}$.

By the definition of $\mathcal{B}$ in Figure 3, one can easily check that $f_{\mathcal{B}}$ is defined by $f_{\mathcal{B}}(\epsilon) = 0$, for the empty word ϵ, and

$$f_{\mathcal{B}}(w) \quad = \quad \max_{j \in \{-1,0,\ldots,k\}} \left\{ m_j + \sum_{i=j+1}^{k} n_i \right\} \tag{1}$$

for $w \neq \epsilon$, where $m_{-1} = 0$. From the above definition, one can also give a formal definition of $f^R_{\mathcal{B}}$, the reverse function of $f_{\mathcal{B}}$, which is given by reversing the role of n_i and m_j in (1). Formally, one can easily check that $f^R_{\mathcal{B}}$ is defined by $f^R_{\mathcal{B}}(\epsilon) = 0$, and

$$f^R_{\mathcal{B}}(w) \quad = \quad \max_{j \in \{0,\ldots,k,k+1\}} \left\{ \sum_{i=0}^{j-1} n_i + m_j \right\}, \tag{2}$$

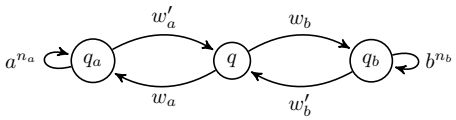
for $w \neq \epsilon$, where $m_{k+1} = 0$. The following theorem is the main result of this section.

▶ **Theorem 8.** *The function $f^R_{\mathcal{B}}$ is not recognizable by any copyless CRA.*

Proof (sketch). By contradiction, suppose that $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \delta, q_0, \nu_0, \mu)$ is a copyless CRA in normal form with respect to $\preceq$, which recognizes (2). To prove Theorem 8, we analyze the content of the registers after reading the word $w_0 \cdot w(j,s)$, where

$$w(j,s) = (w_a \cdot (a^{n_a \cdot j}) \cdot w'_a)^s \cdot w_b \cdot (b^{n_b \cdot j^2}) \cdot w'_b \cdot w_a \cdot (a^{n_a \cdot j}) \cdot w'_a.$$

for $j, s \in \mathbb{N}$. To understand the construction of $w(j,s)$, consider the following diagram, which is a fragment of $\mathcal{A}$.



First, the prefix w_0 is used to reach the connected component of $\mathcal{A}$ to make the Proposition 5 work and q is the state that $\mathcal{A}$ reaches after reading w_0. By a standard automata argument, one can find a state q_a such that for some number n_a the word a^{n_a} is a cycle over q_a. Similarly there is a state q_b and a number n_b such that b^{n_b} is a cycle over q_b. The words w_a and w_b go from q to the states q_a and q_b, respectively, and the words w'_a and w'_b go back to q. To analyze the run of $\mathcal{A}$ on $w(j,s)$ we need only the component above.

The next step is to study the asymptotic growing of $f^R_{\mathcal{B}}$ over $w_0 \cdot w(j,s)$ in terms of j and s. For this, the most important fragments of $w(j,s)$ are the subwords $a^{n_a \cdot j}$ and $b^{n_b \cdot j^2}$. To omit technical difficulties we do not discuss further here the remaining parts of $w(j,s)$. In a nutshell, the words w_a, w_b, w'_a, and w'_b are the reset words discussed in Proposition 5. Recall that the size of these words does not depend on j and s and they contain all symbols in Σ, in particular #.

We can estimate the output of $f_{\mathcal{B}}^R$ over $w_0 \cdot w(j,s)$ in terms of j and s by using (2). Given that the subwords $a^{n_a \cdot j}$ and $b^{n_b \cdot j^2}$ are separated by # (by Proposition 5), we conclude that the last sequence of a's is not included in the sum. This is because the maximum in (2) is achieved when m_j contains the subword $b^{n_b \cdot j^2}$ and thus the last sequence of a's will not be included in the final sum. Then one can prove that the value of the output of $f_{\mathcal{B}}^R$ over $w(j,s)$ is equal to

$$f_{\mathcal{B}}^R(w) = n_b \cdot j^2 + s \cdot n_a \cdot j + \mathcal{O}(1).$$

There are some a's and b's in the remaining parts of $w(j,s)$ and w_0 that contribute to the output, but these are constant terms that can be hidden in $\mathcal{O}(1)$.

By Lemma 6 we can take n_a big enough such that if $\delta^*(q_a, a^{n_a}) = (q_a, \sigma_a)$ then the substitution σ_a is a reset substitution. Let x be a stable register. By Proposition 7 we can approximate the expressions $\sigma_a^{j+1}(x)$ for j big enough. In the semiring $\mathbb{N}_{-\infty}(\max, +)$ the content of x when reading this loop can be estimated by

$$\sigma_a^{j+1}(x) = \max\{j \cdot c_x + \sigma_a(x) + \mathcal{O}(1),\ j \cdot c_x + \mathcal{O}(1)\}$$

for some constant c_x. Intuitively, this means that for j big enough after reading $a^{n_a \cdot j}$ the content of every stable register x is growing linearly to j or the content of x becomes linear in j. Similarly we can estimate the content of stable registers after reading b^{j^2}. The word $w(j,s)$ ends with w'_a, which resets the content of all non-stable registers to a constant, thus we can estimate their content with $\mathcal{O}(1)$.

Consider now j as a variable and s as a fixed parameter in $w(j,s)$. By the previous analysis the contents of the stable registers after reading $w(j,s)$ are quadratic functions in j, where the coefficient of j comes from the growth of registers when reading the subwords $a^{n_a \cdot j}$. We show that in every register either the coefficient of j is small compared to $s \cdot n_a$, or linear in $(s+1) \cdot n_a$. In both cases the output function cannot combine the registers to get a polynomial with $s \cdot n_a$ as a coefficient of j. Finally, for any $\mathcal{A}$ we can fix the parameter s to be big enough comparing to the number of registers. ◀

From Theorem 8 we immediately get the following corollary.

▶ **Corollary 9.** *There exists a semiring $\mathbb{S}$ such that the class of functions recognizable by copyless register automata over $\mathbb{S}$ is not closed under reverse.*

In [13] it is shown that copyless CRA are contained in weighted automata (WA). The previous results delimit the expressiveness of copyless CRA: they are strictly less expressive than WA. It is easy to define $f_{\mathcal{B}}^R$ by a polynomial ambiguous weighted automaton (i.e. a subclass of WA where the numbers of accepting runs is bounded by a polynomial function), which shows that copyless CRA is a strict subclass of weighted automata. Furthermore, in [2] Alur et al. introduced the class of *regular cost functions* defined in terms of streaming string-to-tree transducers (see [2] for more details). This class contained copyless CRA but it was left open whether this inclusion is strict or not. Since the class of regular cost functions is closed under reverse operation, Corollary 9 proves that copyless CRA are strictly contained in the class of regular cost functions.

▶ **Corollary 10.** *The class of functions defined by copyless CRA is strictly contained in the class of functions defined by weighted automata and in the class of regular cost functions.*

A natural question is whether the class of functions defined by copyless CRA is strictly contained in the class of functions defined by polynomial ambiguous weighted automata. We

conjecture that copyless CRA can express functions that are not expressible by polynomial ambiguous automata. A candidate function is the copyless CRA $\mathcal{A}_2$ in Example 2 for which it seems that one needs exponentially many runs in order to be computed by a weighted automaton.

5 Bounded Alternation Copyless CRA

Given that copyless CRA are not closed under reverse operation we look for a robust subclass of copyless CRA. The proof of Theorem 8 suggests that the alternation between semiring operations is the reason why copyless CRA cannot replicate the behavior of the CRA $\mathcal{B}$ in backward mode: $\mathcal{B}$ can sum the number of a- and b-symbols to maximize each time that a #-symbol is read, but it cannot do the same alternation of operations when the word is read in the other direction. This fact inspires the definition of *bounded alternation* copyless CRA, a strict subclass of copyless CRA where the output is restricted to expressions with bounded alternation. This class was proposed in [13] and characterized in terms of the so-called *Maximal Partition logic*. In this section, we show that bounded alternation copyless CRA has also good closure properties; this class is closed under unambiguous non-determinism, regular look-ahead and, moreover, under reverse.

The *alternation* of $e \in \mathrm{Expr}(\mathcal{X})$ is defined as the maximum number of switches between $\oplus$ and $\odot$ operations over all branches of the parse-tree of e. Formally, let $\otimes \in \{\oplus, \odot\}$ and $\bar{\otimes}$ be the dual operation of $\otimes$ in $\mathbb{S}$. We define the set of expressions $\mathrm{Expr}_0^{\otimes}(\mathcal{X})$ with 0-alternation by $\mathrm{Expr}_0^{\otimes} = \mathcal{X} \cup \mathbb{S}$. For any $N \geq 1$, we define the set of expressions $\mathrm{Expr}_N^{\otimes}(\mathcal{X})$ as the $\otimes$-closure of $\mathrm{Expr}_{N-1}^{\bar{\otimes}}(\mathcal{X})$, namely, $\mathrm{Expr}_N^{\otimes}(\mathcal{X})$ is the minimal set of expressions that contains $\mathrm{Expr}_{N-1}^{\bar{\otimes}}(\mathcal{X})$ and satisfies that $e_1 \otimes e_2 \in \mathrm{Expr}_N^{\otimes}(\mathcal{X})$ whenever $e_1, e_2 \in \mathrm{Expr}_N^{\otimes}(\mathcal{X})$. We define $\mathrm{Expr}_N(\mathcal{X}) = \mathrm{Expr}_N^{\oplus}(\mathcal{X}) \cup \mathrm{Expr}_N^{\odot}(\mathcal{X})$, namely, the set of all expressions with alternation bounded by N.

We say that a copyless CRA $\mathcal{A}$ has *bounded alternation* if the number of alternations of all ground expressions output by $\mathcal{A}$ is uniformly bounded by a constant, that is, there exists N such that $|\mathcal{A}|(w) \in \mathrm{Expr}_N(\mathcal{X})$ for every $w \in \Sigma^*$. A copyless CRA $\mathcal{A}$ is called a bounded alternation copyless CRA (BAC) if $\mathcal{A}$ has bounded alternation. All the examples of copyless CRA presented in Section 2 have bounded alternation. For example, one can easily check that the alternation of the copyless CRA in Example 1 is bounded by 2.

The alternation of any expression can be easily derived just by counting what is the maximum number of alternation between $\oplus$ and $\odot$. However, it is not directly clear how to check if a copyless CRA has bounded alternation from its definition. We show that this semantical property can be verified in NLogSpace in the size of a copyless CRA.

▶ **Proposition 11.** *The problem of deciding whether a copyless CRA has bounded alternation can be computed in* NLogSpace. *Furthermore, if a copyless CRA has bounded alternation, the alternation is bounded by* $|Q| \cdot \max\{\, \mathrm{alt}(\sigma) \mid \exists p, q \in Q.\ \delta(q, a) = (p, \sigma) \,\}$ *where* $\mathrm{alt}(\sigma)$ *is the alternation of* σ.

Closure under unambiguous non-determinism. We first extend the model of bounded alternation copyless CRA with non-determinism. The class CRA was designed as a deterministic model in contrast to weighted automata, where non-determinism plays a crucial role. Thus, we restrict non-determinism to be unambiguous, namely, we allow many runs over a word but at most one accepting run, which defines the output. Formally, a non-deterministic CRA is a tuple $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \Delta, I_0, V_0, F, \mu)$ where Q, Σ, $\mathcal{X}$, and μ are defined as before, $\Delta \subseteq Q \times \Sigma \times Q \times \mathrm{Subs}(\mathcal{X})$ is a finite transition relation, $I_0 \subseteq Q$ is a set of initial states,

$V_0 : I_0 \to \mathrm{Val}(\mathcal{X})$ assigns an initial valuation for each initial state, and F is the set of final states. Additionally, we assume that for every $q, q' \in Q$ and $a \in \Sigma$ there exists at most one $\sigma \in \mathrm{Subs}(\mathcal{X})$ such that $(q, a, q', \sigma) \in \Delta$. Given a string $w = a_1 \dots a_n \in \Sigma^*$, a run of $\mathcal{A}$ over w is a sequence of configurations: $(q_0, \nu_0) \xrightarrow{a_1} (q_1, \nu_1) \xrightarrow{a_2} \dots \xrightarrow{a_n} (q_n, \nu_n)$ such that $q_0 \in I_0$, $\nu_0 = V_0(q_0)$, and for $1 \leq i \leq n$, $(q_{i-1}, a_i, q_i, \sigma_i) \in \Delta$ and $\nu_i(x) = [\![\nu_{i-1} \circ \sigma_i(x)]\!]$ for each $x \in \mathcal{X}$. Furthermore, a run of $\mathcal{A}$ over w is an accepting run if $q_n \in F$. We say that $\mathcal{A}$ is unambiguous if for every $w \in \Sigma^*$ there exists exactly one accepting run of $\mathcal{A}$ over w. The output of $\mathcal{A}$ over w is defined as $[\![\mathcal{A}]\!](w) = [\![\nu_n \circ \mu(q_n)]\!]$ where (q_n, ν_n) is the final configuration of the only accepting run of $\mathcal{A}$ over w. The definitions of unambiguous copyless CRA and unambiguous BAC are straightforward restrictions of this definition.

We do not know whether for each unambiguous copyless CRA there is an equivalent deterministic copyless CRA. However, this is true when we assume bounded alternation.

▶ **Theorem 12.** *Let $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \delta, I_0, V_0, F, \mu)$ be an unambiguous BAC whose alternation is bounded by N. There exists a deterministic BAC $\mathcal{A}'$ that computes the same function as $\mathcal{A}$, that is, $[\![\mathcal{A}]\!](w) = [\![\mathcal{A}']\!](w)$ for every $w \in \Sigma^*$. Furthermore, the number of states of $\mathcal{A}'$ is of size $2^{\mathcal{O}(|Q|^3 \cdot |\mathcal{X}|^5 \cdot N^2)}$ and the number of registers in $\mathcal{A}'$ is of size $\mathcal{O}(|Q| \cdot |\mathcal{X}|^2 \cdot N)$.*

Proof (sketch). The construction goes by storing in $\mathcal{A}'$ the tree of $\mathcal{A}$-runs over the input. Of course, if we would keep all possible runs, then the set of states of $\mathcal{A}'$ would be infinite. A well-known property of unambiguous CRA is that, for each prefix, there is at most $|Q|$ possible runs and, furthermore, the last state of each run is different. Thus, $\mathcal{A}'$ can keep a tree structure in the states representing the runs where each branch represents a run and the number of branches is bounded by $|Q|$.

Unfortunately, we cannot do the same trick by naively keeping copies of the registers for each $\mathcal{A}$-run (recall that $\mathcal{A}'$ must be copyless). To overcome this problem, $\mathcal{A}'$ will do partial evaluation of the runs and postpone the use of common registers whenever it is possible by exploiting the copyless restriction and bounded alternation of $\mathcal{A}$. The full construction is in the appendix, available online. ◀

Closure under regular look-ahead. Our next extension of the CRA model is based on regular look-ahead, namely, transitions that can check regular properties over the input. Regular look-ahead has been extensively studied for finite automata [9, 10] and has been stated as a key property of a model for computing non-boolean functions [3, 2]. Let REG_Σ be the set of all regular languages over Σ. A CRA with regular look-ahead (CRA-RLA) is a tuple $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \Delta, q_0, \nu_0, \mu)$ where Q, Σ, $\mathcal{X}$, q_0, ν_0, and μ are defined as before and $\Delta : Q \times \mathrm{REG}_\Sigma \rightharpoonup Q \times \mathrm{Subs}(\mathcal{X})$ is a partial transition function. Given a string $w = a_1 \dots a_n \in \Sigma^*$, the run of $\mathcal{A}$ over w is a sequence of configurations: $(q_0, \nu_0) \xrightarrow{L_1} (q_1, \nu_1) \xrightarrow{L_2} \dots \xrightarrow{L_n} (q_n, \nu_n)$ such that for $1 \leq i \leq n$, $\Delta(q_{i-1}, L_i) = (q_i, \sigma_i)$, $a_i \dots a_n \in L_i$ and $\nu_i(x) = [\![\nu_{i-1} \circ \sigma_i(x)]\!]$ for each $x \in \mathcal{X}$. The output of $\mathcal{A}$ over w is defined as usual, i.e. $[\![\mathcal{A}]\!](w) = [\![\nu_n \circ \mu(q_n)]\!]$. To keep determinism, we also restrict Δ as follows: for a fixed state q let $\Delta(q, L_1) = (q_1, \sigma_1), \Delta(q, L_2) = (q_2, \sigma_2), \dots, \Delta(q, L_k) = (q_k, \sigma_k)$ be all transitions with q in the first coordinate and $L_1, \dots, L_k \in \mathrm{REG}_\Sigma$. Then the languages $L_1, \dots, L_k$ are pairwise disjoint (i.e. $L_i \cap L_j = \varnothing$). Note that $\mathcal{A}$ is always deterministic, i.e. after reading the remaining suffix the automaton is forced to take at most one available transition. The restrictions to copyless and bounded alternation CRA-RLA are defined as expected.

Like for unambiguous copyless CRA, we do not know if extending copyless CRA with regular look-ahead results in a more expressive model. Assuming bounded alternation we prove that the resulting class of functions is the same.

▶ **Theorem 13.** *For every BAC $\mathcal{A}$ with regular look-ahead there exists a BAC $\mathcal{A}'$ without regular look-ahead that computes the same function, that is, $[\![\mathcal{A}]\!](w) = [\![\mathcal{A}']\!](w)$ for every $w \in \Sigma^*$. Furthermore, the number of states and the number of registers of $\mathcal{A}'$ is double-exponential and polynomial, respectively, in the size of $\mathcal{A}$.*

Proof (sketch). The proof goes by constructing an unambiguous BAC $\mathcal{A}''$ that guesses the sequence of transitions $\Delta(p, L) = (q, \sigma)$ and checks later that L is satisfied over the suffix. If L is given by a finite deterministic automaton $\mathcal{R}$, this check can be done unambiguously by keeping the current state of each $\mathcal{R}$ and removing repeated runs. By Theorem 12, we know that $\mathcal{A}''$ can be converted into an equivalent deterministic BAC $\mathcal{A}'$. ◀

Closure under reverse. We finish this section proving that, in contrast to copyless CRA (see Section 4), BAC are closed under reverse. Recall that a subclass $\mathcal{C}$ of CRA is closed under reverse if for every $\mathcal{A} \in \mathcal{C}$ there exists $\mathcal{A}' \in \mathcal{C}$ such that $[\![\mathcal{A}]\!](w) = [\![\mathcal{A}']\!](w^r)$ for every $w \in \Sigma^*$.

▶ **Theorem 14.** *For every BAC $\mathcal{A}$ there exists a BAC $\mathcal{A}'$ that computes the reverse function of $\mathcal{A}$, that is, $[\![\mathcal{A}]\!](w) = [\![\mathcal{A}']\!](w^r)$ for every $w \in \Sigma^*$. Furthermore, the number of states is double-exponential and the number of registers is polynomial in the size of $\mathcal{A}$.*

We conclude this section by stressing the robustness of bounded alternation copyless CRA: they are closed under unambiguous non-determinism, regular look-ahead, and reverse operation. Notice that all the structural properties studied in Section 3 also apply to this class, in particular, the results regarding normal form and stable registers.

6 Conclusions and Future Work

In this paper, we studied structural properties, expressiveness and closure properties of copyless CRA. In particular, we showed that the class of functions recognized by copyless CRA are not closed under reverse. To recover the closure properties of CRA, we proposed the subclass of bounded alternation copyless CRA (BAC). We prove that BAC are closed under unambiguous non-determinism, regular look-ahead, and reverse. For general copyless CRA, we do not know whether this class is closed under unambiguous non-determinism or regular look-ahead. A positive answer would be surprising, since unambiguous automata are often trivially closed under the reverse operation (e.g. finite automata or weighted automata).

To study whether a subclass of CRA is closed under reverse, one could approach the problem in a more general way. A natural extension of BAC is to enhance the model with the ability of moving in both directions. Our results do not give a straightforward argument that BAC is closed under two-way extension, but we believe that this could be shown by exploiting the copyless restriction and bounded alternation similar as in the proof of Theorem 13.

The most important task for future work is to study the decidability properties of copyless CRA or BAC. The classical questions for computational models, like boundedness or equivalence of two automata, remain unanswered. We hope that the machinery developed in Section 3 is a first step towards answering these questions.

Acknowledgments. We thank the anonymous referees for their helpful comments. The first author was partially supported by Poland's National Science Center grant 2013/09/B/ST6/01575. The last author was supported by CONICYT + PAI / Concurso Nacional Apoyo al Retorno de Investigadores/as desde el extranjero – Convocatoria 2013 + 821320001 and by the Millenium Nucleus Center for Semantic Web Research under grant NC120004.

References

1 Shaull Almagor, Udi Boker, and Orna Kupferman. What's decidable about weighted automata? In *ATVA*, pages 482–491, 2011. doi:10.1007/978-3-642-24372-1_37.

2 Rajeev Alur, Loris D'Antoni, Jyotirmoy Deshmukh, Mukund Raghothaman, and Yifei Yuan. Regular functions and cost register automata. In *LICS*, pages 13–22. IEEE Computer Society, 2013.

3 Rajeev Alur and Loris D'Antoni. Streaming tree transducers. In *Automata, Languages, and Programming*, pages 42–53. Springer, 2012.

4 Rajeev Alur and Mukund Raghothaman. Decision problems for additive regular functions. In *Automata, Languages, and Programming*, pages 37–48. Springer, 2013.

5 J. Richard Buchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

6 Karel Culik II and Jarkko Kari. Image compression using weighted finite automata. In *Mathematical Foundations of Computer Science 1993*, pages 392–402. Springer, 1993.

7 Manfred Droste and Paul Gastin. Weighted automata and weighted logics. *Theor. Comput. Sci.*, 380(1-2):69–86, 2007.

8 Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of Weighted Automata*. Springer, 1st edition, 2009.

9 Joost Engelfriet. Top-down tree transducers with regular look-ahead. *Mathematical Systems Theory*, 10(1):289–303, 1976.

10 Joost Engelfriet and Sebastian Maneth. Macro tree transducers, attribute grammars, and mso definable tree translations. *Information and Computation*, 154(1):34–91, 1999.

11 Stephan Kreutzer and Cristian Riveros. Quantitative monadic second-order logic. In *LICS*, pages 113–122. IEEE Computer Society, 2013.

12 Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. In *ICALP*, pages 101–112, 1992. doi:10.1007/3-540-55719-9_67.

13 Filip Mazowiecki and Cristian Riveros. Maximal Partition Logic: Towards a Logical Characterization of Copyless Cost Register Automata. In Stephan Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic (CSL 2015)*, volume 41 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 144–159, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CSL.2015.144.

14 Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

15 Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1993.

16 M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2):114–125, April 1959. doi:10.1147/rd.32.0114.

17 M. P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4:245–270, 1961.