Journal of Computer and System Sciences 100 (2019) 1–29

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Copyless cost-register automata: Structure, expressiveness, and closure properties [☆]

Filip Mazowiecki [a,*], Cristian Riveros [b,c,**]

[a] University of Oxford, UK
[b] Pontificia Universidad Católica de Chile, Chile
[c] Millennium Institute for Foundational Research on Data (IMFD), Chile

ARTICLE INFO

Article history:
Received 20 June 2017
Received in revised form 27 July 2018
Accepted 30 July 2018
Available online 12 September 2018

Keywords:
Cost register automata
Weighted automata
Semirings

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 some 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.

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 [25], and its decidability problems [16,1], extensions [9], logic characterization [9,15], and applications [20,8] have been deeply investigated.

Recently, Alur et al. [4,6] 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 updated by combining registers contents using operations over a fixed semiring. In contrast to automata models with counters [14], CRA blindly updates its registers on each transition by using values computed on the previous state. In [4], 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.

New representations 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 were proposed in [4,2]. The idea of the so-called copyless restriction is to use each register at most once in every transition. Intuitively, this automaton model is what we call "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 [4] that "the existing proofs of the undecidability of equivalence rely on the unrestricted non-deterministic nature of weighted au-

[☆] A preliminary version of this paper appeared in [19] at the 33rd Symposium on Theoretical Aspects of Computer Science, STACS.
[*] Corresponding author.
[**] Corresponding author at: University of Oxford, UK.
E-mail addresses: filip.mazowiecki@labri.fr (F. Mazowiecki), cristian.riveros@uc.cl (C. Riveros).

https://doi.org/10.1016/j.jcss.2018.07.002
0022-0000/© 2018 Elsevier Inc. All rights reserved.

tomata" 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 analyzing 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 that cannot be reset and are constantly growing during a run. These registers are called stable registers, in the sense of having a stable assignment for transitions (see Section 3). We show that stable registers lead the behavior of copyless CRA and that they are crucial to analyze 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. In the original paper [4] it was shown that a restricted fragment of copyless CRA is a fragment of WA, and later in [18] it was shown that the whole class of copyless CRA is contained in WA. In this paper we show that the full class of copyless CRA is strictly less expressive than WA by analyzing some functions over specific semirings.

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 operations like union or intersection and, further, they can be enhanced by non-determinism and regular look-ahead without changing the class of recognized languages. These properties are probably one of the reasons behind its fruitful connection with MSO logic or finite monoids [7,22]. 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 behavior 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 [18] and characterized in terms of the so-called Maximal Partition logic. In contrast to copyless CRA, BAC are robust under several and natural extensions previously considered in [4,3]. Specifically, we show that BAC are closed under unambiguous non-determinism, regular look-ahead and under reverse. Furthermore, all the structural toolkit introduced for copyless CRA also extend for this class. These results emphasize that BAC is a promising computational model in the world of quantitative functions, showing that there exists a rich theory of functions below the class of WA.

New material in this article. Preliminary versions of some of the results in this work appeared in [19]. However, this article contains substantial new material. We include the full proofs of the main results and characterization presented in [19]. In particular, the proofs in Section 3 are of fundamental importance to understand the structure of copyless CRA (e.g. Propositions 1 to 3) and the proof of Theorem 2 shows how to use all the structural results to show its limits of expressibility. Furthermore, we believe that the proof of Theorem 3 is interesting on its own: the determinization of unambiguous non-deterministic BAC is a non-standard automata construction. The construction has some resemblance to the well-known Safra's construction [23], in the sense of keeping a tree structure of runs that is updated in a non-trivial way. In terms of new results, in Section 4 we show that copyless CRA is strictly less expressive than WA over the semiring of natural numbers. Interestingly, the proof over the natural semiring is simpler than over the max-plus semiring and works over the one-letter alphabet.

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. In Section 4 we work over the natural semiring and we show that copyless CRA are strictly contained in the class of weighted automata. Then we turn our attention to the max-plus semiring and we show in Section 5 that the class of copyless CRA is not closed under reverse, which implies that copyless CRA are strictly contained in weighted automata also over the max-plus semiring. In Section 6 we define BAC and show some closure properties of this class. We conclude in Section 7 with possible directions for future research.

2. Preliminaries

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

Basic concepts. We assume familiarity with basic definitions used in automata theory. We will work over finite alphabets, usually denoted Σ whose elements are called letters. A finite sequence of letters is a string. The set of all finite strings with letters in Σ is denoted Σ^*. For any two strings $w, u \in \Sigma^*$ the concatenation of them denoted wu is the string obtained by extending the sequence w with the sequence u. By the empty string ϵ we denote the empty sequence of letters that has the property $w\epsilon = \epsilon w = w$ for every $w \in \Sigma^*$.

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 from both sides, 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 and we usually denote the set of elements S by the name of the semiring $\mathbb{S}$. For 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 [10].

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 $[\![\mathcal{A}]\!] : \Sigma^* \to \mathbb{S}$ is the function computed by $\mathcal{A}$. Sometimes we call $[\![\mathcal{A}]\!]$ the semantics of $\mathcal{A}$.

For any string w we denote by w^r the reverse string and for each function f we define the function f^r by $f^r(w) = f(w^r)$ for all $w \in \Sigma^*$. A class F of functions is closed under reverse [4] if for every $f \in F$ the function f^r is also in F.

Variables, expressions, and substitutions. Fix a semiring $\mathbb{S} = (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$ and a finite set of variables $\mathcal{X}$ disjoint from S. We denote by $\mathrm{Expr}(\mathcal{X})$ the set of all expressions generated by the grammar

$$e ::= s \mid x \mid e \oplus e \mid e \odot e,$$

where $s \in \mathbb{S}$ and $x \in \mathcal{X}$. 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 $\mathrm{Var}(e) = \emptyset$. 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 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 $\sigma_1 \circ \sigma_2$ of substitutions σ_1 and σ_2 by $(\sigma_1 \circ \sigma_2)(x) = \hat{\sigma}_1(\sigma_2(x))$ for each $x \in \mathcal{X}$. It is readily verified that this operation is associative. For readability we will drop the hat in the notation and write substitution in composition form, e.g. we write $\sigma \circ e$ instead of $\hat{\sigma}(e)$. Moreover, we will often apply a chain of substitutions to an expression and then we will also drop the parentheses, writing for example $\sigma_1 \circ \sigma_2 \circ \sigma_3 \circ e$ instead of $\hat{\sigma}(e)$ for $\sigma = \sigma_1 \circ (\sigma_2 \circ \sigma_3)$.

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 $\mathrm{Var}(\nu \circ \sigma(x)) = \emptyset$ for all $x \in \mathcal{X}$. We say that two expressions e_1 and e_2 are equal (denoted by $e_1 \equiv e_2$) if they are equal up to evaluation equivalence, that is, $[\![\nu \circ e_1]\!] = [\![\nu \circ e_2]\!]$ for every valuation $\nu \in \mathrm{Val}(\mathcal{X})$. Sometimes when it is convenient we write $=$ instead of $\equiv$ (e.g. to allow writing $e = \max\{1, 2\} = 2$). We say that two substitutions σ_1 and σ_2 are equal if $\sigma_1(x) = \sigma_2(x)$ for every $x \in \mathcal{X}$.

Cost register automata. A cost register automaton (CRA) over a semiring $\mathbb{S}$ [4] is a tuple $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \delta, q_0, \nu_0, \mu)$ where Q is a finite set of states, Σ is a finite alphabet, $\mathcal{X}$ is a finite 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 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 $\mathcal{A}$. The automaton $\mathcal{A}$ defines the function $[\![\mathcal{A}]\!] : \Sigma^* \to \mathbb{S}$ as follows. Given a string $w = a_1 \dots a_n \in \Sigma^*$ ($n \geq 0$), 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 $1 \leq i \leq n$, $\delta(q_{i-1}, a_i) = (q_i, \sigma_i)$ 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 $[\![\nu_n \circ \mu(q_n)]\!]$. Note that we assume that every CRA defines a (total) function from words to values, opposed to the definition in [5] where CRA can define partial functions. This difference is not significant and all our results can be extended from total functions to partial functions.

Alternatively, $\mathcal{A}$ computes a function $|\mathcal{A}|$ from Σ^* to the set of ground expressions over $\mathbb{S}$. 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 $1 \leq i \leq n$, $\delta(q_{i-1}, a_i) = (q_i, \sigma_i)$, $\varsigma_0 = \nu_0$ and $\varsigma_i = \varsigma_{i-1} \circ \sigma_i$. 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 $[\![\mathcal{A}]\!](w) = [\![|\mathcal{A}|(w)]\!]$ for every $w \in \Sigma^*$. Hence, for any two automata $\mathcal{A}$ and $\mathcal{B}$, the equality $|\mathcal{A}| = |\mathcal{B}|$ implies $[\![\mathcal{A}]\!] = [\![B]\!]$.

We define the transitive closure of the transition function $\delta^* : Q \times \Sigma^* \to Q \times \mathrm{Subs}(\mathcal{X})$, by induction over the string length. Formally, $\delta^*(q, \epsilon) = (q, \mathrm{id})$ where $\mathrm{id}(x) = x$ 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 $\mathrm{Subs}(\mathcal{A}) = \{\sigma \mid \delta^*(p, w) = (q, \sigma)$ for $p, q \in Q$ and $w \in \Sigma^*\}$.

Copyless restriction and copyless CRA. We say that an expression $e \in \mathrm{Expr}(\mathcal{X})$ is copyless if each variable from $\mathcal{X}$ occurs at most once in e. For example, $x \odot (y \oplus z)$ is copyless but $(x \odot y) \oplus (x \odot z)$ is not copyless (because x is mentioned twice). Notice that the copyless restriction is a syntactic 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 $\mathrm{Var}(\sigma(x)) \cap \mathrm{Var}(\sigma(y)) = \emptyset$ for $x, y \in \mathcal{X}$ and $x \neq y$.

A CRA $\mathcal{A}$ is called copyless [4] if for every transition $\delta(q_1, a) = (q_2, \sigma)$ the substitution σ is copyless; and for every state $q \in Q$ the output expression $\mu(q)$ is copyless. In other words, every time $\mathcal{A}$ updates registers or outputs a value,

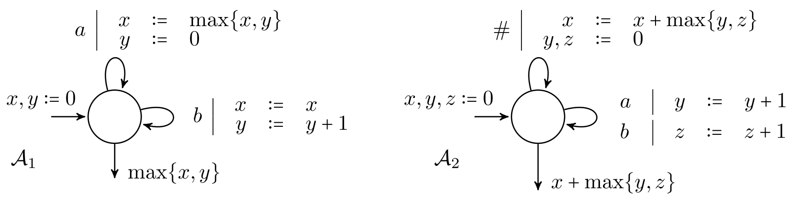


Fig. 1. Examples of copyless cost-register automata.

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 some examples of 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 Fig. 1. The CRA $\mathcal{A}_1$ stores in register y the length of the last suffix of b's and in register x 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 $w_0\#w_1\#\ldots\#w_n$ with $w_i \in \{a, b\}^*$, it computes the maximum of the number of a's and of 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 Fig. 1 computes f_2. In fact, the idea is similar to that of $\mathcal{A}_1$: the registers y and z count the number of a's and b's, respectively, in the longest suffix without # and the register x stores the partial output without considering the last suffix of a's and b's. Note that in Fig. 1 we omit some assignments if a register is not updated (i.e. it keeps its previous value). For example, for the a-transition we omit the assignments $x := x$ and $z := z$ for the sake of presentation. One should keep in mind these assignments because of the copyless restriction.

Weighted automata and linear CRA. We will use the class of functions defined by weighted automata only to compare its expressibility with copyless CRA (see Section 4 and 5). For the sake of completeness, we give the definition and semantics of weighted automata here (see [24,10] for some concrete examples of weighted automata). Fix a finite alphabet Σ and a commutative semiring $\mathbb{S}$. A weighted automaton (WA) over Σ and $\mathbb{S}$ is a tuple $\mathcal{A} = (Q, \Sigma, E, I, F)$ where Q is a finite set of states, $E : Q \times \Sigma \times Q \to \mathbb{S}$ is a weighted transition relation, and $I, F : Q \to \mathbb{S}$ are the initial and the final function, respectively [24,10]. Usually, if $E(p, a, q) = s$, we denote this transition graphically by $p \xrightarrow{a/s} q$. A run ρ of $\mathcal{A}$ over a word $w = w_1 \ldots w_n$ is a sequence of transitions: $\rho = q_0 \xrightarrow{w_1/s_1} q_1 \xrightarrow{w_2/s_2} \cdots \xrightarrow{w_n/s_n} q_n$. The weight of a run ρ of $\mathcal{A}$ over w is defined by $|\rho| = I(q_0) \odot (\bigodot_{i=1}^{n} s_i) \odot F(q_n)$. We define $\mathrm{Run}_{\mathcal{A}}(w)$ as the set of all runs of $\mathcal{A}$ over w. Finally, the weight of $\mathcal{A}$ over a word w is defined by $[\![\mathcal{A}]\!](w) = \bigoplus_{\rho \in \mathrm{Run}_{\mathcal{A}}(w)} |\rho|$ where the sum is equal to $\mathbb{0}$ if $\mathrm{Run}_{\mathcal{A}}(w)$ is empty.

In [4] it was shown that the class of functions defined by WA over some semiring $\mathbb{S}$ is equally expressive to the class of functions defined by linear CRA over $\mathbb{S}$. Formally, we call a CRA linear if all its substitutions and also its final output function are made by linear expressions over $\mathbb{S}$, namely, expressions of the form $\bigoplus_{i=1}^{n} s_i \odot x_i$ for some values $s_i \in \mathbb{S}$. For example, the CRA $\mathcal{A}_1$ in Example 1 is linear and can be defined by a WA, but the CRA $\mathcal{A}_2$ in Example 2 is not linear, given that the expression $x + \max\{y, z\}$ is not linear (although it is not hard to redefine $\mathcal{A}_2$ by an equivalent weighted automaton or linear CRA).

Trim assumption. For technical reasons, in this paper we assume that our finite automata and cost register automata are always trim, namely, for every state there exists a run that goes through that state (see e.g. [10]). It is worth noticing that verifying this property can be reduced to reachability tests in the transition graph [21] and this can be done in NLOGSPACE. Thus, we can assume without lost of generality that all our automata are trimmed.

Copyless expressions as sum of monomials. An expression that does not use $\oplus$ is called a monomial. By distributivity it is easy to show that every copyless expression is equivalent to a sum of monomials (despite that the new expression does not have to preserve the copyless property). This representation of copyless expressions as sum of monomials is an easy technical result and the reader can safely skip its proof in a first read.

Lemma 1. *For any copyless expression e, there exist an expression e' of the form $\bigoplus_{i=1}^{k} \left(c_i \odot \bigodot_{x \in X_i} x\right)$ where $k \geq 0$, $X_i \subseteq \mathrm{Var}(e)$, $c_i \in \mathbb{S}$, and all sets $X_1, \ldots, X_k$ are different, such that $e \equiv e'$.*

Proof. The lemma is shown by induction on the size of e. For the base case, when e is equal to a constant or a variable, the lemma trivially holds by taking $e' = e$. For the inductive case, suppose that $e = e_1 \otimes e_2$ where $\otimes$ is either $\oplus$ or $\odot$. By the inductive hypothesis we know that there exist expressions e'_1 and e'_2 equivalent to e_1 and e_2, respectively, such that for $j \in \{1,2\}$:

$$e'_j \equiv \bigoplus_{i=1}^{k_j} \left(c_i^j \odot \bigodot_{x \in X_i^j} x \right)$$

where $X_1^j, \ldots, X_{k_j}^j$ is a sequence of different sets over $\mathcal{X}$ and $c_1^j, \ldots, c_{k_j}^j$ is a sequence of values over $\mathbb{S}$ for $k_j \geq 0$. Given that e is a copyless expression we know that $\mathrm{Var}(e_1) \cap \mathrm{Var}(e_2) = \emptyset$. Then since $X_i^j \subseteq \mathrm{Var}(e_j)$ and $\mathrm{Var}(e_1) \cap \mathrm{Var}(e_2) = \emptyset$, we get:

$$X_{i_1}^1 \cap X_{i_2}^2 = \emptyset \qquad \text{for every } i_1 \leq k_1 \text{ and } i_2 \leq k_2. \tag{1}$$

Now we consider two cases when $\otimes$ is either $\oplus$ or $\odot$. For $\oplus$ we have $e = e_1 \oplus e_2 \equiv e'_1 \oplus e'_2 = e'$ and for $\odot$:

$$\begin{aligned} e &= e_1 \odot e_2 \ \equiv \ e'_1 \odot e'_2 \\ &= \bigoplus_{i=1}^{k_1} \big(c_i^1 \odot \bigodot_{x \in X_i^1} x\big) \odot \bigoplus_{i=1}^{k_2} \big(c_i^2 \odot \bigodot_{x \in X_i^2} x\big) \\ &\equiv \bigoplus_{i_1=1}^{k_1} \bigoplus_{i_2=1}^{k_2} \big(c_{i_1}^1 \odot c_{i_2}^1 \odot \bigodot_{x \in X_{i_1}^1 \cup X_{i_2}^2} x\big) \ = \ e' \end{aligned}$$

Recall that $X_{i_1}^1 \cap X_{i_2}^2 = \emptyset$ by (1). It remains to show that all sets of the form $X_{i_1}^1 \cup X_{i_2}^2$ are different. Indeed, all sets $X_{i_1}^1$ are pairwise different and the same holds for $X_{i_2}^2$. This means that all sets $X_{i_1}^1 \cup X_{i_2}^2$ are pairwise different as well. □

Removing zeros from copyless CRA. We say that an expression $e \in \mathrm{Expr}(\mathcal{X})$ is reduced if $e = \mathbb{0}$ or $\mathbb{0}$ does not occur in e. It is straightforward to show that for any expression e there exists an equivalent expression e^* that is reduced. Indeed, one can construct an equivalent reduced expression by replacing subexpressions of the form $e \oplus \mathbb{0}$ by e and $e \odot \mathbb{0}$ by $\mathbb{0}$, respectively. Then by reducing each subexpression recursively, the resulting expression is either $\mathbb{0}$ or $\mathbb{0}$ does not occur in it. We say that an expression is non-zero if it is different from $\mathbb{0}$. Further, note that if e is copyless, then its reduced expression e^* is copyless as well. The exact proof can be given by induction on the structure of e.

Similarly, we say that a substitution σ is reduced if $\sigma(x)$ is reduced for all registers x in the domain of σ. Additionally, we write that substitutions are non-zero if $\sigma(x) \neq \mathbb{0}$ for all x.

For a given $f : \Sigma^* \to \mathbb{S}$ we say that f is a non-zero function if $f(w) \neq \mathbb{0}$ for all $w \in \Sigma^*$. We say that an automaton $\mathcal{A}$ is non-zero if all substitutions and expressions (in transitions, initial valuation and final output function) used in it are non-zero. The following result shows that, without lost of generality, we can assume that all constants in a copyless CRA are different from $\mathbb{0}$ whenever the function defined by $\mathcal{A}$ is a non-zero function. Like for Lemma 1, the proof of Lemma 2 is an easy but technical construction, and the reader can skip its proof in a first read.

Lemma 2. *Let $\mathcal{A}$ be a copyless CRA such that $[\![\mathcal{A}]\!]$ is a non-zero function. Then there exists a copyless and non-zero CRA $\mathcal{A}'$ such that $[\![\mathcal{A}]\!] = [\![\mathcal{A}']\!]$.*

Proof. The idea of the proof is to keep the information about registers equal to $\mathbb{0}$ in the states. Let $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \delta, q_0, \nu_0, \mu)$ be a copyless CRA. We define a new copyless CRA $\mathcal{A}' = (Q', \Sigma, \mathcal{X}, \delta, q'_0, \nu'_0, \mu')$, where $Q' = Q \times 2^{\mathcal{X}}$ is the new set of states; $q'_0 = (q_0, S_0)$ where $S_0 = \{x \in \mathcal{X} \mid \nu_0(x) = \mathbb{0}\}$, and $\nu'_0(x) = \nu_0(x)$ for $x \in \mathcal{X} \setminus S_0$. Further, for registers $x \in S_0$ we define $\nu'_0(x) = \mathbb{1}$ (or any other constant in $\mathbb{S} \setminus \{\mathbb{0}\}$).

For the definition of δ' and μ' we introduce some notation. Let $S \subseteq \mathcal{X}$ and e be an expression over $\mathcal{X}$. We define $e[S]$ to be the reduced expression d^*, where d is the result of replacing all registers $x \in S \cap \mathrm{Var}(e)$ by $\mathbb{0}$ in the expression e. We are ready to define δ' and μ'. For every $(q, S) \in Q'$ and $a \in \Sigma$, we define $\delta'((q,S),a) = ((q', S'), \sigma')$, where: $\delta(q,a) = (q', \sigma)$; S' is the set of all registers x such that $\sigma(x)[S] = \mathbb{0}$; and σ' is defined for every $x \in \mathcal{X}$ as follows:

$$\sigma'(x) = \begin{cases} \sigma(x)[S] & \text{if } x \notin S' \\ \mathbb{1} & \text{if } x \in S' \end{cases}$$

Finally, we define the output function $\mu'((q,S)) = \mu(q)[S]$, for every $(q,S) \in Q'$.

It is straightforward to show that $[\![\mathcal{A}']\!] = [\![\mathcal{A}]\!]$ and that ν'_0, δ' and μ' do not mention $\mathbb{0}$. Note that there cannot exist a reachable state $(q, S) \in Q'$ such that $\mu'((q,S)) = \mathbb{0}$. Otherwise, $[\![\mathcal{A}]\!](w) = \mathbb{0}$ for some $w \in \Sigma^*$, which contradicts the fact that $[\![\mathcal{A}]\!]$ is a non-zero function. □

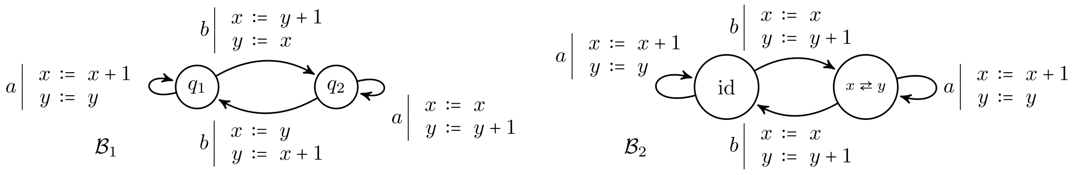

a | $x := x+1$, $y := y$
b | $x := y+1$, $y := x$
q_1
q_2
b | $x := y$, $y := x+1$
a | $x := x$, $y := y+1$
$\mathcal{B}_1$
a | $x := x+1$, $y := y$
b | $x := x$, $y := y+1$
id
$x \rightleftarrows y$
a | $x := x+1$, $y := y$
b | $x := x$, $y := y+1$
$\mathcal{B}_2$

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

Furthermore, we say that a semiring is a non-zero semiring if every ground expression e that does not use $\mathbb{0}$ cannot evaluate to $\mathbb{0}$. For example in the $\mathbb{N}_{-\infty}(\max, +)$ semiring, where $\mathbb{0} = -\infty$ if an expression uses only natural numbers then with $+$ and max operation one can never get $-\infty$. With this property every non-zero CRA over a non-zero semiring has the property that $\nu(x) \neq \mathbb{0}$ for each reachable configuration (q, ν) and every register $x \in \mathcal{X}$. We will need this property of the $\mathbb{N}_{-\infty}(\max, +)$ semiring later in Sections 3 and 5.

3. Structural properties of copyless CRA

In this section we analyze the structure of copyless CRA and develop some machinery that will be useful in Section 5. These results will help to understand the internal structure of copyless CRA.

3.1. Normal form

Let $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \delta, q_0, \nu_0, \mu)$ be a copyless CRA and let $\preceq$ be a predefined linear order over $\mathcal{X}$. We say that a substitution $\sigma : \mathcal{X} \to \mathrm{Expr}(\mathcal{X})$ is in normal form with respect to $\preceq$ if $x \preceq y$ for all $x \in X$ and all $y \in \mathrm{Var}(\sigma(x))$. In other words, all variables mentioned in $\sigma(x)$ are greater than or equal to x with respect to $\preceq$. Furthermore, we write that $\mathcal{A}$ is in normal form with respect to $\preceq$ if every $\sigma \in \mathrm{Subs}(\mathcal{A})$ is in normal form with respect to $\preceq$. For example, the copyless CRAs in Example 1 and 2 are in normal form with respect to the orders $x \preceq y$ and $x \preceq y \preceq z$, respectively. For the sake of simplification, we assume that every set of registers $\mathcal{X}$ is given with a linear order $\preceq_{\mathcal{X}}$ and instead of writing that $\mathcal{A}$ or σ are in normal form with respect to $\preceq_{\mathcal{X}}$ we write in short that $\mathcal{A}$ or σ are in normal form. Further for every nonempty subset $S \subseteq \mathcal{X}$, we denote by $\min(S)$ the least element of S with respect to $\preceq_{\mathcal{X}}$.

Example 3. Consider the set of registers $\mathcal{X} = \{x, y\}$ with the order $x \preceq y$. The copyless CRA $\mathcal{B}_1$ in Fig. 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 the automaton $\mathcal{B}_2$ in Fig. 2. We generalize this idea for all copyless CRA in the next proposition.

Proposition 1. *For every copyless CRA $\mathcal{A}$ there exists a copyless CRA in normal form $\mathcal{A}'$ with the same set of registers such that $|\mathcal{A}| = |\mathcal{A}'|$ 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. Let $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \delta, q_0, \nu_0, \mu)$ be a copyless CRA. We define a copyless CRA $\mathcal{A}'$ in normal form such that $\mathcal{A}'$ computes the same function as $\mathcal{A}$. The idea of $\mathcal{A}'$ is to store a state $q \in Q$ and a permutation ρ of the set $\mathcal{X}$ such that, if (q, ν) is the current configuration of $\mathcal{A}$ over an input w, then $((q, \rho), \nu')$ is the configuration of $\mathcal{A}'$ over w and $\nu(x) = \nu'(\rho(x))$. In other words, the content of ν is still defined by ν' but the value $\nu(x)$ of x is now in the register $\rho(x)$ for every $x \in \mathcal{X}$. The permutation of registers' content will allow us to keep the normal form in $\mathcal{A}'$. Formally, for an expression $e \in \mathrm{Expr}(\mathcal{X})$ and a permutation ρ over $\mathcal{X}$, we define $\rho(e)$ to be the expression obtained by replacing x in e with $\rho(x)$ for every $x \in \mathcal{X}$. Let $\mathcal{A}' = (Q', \Sigma, \mathcal{X}, \delta', q_0', \nu_0', \mu')$, where:

$$Q' = Q \times \{\rho \mid \rho \text{ is a permutation of the set } \mathcal{X}\}$$

is the set of states, $q_0' = (q_0, \mathrm{id})$ where id is the identity permutation, $\nu_0' = \nu_0$, and $\mu'((q, \rho)) = \rho(\mu(q))$. For the sake of presentation, let us show how the run of $\mathcal{A}'$ will correspond to the run of $\mathcal{A}$ before defining the transition function δ'. Let (q, ν) and (q', ν') be the configuration of $\mathcal{A}$ and $\mathcal{A}'$, respectively, after reading $w \in \Sigma^*$. We will show:

$$q' = (q, \rho) \text{ for some permutation } \rho \text{ and } \nu(x) = \nu'(\rho(x)) \text{ for every } x \in \mathcal{X}. \tag{2}$$

Note that, for the word ϵ, this correspondence holds since (q_0, ν_0) and $((q_0, \mathrm{id}), \nu'_0)$ are the initial configuration of $\mathcal{A}$ and $\mathcal{A}'$, respectively, and $\nu'_0(\mathrm{id}(x)) = \nu'_0(x) = \nu_0(x)$. We show that if (2) always holds, then $\mathcal{A}$ and $\mathcal{A}'$ computes the same function. Indeed, if the transition function δ' preserves (2), then:

$$\begin{aligned} \hat{\nu}'(\mu'((q,\rho))) &= \hat{\nu}'(\rho(\mu(q))) && \text{(by definition of } \mu'\text{)} \\ &= \hat{\nu}(\mu(q)) && \text{(by Property (2)).} \end{aligned}$$

This proves that the outputs of $\mathcal{A}$ and $\mathcal{A}'$ are the same (provided that δ' preserves (1)). It remains to define δ' such that $\mathcal{A}'$ is a copyless CRA in normal form and its definition satisfies (2).

We need some additional definitions. For a copyless substitution σ and a permutation ρ both over $\mathcal{X}$ we define the set $S_{\sigma,\rho} = \{x \in \mathcal{X} \mid \mathrm{Var}(\rho(\sigma(x))) \neq \emptyset\}$, that is, the set of all variables x where $\sigma(x)$ is not ground. Further, define the set $S'_{\sigma,\rho} = \{y \mid \exists x \in \mathcal{X}.\, y = \min(\mathrm{Var}(\rho(\sigma(x))))\}$. For each $x \in S_{\sigma,\rho}$, the set $\mathrm{Var}(\rho(\sigma(x)))$ is non-empty and thus has a least element. This motivates the function $\tau^0_{\sigma,\rho} : S_{\sigma,\rho} \to S'_{\sigma,\rho}$ that associates to each $x \in S_{\sigma,\rho}$ its least element, formally:

$$\tau^0_{\sigma,\rho}(x) = \min\left(\mathrm{Var}\left(\rho(\sigma(x))\right)\right) \tag{3}$$

One can easily check that $\tau^0_{\sigma,\rho}$ is a bijective function from $S_{\sigma,\rho}$ to $S'_{\sigma,\rho}$. It is surjective by the definition of $S_{\sigma,\rho}$ and $S'_{\sigma,\rho}$, and injective by the copyless restriction over σ. To see the last claim, recall that any copyless substitution σ satisfies $\mathrm{Var}(\sigma(x)) \cap \mathrm{Var}(\sigma(y)) = \emptyset$ and, in particular, $\mathrm{Var}(\rho(\sigma(x))) \cap \mathrm{Var}(\rho(\sigma(y))) = \emptyset$ for any permutation ρ. Finally, given that $\tau^0_{\sigma,\rho}$ is a bijective function from $S_{\sigma,\rho} \subseteq \mathcal{X}$ to $S'_{\sigma,\rho} \subseteq \mathcal{X}$, we can extend $\tau^0_{\sigma,\rho}$ to a bijection $\tau_{\sigma,\rho} : \mathcal{X} \to \mathcal{X}$ such that $\tau_{\sigma,\rho}(x) = \tau^0_{\sigma,\rho}(x)$ for every $x \in S_{\sigma,\rho}$. Of course, there might be many extensions of $\tau^0_{\sigma,\rho}$ to $\tau_{\sigma,\rho}$ but we can choose any extension (the decision is not important for the construction).

We have now all the ingredients to define the transition function δ'. For every $p \in Q$, $a \in \Sigma$, and permutation ρ over $\mathcal{X}$, if $\delta(p,a) = (q,\sigma)$ then we define $\delta'((p,\rho),a) = ((q,\tau_{\sigma,\rho}),\sigma')$ such that

$$\sigma'(x) = \rho(\sigma(\tau^{-1}_{\sigma,\rho}(x)))$$

for every $x \in \mathcal{X}$. The substitution $\rho \circ \sigma \circ \tau^{-1}_{\sigma,\rho}$ is a copyless substitution for every copyless σ because ρ and $\tau^{-1}_{\sigma,\rho}$ are just permuting the variables. Therefore, we can conclude that σ' is copyless as well.

Our next step is to show that σ' is in normal form, i.e., for every $x \in \mathcal{X}$ it holds that $x \preceq y$ for every $y \in \mathrm{Var}(\sigma'(x))$. We prove this by case analysis by considering whether $x \in S'_{\sigma,\rho}$ or not. First suppose that $x \in S'_{\sigma,\rho}$. Since x is in the codomain of $\tau^0_{\sigma,\rho}$ and $\tau_{\sigma,\rho}$ is an extension of $\tau^0_{\sigma,\rho}$, we have that $x = \min(\mathrm{Var}(\rho(\sigma(\tau^{-1}_{\sigma,\rho}(x)))))$ by (3). Then if we replace $\rho(\sigma(\tau^{-1}_{\sigma,\rho}(x)))$ by σ', we get that $x = \min(\mathrm{Var}(\sigma'(x)))$. In particular, we have that $x \preceq y$ for every $y \in \mathrm{Var}(\sigma'(x))$. Now suppose that $x \notin S'_{\sigma,\rho}$. This means that x is not in the codomain of $\tau^0_{\sigma,\rho}$ which implies that $\tau^{-1}_{\sigma,\rho}(x) \notin S_{\sigma,\rho}$. In other words, $\sigma'(x) = \rho(\sigma(\tau^{-1}_{\sigma,\rho}(x)))$ is a ground expression and $\mathrm{Var}(\sigma'(x)) = \emptyset$. Since for both cases it holds that $x \preceq y$ for every $x \in \mathcal{X}$ and $y \in \mathrm{Var}(\sigma'(x))$, then we conclude that σ' is in normal form.

For the last part of the proof, we show by induction that δ' satisfies the correspondence (2) between $\mathcal{A}$ and $\mathcal{A}'$. Let (p, ν_n) and $((p,\rho), \nu'_n)$ be the configuration of $\mathcal{A}$ and $\mathcal{A}'$, respectively, after reading $w \in \Sigma^*$. Assume by induction that $\nu_n(x) = \nu'_n(\rho(x))$ for every $x \in \mathcal{X}$ and let:

$$(p, \nu_n) \xrightarrow{a} (q, \nu_{n+1}) \quad \text{and} \quad ((p,\rho), \nu'_n) \xrightarrow{a} ((q, \tau_{\sigma,\rho}), \nu'_{n+1})$$

be the transitions for $\mathcal{A}$ and $\mathcal{A}'$, respectively, after reading a new letter $a \in \Sigma$. Fix the transitions of the last steps $\delta(p,a) = (q,a)$ and $\delta'((p,\rho),a) = ((q,\tau_{\sigma,\rho}),\sigma')$, where σ' is defined as above. We prove the correspondence (2) between ν_{n+1} and ν'_{n+1} as follows:

$$\begin{aligned} \nu'_{n+1}(\tau_{\sigma,\rho}(x)) &= \nu'_n(\sigma'(\tau_{\sigma,\rho}(x))) && \text{(by definition of } \nu'_{n+1}\text{)} \\ &= \nu'_n(\rho(\sigma(\tau^{-1}_{\sigma,\rho}(\tau_{\sigma,\rho}(x))))) && \text{(by definition of } \sigma'\text{)} \\ &= \nu'_n(\rho(\sigma(x))) && \text{(by composing } \tau_{\sigma,\rho} \text{ and } \tau^{-1}_{\sigma,\rho}\text{)} \\ &= \nu_n(\sigma(x)) && \text{(by the induction assumption)} \\ &= \nu_{n+1}(x) && \text{(by definition of } \nu_{n+1}\text{).} \end{aligned}$$

This proves that the transition function δ' keeps the correspondence (2) between $\mathcal{A}$ and $\mathcal{A}'$. Since it also holds for the initial configuration then by induction it holds for all reachable configurations, which proves that the outputs of $\mathcal{A}$ and $\mathcal{A}'$ are the same.

So far, we have shown that $\mathcal{A}$ and $\mathcal{A}'$ define the same function and every substitution on $\mathcal{A}'$-transitions are in normal form. To conclude the proof we need to show that if every substitution on $\mathcal{A}$-transitions is in normal form, then every $\sigma \in \mathrm{Subs}(\mathcal{A})$ is in normal form. We prove this by induction over the length of the word. Assume that it holds for all words w of length at most n and let $\delta^*(q,w) = (q',\sigma)$. Suppose we want to extend w with $a \in \Sigma$ and let $\delta(q',a) = (q'',\sigma')$.

By definition, we know that $\delta^*(q, w \cdot a) = (q'', \sigma \circ \sigma')$. Set $y \in \mathrm{Var}(\sigma \circ \sigma'(x))$. By definition there exists a register z such that $z \in \mathrm{Var}(\sigma'(x))$ and $y \in \mathrm{Var}(\sigma(z))$. Since δ is in normal form and $z \preceq y$ by the induction hypothesis, we conclude that $x \preceq z \preceq y$. In other words, $x \preceq y$ for every $y \in \mathrm{Var}(\sigma \circ \sigma'(x))$ and, thus, $\sigma \circ \sigma'$ is in normal form. □

3.2. Stable registers and reset substitutions

Let $\mathcal{A}$ 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 contributes to 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 no register contributes to the content of another one. 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 behavior and output of copyless CRA. Let $\sigma \in \mathrm{Subs}(\mathcal{X})$ 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 $\sigma \in \mathrm{Subs}(\mathcal{A})$. Moreover, we say that a register x is stable if it is stable in a CRA $\mathcal{A}$ which is clear from the context.

Stable registers play a crucial role in the behavior of copyless CRA. Indeed, we will 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 y to 0 each time the symbol a is read. On the other side, the register x 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 $\mathrm{Var}(\sigma(x)) = \emptyset$ 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 $\mathrm{Var}(\sigma(x)) = \emptyset$ for all non-stable registers x in $\mathcal{A}$.

We start with the following lemma that shows that the composition preserves stability between registers.

Lemma 3. *Let σ, σ' be two copyless substitution in normal form. For any register $x \in \mathcal{X}$ it holds that x is stable on σ and σ' if, and only if, x is $(\sigma \circ \sigma')$-stable.*

Proof. Suppose that x is stable on σ and σ'. This means that $x \in \mathrm{Var}(\sigma(x))$ and $x \in \mathrm{Var}(\sigma'(x))$ and, thus, $x \in \mathrm{Var}(\sigma \circ \sigma'(x))$ by composition, implying that x is $(\sigma \circ \sigma')$-stable. For the other direction, suppose that $x \in \mathrm{Var}(\sigma \circ \sigma'(x))$. Then we know that there exists y such that $x \in \mathrm{Var}(\sigma(y))$ and $y \in \mathrm{Var}(\sigma'(x))$. Since σ and σ' are in normal form, then $x \preceq y \preceq x$ which implies that $x = y$ and, thus, x is stable on σ and σ'. □

In our constructions we will often require that registers are stable in $\mathcal{A}$. By the previous lemma, it is enough to check that a register x is σ-stable for all transitions $\delta(p, a) = (q, \sigma)$ to know whether x is stable on all substitutions from $\mathrm{Subs}(\mathcal{A})$.

For $Q' \subseteq Q$ we say that Q' is a bottom strongly connected component (BSCC) of $\mathcal{A}$ if (1) for every pair $q_1, q_2 \in Q'$ there exists $w \in \Sigma^*$ such that $\delta^*(q_1, w) = (q_2, \sigma)$ for some substitution σ and (2) for every $q_1 \in Q'$ and $w \in \Sigma^*$ if $\delta^*(q_1, w) = (q_2, \sigma)$ then $q_2 \in Q'$. Intuitively a BSCC Q' of $\mathcal{A}$ is a set of mutually reachable states such that there is no word that leaves Q'. We say that $\mathcal{A}$ is strongly connected if the whole set Q is a BSCC of $\mathcal{A}$.

Proposition 2. *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.*

For instance in Example 1 it suffices to take the word $w = ba$ or any word that contains a, given that the substitution defined by the a-transition is an $\mathcal{A}$-reset substitution. For simplicity if $q = q'$, we write w^q and σ^q instead of $w^{q,q}$ and $\sigma^{q,q}$. Note that the additional requirement that $w^{q,q'}$ contains all letters in Σ is rather technical and its usefulness will be clear later in Section 5.

Proof of Proposition 2. Let $x_1, \ldots, x_n$ be all registers in $\mathcal{X}$ in the increasing order with respect to $\preceq$. We prove the following statement. For every $i \leq n$ there exist a word $w_i^{q,q'}$ and a substitution $\sigma_i^{q,q'}$ such that $\delta^*(q, w_i^{q,q'}) = (q', \sigma_i^{q,q'})$ and $\mathrm{Var}(\sigma_i^{q,q'}(x)) = \emptyset$ for all registers x with $x_i \preceq x$ which are non-stable in $\mathcal{A}$. Then the proposition follows from this statement with $w^{q,q'} = w_1^{q,q'}$ and $\sigma = \sigma_1^{q,q'}$.

We start with the base case $i = n$. Now $x = x_n$ and we consider whether x is stable in $\mathcal{A}$ or not. If x_n is stable, then take a word u and a substitution τ such that $\delta^*(q, u) = (q', \tau)$ and u contains each letter in Σ. Given that $\mathcal{A}$ is strongly connected we know that u and τ always exists. Then by defining $w_n^{q,q'} = u$ and $\sigma_n^{q,q'} = \tau$, the statement holds because x_n is stable in $\mathcal{A}$. Now, suppose that x_n is non-stable which means that there exist a pair $p, p' \in Q$ and a word u such that $\delta^*(p, u) = (p', \tau)$ and x_n is non-stable on τ. Given that $\mathcal{A}$ is in normal form and x_n is the maximum register with respect to $\preceq$, this implies that $\mathrm{Var}(\tau(x_n)) = \emptyset$. Pick two words $v_1, v_2 \in \Sigma^*$ such that $\delta^*(q, v_1) = (p, \sigma_1)$ and $\delta^*(p', v_2) = (q', \sigma_2)$ for some substitutions

σ_1 and σ_2, and v_1 contains each letter in Σ. Again, we know that v_1 and v_2 always exists since $\mathcal{A}$ is strongly connected. Then define $w_n^{q,q'} = v_1 \cdot u \cdot v_2$ and $\sigma_n^{q,q'} = \sigma_1 \circ \tau \circ \sigma_2$. By construction, we know that $\delta^*(q, w_n^{q,q'}) = (q', \sigma_n^{q,q'})$ and $w_n^{q,q'}$ contains all letters in Σ. To prove that $\mathrm{Var}(\sigma_n^{q,q'}(x_n)) = \emptyset$, notice that x_n is non-stable on τ. By Lemma 3 this implies that x_n is non-stable on $\sigma_1 \circ \tau \circ \sigma_2$. Given that $\mathcal{A}$ is in normal form and x_n is the maximal register, we get that $\mathrm{Var}(\sigma_n^{q,q'}(x_n)) = \emptyset$.

For the inductive step, assume that $w_{i+1}^{q,q'}$ and $\sigma_{i+1}^{q,q'}$ exist. We show how to construct $w_i^{q,q'}$ and $\sigma_i^{q,q'}$ that satisfy the statement for registers greater than or equal to x_i. Again, we consider two cases depending on whether x_i is stable or not. If x_i is stable, then by defining $w_i^{q,q'} = w_{i+1}^{q,q'}$ and $\sigma_i^{q,q'} = \sigma_{i+1}^{q,q'}$ the inductive step trivially holds. Suppose that x_i is non-stable and, therefore, there exist a pair $p, p' \in Q$ and a word u such that $\delta^*(p, u) = (p', \tau)$ and x_i is non-stable on τ. Let $v_1, v_2 \in \Sigma^*$ such that $\delta^*(q, v_1) = (p, \sigma_1)$ and $\delta^*(p', v_2) = (q', \sigma_2)$ for some substitutions σ_1 and σ_2. Recall that v_1 and v_2 exist because $\mathcal{A}$ is strongly connected. Now, define:

$$\begin{aligned} w_i^{q,q'} &= w_{i+1}^{q} \cdot v_1 \cdot u \cdot v_2 \\ \sigma_i^{q,q'} &= \sigma_{i+1}^{q} \circ \sigma_1 \circ \tau \circ \sigma_2 \end{aligned}$$

It is clear by construction that $\delta^*(q, w_i^{q,q'}) = (q', \sigma_i^{q,q'})$ and $w_i^{q,q'}$ contains all letter in Σ (because already w_{i+1}^{q} contains all letters in Σ). To conclude the proof, we show that $\mathrm{Var}(\sigma_i^{q,q'}(x)) = \emptyset$ for every non-stable register $x \succeq x_i$. Let x be any non-stable register $x \succeq x_i$ (possibly x_i) and let $\tau^* = \sigma_1 \circ \tau \circ \sigma_2$. First, note that all $y \in \mathrm{Var}(\tau^*(x))$ are non-stable. Otherwise, if $y \in \mathrm{Var}(\tau^*(x))$ is stable, then $y \in \mathrm{Var}(\tau^*(y))$, which is a contradiction because $\mathrm{Var}(\tau^*(x)) \cap \mathrm{Var}(\tau^*(y)) = \emptyset$ by the definition of being copyless. Therefore, we have that every register in $\mathrm{Var}(\tau^*(x))$ is non-stable. Note also that $x_i \notin \mathrm{Var}(\tau^*(x))$. This is obvious when $x \neq x_i$ because $\mathcal{A}$ is in normal form. Otherwise, $x = x_i$ and we know that $x_i \notin \mathrm{Var}(\tau^*(x_i))$ because x_i is non-stable. Then we have that all registers in $\mathrm{Var}(\tau^*(x))$ are non-stable and strictly greater than x_i. By the induction assumption $\mathrm{Var}(\sigma_{i+1}^{q}(y)) = \emptyset$ for all $y \in \mathrm{Var}(\tau^*(x))$. By composing σ_{i+1}^{q} and τ^*, we conclude that $\mathrm{Var}(\sigma_i^{q,q'}(x)) = \emptyset$. □

3.3. *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 stable 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, where σ^n is σ composed n-times with itself. We will need the next result to show that by iterating cycles one can always "reset" the content of non σ-stable registers.

Lemma 4. *Let $\mathcal{A}$ be in normal form and $\sigma \in \mathrm{Subs}(\mathcal{A})$ a copyless substitution. There exists $N \leq |\mathcal{X}|$ such that σ^N is a reset substitution.*

Proof. Suppose x is non-stable over σ, i.e. $x \notin \mathrm{Var}(\sigma(x))$, and assume that $\mathrm{Var}(\sigma(x)) \neq \emptyset$. Then consider the register $y_1 = \min(\mathrm{Var}(\sigma(x)))$. Since $\mathcal{A}$ is copyless and in normal form then $x \prec y_1$. But since $y_1 \in \mathrm{Var}(\sigma(x))$ and $\mathcal{A}$ is copyless then $y_1 \notin \mathrm{Var}(\sigma(y_1))$ (because y_1 cannot appear twice in σ). Recall that $\mathrm{Var}(\sigma(y_1)) \subseteq \mathrm{Var}(\sigma^2(x))$ given that $y_1 \in \mathrm{Var}(\sigma(x))$. Combining both facts we get that $y_1 \notin \mathrm{Var}(\sigma^2(x))$ and because $\mathcal{A}$ is in normal form $y_1 \prec \min(\mathrm{Var}(\sigma^2(x)))$.

So far, we have proved that $x \prec y_1 = \min(\mathrm{Var}(\sigma(x))) \prec \min(\mathrm{Var}(\sigma^2(x)))$ and $x \notin \mathrm{Var}(\sigma^2(x))$. Also, we know that σ^2 is a copyless substitution in normal form. Then, we can repeat the same argument above for x and σ^2, forming an increasing sequence of registers:

$$x \prec y_1 = \min(\mathrm{Var}(\sigma(x))) \prec y_2 = \min(\mathrm{Var}(\sigma^2(x))) \prec y_3 = \min(\mathrm{Var}(\sigma^3(x))) \prec \ldots$$

The number of registers is finite so this sequence cannot be infinite. Thus for every x there exists $N_x \leq |\mathcal{X}|$ such that $\mathrm{Var}(\sigma^{N_x}(x)) = \emptyset$. Then σ^N is a reset substitution, where $N = \max\{N_x \mid x \text{ is non-stable}\}$. □

By Lemma 4 we know that a non-stable register x over σ becomes a constant when σ is iterated at least $|\mathcal{X}|$-times. 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 additional assumptions that automata do not use $\mathbb{0}$ in their expressions and that the semiring $\mathbb{S}$ is non-zero (see Section 1 for details).

Proposition 3. *Let $\mathbb{S}$ be a non-zero semiring and let $\mathcal{A}$ be a non-zero automaton in normal form. Let $\sigma \in \mathrm{Subs}(\mathcal{A})$ be 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 $j \geq 0$ we have:*

$$\sigma^{j+1}(x) \;\equiv\; (c^j \odot \sigma(x)) \oplus \big(d \odot \bigoplus_{k=0}^{j-1} c^k\big)$$

Before proving Proposition 3, we must show a straightforward fact about copyless expressions.

Lemma 5. *Suppose the semiring $\mathbb{S}$ is a non-zero semiring and $\mathcal{A}$ is a non-zero automaton. Let $\sigma, \tau \in \mathsf{Subs}(\mathcal{A})$ where σ is a reset substitution and let x be a σ-stable and τ-stable register such that $\mathsf{Var}(\sigma(y)) = \emptyset$ for all $y \in \mathsf{Var}(\tau(x)) \setminus \{x\}$. Then the expression $\sigma \circ \tau(x)$ is equivalent to an expression of the form $(c \odot \sigma(x)) \oplus d$, where $c, d \in \mathbb{S}$ and $c \neq \mathbb{0}$.*

We will use Lemma 5 in two cases: for $\tau = \sigma$, or when σ is $\mathcal{A}$-reset substitution. Notice that in both cases the assumption $\mathsf{Var}(\sigma(y)) = \emptyset$ for all $y \in \mathsf{Var}(\tau(x)) \setminus \{x\}$ holds.

Proof of Lemma 5. We prove this by induction on the length of expression $\tau(x)$. For the base step take $\tau(x) = x$. Then $\sigma \circ \tau(x) = \sigma(x)$ is equivalent to $(\mathbb{1} \odot \sigma(x)) \oplus \mathbb{0}$. Suppose we can write such an expression $(c \odot \sigma(x)) \oplus d$ for τ of length at most n. By the inductive assumption when τ is of length $n+1$ then $\sigma \circ \tau(x)$ is equivalent to $((c \odot \sigma(x)) \oplus d) \oplus \sigma(e)$ or $((c \odot \sigma(x)) \oplus d) \odot \sigma(e)$ for some expression e. Since $\tau(x)$ is copyless then $\mathsf{Var}(e) \subseteq \mathcal{X} - \{x\}$. Furthermore, $\mathsf{Var}(\sigma(y)) = \emptyset$ for $y \in \mathsf{Var}(e)$ and we get $\sigma(e) \equiv d'$ for some constant $d' \neq \mathbb{0}$ (by the non-zero assumptions on $\mathcal{A}$ and $\mathbb{S}$). Thus, $\sigma \circ \tau(x)$ is equivalent to $(c \odot \sigma(x)) \oplus (d \oplus d')$ or $(((d' \odot c) \odot \sigma(x)) \oplus (d' \odot d))$, respectively. □

Proof of Proposition 3. Since $\sigma \in \mathsf{Subs}(\mathcal{A})$ is a copyless and reset substitution, then for every $y \in \mathsf{Var}(\sigma(x))$ either $y = x$ or $\mathsf{Var}(\sigma(y)) = \emptyset$. By Lemma 5 $e = \sigma \circ \sigma(x)$ (setting $\tau = \sigma$) is equivalent to $e^* = (c \odot \sigma(x)) \oplus d$ for some $c, d \in S$ and $c \neq \mathbb{0}$.

We prove the proposition by induction using the constants c, d from e^*. For $j = 0$ the claim is trivially true and for $j = 1$ the expression e^* is of the desired form. For the inductive step, we have $\sigma^{j+2}(x) = \sigma^j \circ \sigma^2(x) = \sigma^j \circ e^*$. Then:

$$\begin{aligned}
\sigma^{j+2}(x) &\equiv \sigma^j \circ ((c \odot \sigma(x)) \oplus d) && \text{(because } e^* = (c \odot \sigma(x)) \oplus d\text{)} \\
&\equiv (c \odot \sigma^{j+1}(x)) \oplus d \\
&\equiv \left(c \odot \left((c^j \odot \sigma(x)) \oplus \left(d \odot \textstyle\bigoplus_{k=0}^{j-1} c^k\right)\right)\right) \oplus d && \text{(by induction)} \\
&\equiv \left((c^{j+1} \odot \sigma(x)) \oplus \left(d \odot \textstyle\bigoplus_{k=1}^{j} c^k\right)\right) \oplus d \\
&\equiv (c^{j+1} \odot \sigma(x)) \oplus \left(d \odot \textstyle\bigoplus_{k=0}^{j} c^k\right) && \square
\end{aligned}$$

Proposition 3 shows how σ-stable registers grow with respect of the number of times that a cycle is iterated. In particular, when $\mathbb{S} = \mathbb{N}_{-\infty}(\max, +)$ then σ-stable registers grows linearly because $\odot$ operation is $+$ in this semiring.

For the next result we fix the $\mathbb{N}_{-\infty}(\max, +)$-semiring. Intuitively, when some specific substitutions are applied to registers then to measure the value of stable registers we will refine Proposition 3 while non-stable registers will have constant value. This result will be crucial in the proof of Theorem 2. Recall that, by Proposition 2, for any $q \in Q$ there exists a word w^q such that $\delta^*(q, w^q) = (q, \sigma^q)$ and σ^q is an $\mathcal{A}$-reset substitution.

Lemma 6. *Let $\mathcal{A}$ be a copyless, strongly connected and non-zero CRA in normal form over the $\mathbb{N}_{-\infty}(\max, +)$ semiring. Furthermore, let $q \in Q$ and $v \neq \epsilon$ be a cycle in q, namely, $\delta^*(q, v) = (q, \tau)$ for some reset substitution τ. For every $j \in \mathbb{N}$ let $\sigma_j = \sigma^q \circ \tau^{j+1} \circ \sigma^q$. Then for every $x \in \mathcal{X}$ there exists $c, d \in \mathbb{N}$ such that for every $\lambda \in \mathsf{Subs}(\mathcal{A})$ and j big enough the following holds:*

$$\sigma_j \circ \lambda(x) \equiv \begin{cases} \mathcal{O}(1) & \textit{if } x \textit{ is non-stable} \\ \max\{\, j \cdot c + \sigma^q(x) + \mathcal{O}(1),\ j \cdot d + \mathcal{O}(1) \,\} & \textit{otherwise.} \end{cases}$$

The additional substitution λ will be necessary in Section 5. Intuitively, we will compose many substitutions and λ will represent a substitution from previous compositions.

Proof. Fix a substitution λ. For a non-stable register x in $\mathcal{A}$ the result is straightforward. Indeed, $\mathcal{A}$ is in normal form, and thus $\mathsf{Var}(\lambda(x))$ contains just non-stable registers. This implies that:

$$\begin{aligned}
\sigma_j \circ \lambda(x) &= \sigma^q \circ \tau^j \circ \sigma^q \circ \lambda(x) && \text{(by definition)} \\
&= \sigma^q \circ \lambda(x) \\
&= \mathcal{O}(1) && (\sigma^q \text{ and } \lambda \text{ do not depend on } j)
\end{aligned}$$

The second equality holds because $\mathsf{Var}(\lambda(x))$ contains only non-stable register and σ^q is a reset substitution, hence $\mathsf{Var}(\sigma^q \circ \lambda(x)) = \emptyset$. Suppose now that x is a stable register in $\mathcal{A}$. We need to show that:

$$\sigma_j \circ \lambda(x) \equiv \max\{\, j \cdot c + \sigma^q(x) + \mathcal{O}(1),\ j \cdot d + \mathcal{O}(1) \,\} \tag{4}$$

for j big enough and some constants $c, d \in \mathbb{N}$ such that c, d do not depend on λ. Note that by Proposition 3 it holds that for every $j \geq 0$:

$$\begin{aligned}
\tau^{j+1}(x) &\equiv (c_x^j \odot \tau(x)) \oplus \big(d_x \odot \bigoplus_{k=0}^{j-1} c_x^k\big) && \text{(by Proposition 3)}\\
&= \max\left\{ j\cdot c_x + \tau(x), d_x + \max_{k=0}^{j-1}\{k\cdot c_x\} \right\} && \text{(by definition of } \mathbb{N}_{-\infty}(\max,+))\\
&= \max\{\, j\cdot c_x + \tau(x),\ d_x + (j-1)\cdot c_x \,\} && \text{(by definition of max)}\\
&= \max\{\, j\cdot c_x + \tau(x) + \mathcal{O}(1),\ j\cdot c_x + \mathcal{O}(1) \,\} && \text{(by using the } \mathcal{O}\text{-notation)} \qquad (5)
\end{aligned}$$

Recall that $\mathcal{A}$ is a non-zero CRA and in this semiring $\mathbb{0} = -\infty$, thus $c_x \geq 0$. If $d_x = -\infty$ then the last equality is not immediate. It holds because:

$$\max\{\, j\cdot c_x + \tau(x),\ d_x + (j-1)\cdot c_x\} = j\cdot c_x + \tau(x) = \max\{j\cdot c_x + \tau(x), j\cdot c_x\,\} =$$
$$\max\{\, j\cdot c_x + \tau(x) + \mathcal{O}(1),\ j\cdot c_x + \mathcal{O}(1)\,\}$$

Coming back to the proof of (4), note that the expression $\sigma_j \circ \lambda(x)$ is equivalent to the expression obtained by replacing $\sigma^q \circ \tau^{j+1}(y)$ for each variable y in $\sigma^q \circ \lambda(x)$. We start the proof by analyzing these expressions. Let $y \in \mathrm{Var}(\sigma^q \circ \lambda(x))$. If y is not a τ-stable register, then $\mathrm{Var}(\tau(y)) = \emptyset$ and thus $\sigma^q \circ \tau^{j+1}(y) = \mathcal{O}(1)$. Otherwise, by (5) we know that $\tau^{j+1}(y) = \max\big\{j\cdot c_y + \tau(x) + \mathcal{O}(1),\ j\cdot c_y + \mathcal{O}(1)\big\}$ for some $c_y \in \mathbb{N}$ and, thus, by applying σ^q on $\tau^{j+1}(y)$ we get:

$$\sigma^q \circ \tau^{j+1}(y) \;=\; \max\big\{\, j\cdot c_y + \sigma^q \circ \tau(x) + \mathcal{O}(1),\ j\cdot c_y + \mathcal{O}(1)\,\big\} \qquad (6)$$

If y is a τ-stable register, but non-stable on $\mathcal{A}$ (i.e. non-stable in general) then $\sigma^q \circ \tau(y)$ is equal to a constant and does not depend on j. Thus we can estimate $\sigma^q \circ \tau(y)$ by $\mathcal{O}(1)$ and then (6) becomes:

$$\begin{aligned}
\sigma^q \circ \tau^{j+1}(y) &= \max\big\{\, j\cdot c_y + \mathcal{O}(1),\ j\cdot c_y + \mathcal{O}(1)\,\big\}\\
&= j\cdot c_y + \mathcal{O}(1)
\end{aligned}$$

Otherwise, y is a stable register and, moreover, σ^q is a reset substitution. Then by Lemma 5 we know that $\sigma^q \circ \tau(y)$ can be represented as:

$$\begin{aligned}
\sigma^q \circ \tau(y) &= \max\{\, c' + \sigma^q(y),\ d'\,\}\\
&= \max\{\, \sigma^q(y) + \mathcal{O}(1),\ \mathcal{O}(1)\,\} \qquad (7)
\end{aligned}$$

for some constants $c', d' \in \mathbb{N}$ not depending on j and where $c' \neq -\infty$. Thus, by combining (6) and (7) we get:

$$\begin{aligned}
\sigma^q \circ \tau^{j+1}(y) &= \max\big\{\, j\cdot c_y + \sigma^q \circ \tau(x) + \mathcal{O}(1),\ j\cdot c_y + \mathcal{O}(1)\,\big\} && \text{(by (6))}\\
&= \max\big\{\, j\cdot c_y + \max\{\,\sigma^q(y) + \mathcal{O}(1),\ \mathcal{O}(1)\,\} + \mathcal{O}(1),\ j\cdot c_y + \mathcal{O}(1)\,\big\} && \text{(by (7))}\\
&= \max\big\{\, j\cdot c_y + \sigma^q(y) + \mathcal{O}(1),\ j\cdot c_y + \mathcal{O}(1),\ j\cdot c_y + \mathcal{O}(1)\,\big\} && \text{(by distribution)}\\
&= \max\big\{\, j\cdot c_y + \sigma^q(y) + \mathcal{O}(1),\ j\cdot c_y + \mathcal{O}(1)\,\big\} && (8)
\end{aligned}$$

Observe that a variable $y \in \mathrm{Var}(\sigma^q \circ \lambda(x))$ is stable if, and only if, $y = x$. We summarize the three possible cases in the next equation:

$$\sigma^q \circ \tau^{j+1}(y) \;=\; \begin{cases} \mathcal{O}(1) & \text{if } y \text{ is not } \tau\text{-stable}\\ j\cdot c_y + \mathcal{O}(1) & \text{if } y \text{ is non-stable}\\ \max\big\{j\cdot c_y + \sigma^q(y) + \mathcal{O}(1),\ j\cdot c_y + \mathcal{O}(1)\big\} & \text{if } x = y \end{cases} \qquad (9)$$

Now we can prove (4). Recall that the expression $\sigma_j \circ \lambda(x)$ is the expression $\sigma^q \circ \lambda(x)$ where all registers $y \in \mathrm{Var}(\sigma^q \circ \lambda(x))$ are substituted with $\sigma^q \circ \tau^{j+1}(y)$. First notice that by Lemma 5 the expression $\sigma^q \circ \lambda(x)$ is equivalent to $\max\{c'' + \sigma^q(x), d''\} = \max\{\sigma^q(x) + \mathcal{O}(1), \mathcal{O}(1)\}$. We can do these estimations because λ does not depend on j. By combining this equation with the definition of σ_j we have:

$$\begin{aligned}
\sigma_j \circ \lambda(x) &= \sigma^q \circ \tau^{j+1} \circ \sigma^q \circ \lambda(x)\\
&= \sigma^q \circ \tau^{j+1} \circ \max\{\, \sigma^q(x) + \mathcal{O}(1),\ \mathcal{O}(1)\,\}\\
&= \max\{\, \sigma_j(x) + \mathcal{O}(1),\ \mathcal{O}(1)\,\}
\end{aligned}$$

Thus to finish the proof it suffices to show that:

$$\sigma_j(x) \;=\; \max\{\, j\cdot c + \sigma^q(x) + \mathcal{O}(1),\; j\cdot d + \mathcal{O}(1)\,\} \tag{10}$$

for some constants $c, d \in \mathbb{N}$. Indeed, if (10) holds, then $\sigma_j(x) + \mathcal{O}(1) = \max\{j\cdot c + \sigma^q(x) + \mathcal{O}(1), j\cdot d + \mathcal{O}(1)\}$. Similarly the value $\mathcal{O}(1)$ can be estimated by the expression $j\cdot d + \mathcal{O}(1)$ so the last max operation is not needed. Notice that this does not change the constants c and d, which proves that they do not depend on λ.

To conclude the proof, we show that (10) holds for a stable register x. Recall that $\sigma_j = \sigma^q \circ \tau^{j+1} \circ \sigma^q$. We will show that (10) holds even if we change σ_j to $\sigma'_j = \sigma^q \circ \tau^{j+1} \circ \sigma'$, where σ' is any copyless substitution in normal form and x is σ'-stable. The proof is by induction on the size of $\sigma'(x)$. For the base step we must consider $\sigma'(x) = x$ (because x is stable). Then

$$\begin{aligned}
\sigma'_j(x) &= \sigma^q \circ \tau^{j+1} \circ \sigma'(x)\\
&= \sigma^q \circ \tau^{j+1}(x)\\
&= \max\{\, j\cdot c_x + \sigma^q(x) + \mathcal{O}(1),\; j\cdot c_x + \mathcal{O}(1)\,\} \qquad \text{(by (8))}.
\end{aligned}$$

For the inductive step, assume that (10) holds for σ'_j with substitutions σ' such that the length of $\sigma'(x)$ is at most n. Now let σ'' be a substitution such that the length of $\sigma''(x)$ is $n+1$. Then $\sigma''(x) = \sigma'(x) \circledast f$ where $\circledast$ is either $+$ or max, x is σ'-stable and f is an expression where all registers are non-stable (recall that σ'' is copyless). By unraveling $\sigma^q \circ \tau^{j+1} \circ \sigma''(x)$, we get that:

$$\begin{aligned}
\sigma^q \circ \tau^{j+1} \circ \sigma''(x) &= \sigma^q \circ \tau^{j+1} \circ (\sigma'(x) \circledast f)\\
&= (\sigma^q \circ \tau^{j+1} \circ \sigma'(x)) \circledast (\sigma^q \circ \tau^{j+1} \circ f)\\
&= \max\{\, j\cdot c + \sigma^q(x) + \mathcal{O}(1),\; j\cdot d + \mathcal{O}(1)\,\} \circledast (\sigma^q \circ \tau^{j+1} \circ f) \qquad \text{(by induction)}
\end{aligned}$$

In the final expression it is easy to check that $\sigma^q \circ \tau^{j+1} \circ f$ is equal to a constant or to $j\cdot c_f + \mathcal{O}(1)$ for some $c_f \neq -\infty$. Indeed, by (9) we know that $\sigma^q \circ \tau^{j+1}(y) = j\cdot c_y + \mathcal{O}(1)$ for non-stable registers. Since f is an expression over non-stable registers, one can show by induction over the size of f that $\sigma^q \circ \tau^{j+1} \circ f$ is of the form $j\cdot c_f + \mathcal{O}(1)$ for some constant $c_f \neq -\infty$ (but possibly $c_f = 0$) and j big enough.

We assume that $\sigma^q \circ \tau^{j+1} \circ f = j\cdot c_f + \mathcal{O}(1)$. Then we must consider two cases whether $\circledast$ is $+$ or max operation. For the former we have:

$$\begin{aligned}
\sigma^q \circ \tau^{j+1} \circ \sigma''(x) &= \max\{\, j\cdot c + \sigma^q(x) + \mathcal{O}(1),\, d\cdot j + \mathcal{O}(1)\,\} + j\cdot c_f + \mathcal{O}(1)\\
&= \max\{\, j\cdot (c + c_f) + \sigma^q(x) + \mathcal{O}(1),\; j\cdot (d + c_f) + \mathcal{O}(1)\,\},
\end{aligned}$$

and for the latter we have:

$$\begin{aligned}
\sigma^q \circ \tau^{j+1} \circ \sigma''(x) &= \max\{\, \max\{\, j\cdot c + \sigma^q(x) + \mathcal{O}(1),\; j\cdot d + \mathcal{O}(1)\,\},\; j\cdot c_f + \mathcal{O}(1)\,\}\\
&= \max\{\, j\cdot c + \sigma^q(x) + \mathcal{O}(1),\; j\cdot d + \mathcal{O}(1),\; j\cdot c_f + \mathcal{O}(1)\,\},\\
&= \max\{\, j\cdot c + \sigma^q(x) + \mathcal{O}(1),\; j\cdot \max\{d, c_f\} + \mathcal{O}(1)\,\}.
\end{aligned}$$

The number of induction steps depends on the size of the expression $\sigma'(x)$ which for $\sigma'(x) = \sigma^q(x)$ does not depend on j. This concludes the proof. □

4. Inexpressibility of copyless CRA over the natural semiring

In this section we show that there exists a function definable by WA (or linear CRA, see Section 2), which is not definable by copyless CRA over the semiring $\mathbb{N}(+,\cdot)$. For this, we use the structural results of copyless CRA introduced in the previous section. Consider the class of functions over the one-letter alphabet $\{a\}$ over the semiring $\mathbb{N}(+,\cdot)$. Since all words are of the form a^n for $n \in \mathbb{N}$ we identify the domain of the functions with $\mathbb{N}$. For WA this class of functions is equivalent to linear recurrence systems (cf. the matrix characterization of weighted automata [10]). In particular, one can define the Fibonacci sequence with a linear and non-copyless CRA as follows. Let $\mathcal{F}$ be the linear CRA with one state and two registers x, y presented in Fig. 3. The only transition updates the registers by $x := y$, $y := x + y$ with the initial values $x = y = 1$ and x defined as the output. Then one can easily see that $[\![\mathcal{F}]\!](a^n) = F_n$, where F_n is the n-th element in the Fibonacci sequence.

Theorem 1. *The Fibonnaci sequence F_n is not recognizable by any copyless CRA.*

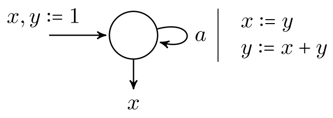


Fig. 3. A CRA recognizing the Fibonacci sequence F_n.

Proof. We will use some simple facts about Fibonacci numbers that are well known or can be easily extracted from the definitions, for a broader discussion see e.g. [13]. Let $\varphi = \frac{1+\sqrt{5}}{2}$ be the golden ratio. Then $F_n = \frac{\varphi^n + (1-\varphi)^n}{\sqrt{5}}$, and the following two properties hold:

$$\lim_{n\to+\infty} \frac{F_{n+k}}{F_n} = \varphi^k \text{ for every fixed } k, \tag{11}$$

$$\varphi^n \notin \mathbb{N} \text{ for any } n > 0. \tag{12}$$

Combining these two properties with results in Section 3 we show that $[\![\mathcal{F}]\!]$ cannot be defined by a copyless CRA. For a contradiction assume that there exists a copyless CRA $\mathcal{A}$ such that $[\![\mathcal{A}]\!] = [\![\mathcal{F}]\!]$. Since $\mathcal{A}$ is deterministic and the alphabet has one letter in every state there is always only one transition. Let q_0 be the initial state of $\mathcal{A}$; ν_0 the initial valuation; and μ its final output function. Then there exists $s, t \in \mathbb{N}$ and a state q such that $\mathcal{A}$ loops in state q when reading a^s; and $\mathcal{A}$ moves from state q_0 to state q when reading a^t. Let σ be the substitution defined by reading a^s from state q; and ρ the substitution defined by reading a^t from q_0. Then $[\![\mathcal{A}]\!](a^{t+i\cdot s}) = \nu_0 \circ \rho \circ \sigma^i \circ \mu(q)$. We will analyze the values of $\sigma^i(x)$ for $x \in \mathsf{Var}(\mu(q))$.

By Proposition 1 we can assume $\mathcal{A}$ is in normal form. Then by Lemma 4 there exists N such that σ^N is a reset substitution. Then by Proposition 3 applied to σ^N and a σ^N-stable register x we have that:

$$\sigma^{N(i+1)}(x) \;=\; c^i \cdot \sigma^N(x) + d \cdot \sum_{j=0}^{i-1} c^j$$

for some $c, d \in \mathbb{N}$. Now σ^N, ν_0 and ρ do not depend on i so we can write that for every $\nu_0 \circ \rho \circ \sigma^N(x) = A_x$, where A_x is a constant not depending on i. Hence we define the sequence $x(i)$ of values of register x

$$x(i) \;=\; \nu_0 \circ \rho \circ \sigma^{N(i+1)}(x) \;=\; c^i \cdot A_x + d \cdot \sum_{j=0}^{i-1} c^j. \tag{13}$$

If $c = 0$ then $x(i) = 0$ for all i. In the remaining cases $c > 0$ and it follows that either $x(i) = 0$ for all i or $x(i) > 0$ for $i > 0$. We will analyze only the latter case, and since we analyze only the asymptotic behavior we assume that $i > 0$. If $c = 1$ we have $\lim_{i\to+\infty} \frac{x(i+1)}{x(i)} = 1$. In the remaining cases $c > 1$ and

$$x(i) \;=\; c^i \cdot A_x + d \cdot \frac{c^i - 1}{c-1}.$$

By taking the limit of $x(i)$ when i tends to infinity we have

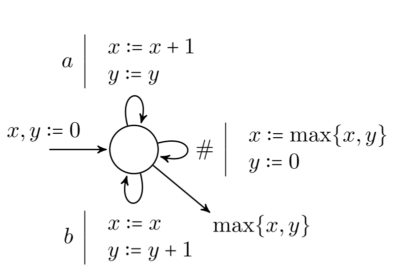
$$\lim_{i\to+\infty} \frac{x(i+1)}{x(i)} \;=\; \lim_{i\to+\infty} \frac{c^{i+1} \cdot A_x + d \cdot \frac{c^{i+1}-1}{c-1}}{c^i \cdot A_x + d \cdot \frac{c^i - 1}{c-1}} \;=\; \lim_{i\to+\infty} \frac{c \cdot A_x + c \cdot d \cdot \frac{1 - \frac{1}{c^{i+1}}}{c-1}}{A_x + d \cdot \frac{1-\frac{1}{c^i}}{c-1}} = c.$$

The goal now is to arrive to a contradiction using properties (11) and (12). Stipulating that $n = t + (i+1) \cdot Ns$ (the number of a's to reach q and make $i+1$ many σ^N loops) and $k = Ns$ (the number of a's to make a single σ^N loop) we analyze the value of $\lim_{i\to+\infty} \frac{[\![\mathcal{A}]\!](a^{n+k})}{[\![\mathcal{A}]\!](a^n)}$. By definition $[\![\mathcal{A}]\!](a^{n+k}) = \nu_0 \circ \rho \circ \sigma^{N(i+1)} \circ \mu(q)$. We can assume $\mu(q) = \sum_l c_l \cdot \prod_j x_{j,l}$ for some constants $c_l \in \mathbb{N}$ (possibly in this form the registers $x_{j,l}$ can repeat and the expression is not copyless).

Now, $\nu_0 \circ \rho \circ \sigma^{N(i+1)} \circ \mu(q) = \sum_l c_l \cdot \prod_j \nu_0 \circ \rho \circ \sigma^{N(i+1)}(x_{j,l})$. For every $x \in \mathsf{Var}(\mu(q))$ if x is stable then its value is $\nu_0 \circ \rho \circ \sigma^{N(i+1)}(x) = x(i)$ as in (13). Otherwise, it is a constant not depending on i. We substitute all such registers with constants and also all registers such that $x(i) = 0$ are replaced with 0. Now, consider $i > 0$ so the remaining registers satisfy $x(i) > 0$. For all registers x their sequences $x(i)$ have the property that $\lim_{i\to+\infty} \frac{x(i+1)}{x(i)}$ is a positive natural number. Consider two sequences $x(i)$ and $y(i)$ with such a property. The following are immediate:

$$\lim_{i\to+\infty} \frac{x(i+1)\cdot y(i+1)}{x(i)\cdot y(i)} \;=\; \lim_{i\to+\infty} \frac{x(i+1)}{x(i)} \cdot \lim_{i\to+\infty} \frac{y(i+1)}{y(i)}$$

$$\lim_{i\to+\infty} \frac{x(i+1)+y(i+1)}{x(i)+y(i)} \;=\; \max\left(\lim_{i\to+\infty} \frac{x(i+1)}{x(i)}, \lim_{i\to+\infty} \frac{y(i+1)}{y(i)}\right).$$



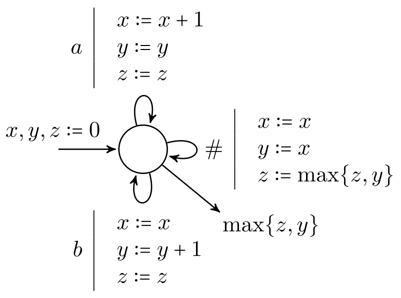


Fig. 4. On the left a copyless cost register automaton $\mathcal{B}$ recognizing $f_{\mathcal{B}}$. On the right a cost-register automaton recognizing $f^R_{\mathcal{B}}$. Notice that the latter automaton is not copyless because it uses x twice when reading #. On the other hand both automata are linear CRA.

It follows that $\lim_{i\to+\infty} \frac{[\![\mathcal{A}]\!](a^{n+k})}{[\![\mathcal{A}]\!](a^n)}$ is a natural number, which is a contradiction with (11) and (12). □

In [18] it is shown that copyless CRA are contained in WA. Given that the Fibonnaci sequence F_n can be defined by a linear CRA and, thus, by a weighted automaton, then Theorem 1 delimit the expressiveness of copyless CRA over the natural semiring: they are strictly less expressive than WA.

Corollary 1. *The class of functions defined by copyless CRA over the natural semiring is strictly contained in the class of functions defined by WA over the natural semiring.*

5. Inexpressibility of copyless CRA over the max-plus semiring

Similar as in the previous section, we use here the techniques introduced in Section 3 to show a function definable by a weighted automaton over the max-plus semiring, that is not definable by any copyless CRA over the same semiring. Apart of delimiting the expressiveness of copyless CRA over the max-plus semiring, this result will show that copyless CRA are not closed under reverse given that the “reverse” of this function is definable by copyless CRA. Before we proceed, notice that the line of reasoning used in the previous section, where functions over one-letter was used, is not possible for automata over the $\mathbb{N}_{-\infty}(\max,+)$ semiring. Indeed, it is known by [17] that over one-letter alphabets weighted automata over $\mathbb{N}_{-\infty}(\max,+)$ semiring can be turned into unambiguous automata. Since by [4,18] we know that unambiguous weighted automata are contained in copyless CRA over any semiring it follows that over one-letter alphabets copyless CRA and weighted automata are equally expressive over the semiring $\mathbb{N}_{-\infty}(\max,+)$. Therefore, to show inexpressibility of copyless CRA over the semiring $\mathbb{N}_{-\infty}(\max,+)$ a more involved proof is needed over a non-unary alphabet.

Consider the function $f_{\mathcal{B}}$ given by the copyless CRA $\mathcal{B}$ over $\Sigma = \{a, b, \#\}$ and $\mathbb{N}_{-\infty}(\max,+)$ in Fig. 4. To understand $f_{\mathcal{B}}$, let us define the output of $\mathcal{B}$ formally. Any $w \in \Sigma^*$ can be written in the form $w = w_0\#w_1\#\ldots\#w_k$ for some $k \geq 0$ and $w_0, \ldots, w_k \in \{a, b\}^*$. So, k is the number of #'s in w. Let n^a_i and n^b_i be the number of a's and b's in w_i, respectively. By the definition of $\mathcal{B}$ in Fig. 4, one can easily check that $f_{\mathcal{B}}$ is defined by $f_{\mathcal{B}}(\epsilon) = 0$ and:

$$f_{\mathcal{B}}(w) \quad = \quad \max_{j\in\{-1,0,\ldots,k\}} \left\{ n^b_j + \sum_{i=j+1}^{k} n^a_i \right\} \tag{14}$$

for $w \neq \epsilon$, where $n^b_{-1} = 0$. 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 changing the interval of the index i from $[j+1, k]$ to $[0, j-1]$ in (14). Formally, one can easily check that $f^R_{\mathcal{B}}$ is defined by $f^R_{\mathcal{B}}(\epsilon) = 0$, and

$$f^R_{\mathcal{B}}(w) \quad = \quad \max_{j\in\{0,\ldots,k,k+1\}} \left\{ \sum_{i=0}^{j-1} n^a_i + n^b_j \right\}, \tag{15}$$

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

Theorem 2. *The function $f^R_{\mathcal{B}}$ is not recognizable by any copyless CRA.*

Proof. Suppose there exists a copyless CRA $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \delta, q_0, \nu_0, \mu)$ which computes the function $f^R_{\mathcal{B}}$. Since the function $f^R_{\mathcal{B}}$ is non-zero (i.e. $f^R_{\mathcal{B}}(w) \neq -\infty$ for every $w \in \Sigma^*$) then by Lemma 2 we can assume that the transitions, initial and final functions in $\mathcal{A}$ do not use $-\infty$. Moreover, since the $\mathbb{N}_{-\infty}(\max,+)$ semiring is also non-zero then additionally we can assume that the registers never store $-\infty$.

Most of the time we will assume that $\mathcal{A}$ is a strongly connected automaton (see Section 3.2). If $\mathcal{A}$ is not strongly connected, we change our analysis by adding a word w_0 from the initial state to a BSCC and then construct a counterexample word from there. Formally, let $\delta^*(q_0, w_0) = (q_0', \sigma_0)$ where q_0' is a state inside a BSCC $Q' \subseteq Q$ of $\mathcal{A}$. We can always redefine $\mathcal{A}$ as follows: q_0' is the new initial state, and the new initialization function ν_0' is defined as $\nu_0'(x) = \nu_0 \circ \sigma_0(x)$. It is straightforward to check that for every word w the new automaton constructed from $\mathcal{A}$ will return the output value of $\mathcal{A}$ over $w_0 \cdot w$. For the sake of simplification, in the following analysis we will assume that $\mathcal{A}$ is strongly connected and later we will include w_0 and σ_0 in the discussion.

We start the proof by analyzing the behavior of $\mathcal{A}$ on words containing cycles of just one letter. Since Q is strongly connected, then there exists a state q_a, a word $v_a = a^{n_a}$ with $n_a > 0$ and a substitution τ_a such that $\delta^*(q_a, v_a) = (q_a, \tau_a)$. We can assume by Lemma 4 that τ_a is a reset substitution, that is, $\mathrm{Var}(\tau_a(x)) = \emptyset$ for all non-stable registers x in τ_a. In addition, define a sequence of words:

$$w_a(j) \;=\; w^{q_a} \cdot v_a^{j+1} \cdot w^{q_a}$$

such that $\delta^*(q_a, w_a(j)) = (q_a, \sigma_a^j)$ for some substitution σ_a^j (i.e. σ_a^j depends on j). Recall that w^{q_a} is the reset word defined in Proposition 2 for the state q_a and σ^{q_a} is an $\mathcal{A}$-reset substitution such that $\delta^*(q_a, w^{q_a}) = (q_a, \sigma^{q_a})$. By Lemma 6 we know that there exist constants c_a^x and d_a^x such that for j big enough:

$$\sigma_a^j \circ \lambda(x) \;=\; \begin{cases} \mathcal{O}(1) & \text{if } x \text{ is non-stable} \\ \max\{\, j \cdot c_a^x + \sigma^{q_a}(x) + \mathcal{O}(1),\ j \cdot d_a^x + \mathcal{O}(1) \,\} & \text{otherwise,} \end{cases} \tag{16}$$

for any $\lambda \in \mathrm{Subs}(\mathcal{A})$ whose size does not depend on j. Analogously, by the definition of $\mathcal{A}$ we can find a state q_b in $\mathcal{A}$, a word $v_b = b^{n_b}$ for some $n_b \geq 0$ and a reset substitution τ_b such that $\delta^*(q_b, v_b) = (q_b, \tau_b)$. Then similar to the sequence $w_a(j)$, we can define the sequence of words:

$$w_b(j) = w^{q_b} \cdot v_b^{j+1} w^{q_b}$$

such that $\delta^*(q_b, w_b(j)) = (q_b, \sigma_b^j)$ for a substitution σ_b^j that satisfies:

$$\sigma_b^j \circ \lambda(x) \;=\; \begin{cases} \mathcal{O}(1) & \text{if } x \text{ is non-stable} \\ \max\{\, j \cdot c_b^x + \sigma^{q_b}(x) + \mathcal{O}(1),\ j \cdot d_b^x + \mathcal{O}(1) \,\} & \text{otherwise,} \end{cases} \tag{17}$$

for any $\lambda \in \mathrm{Subs}(\mathcal{A})$ whose size does not depend on j.

The next step is to understand the growth of stable registers when we repeat the loop $w_a(j)$ several times. For any $j \geq 1$ and $s \geq 1$ we define the sequence of words:

$$w_a(s, j) \;=\; (w^{q_a} \cdot v_a^{j+1})^s \cdot w^{q_a}.$$

Let $\delta^*(q_a, w_a(s, j)) = (q_a, \sigma_a^{s,j})$, then by definition $\sigma_a^{s,j} = (\sigma^{q_a} \circ \tau^{j+1})^s \circ \sigma^{q_a}$. For any natural number $s \geq 1$ if x is a stable register we prove by induction that

$$\sigma_a^{s,j} \circ \lambda(x) \;=\; \max\{\, s \cdot j \cdot c_a^x + \sigma^{q_a}(x) + \mathcal{O}(s),\ (s-1) \cdot j \cdot c_a^x + j \cdot d_a^x + \mathcal{O}(s) \,\}, \tag{18}$$

for any $\lambda \in \mathrm{Subs}(\mathcal{A})$ whose size does not depend on j, and the component hidden in $\mathcal{O}(s)$ depends on s but does not depend on j. For $s = 1$ (18) follows from (16). For $s > 1$ we show:

$$\begin{aligned}
\sigma_a^{s+1,j} \circ \lambda(x) =\ & \sigma^{q_a} \circ \tau^{j+1} \circ \sigma_a^{s,j} \circ \lambda(x) && \text{(by definition)} \\
=\ & \sigma^{q_a} \circ \tau^{j+1} \circ \big(\max\{\, s \cdot j \cdot c_a^x + \sigma^{q_a}(x) + \mathcal{O}(s),\ (s-1) \cdot j \cdot c_a^x + j \cdot d_a^x + \mathcal{O}(s) \,\}\big) && \text{(by induction)} \\
=\ & \max\{\, s \cdot j \cdot c_a^x + \sigma_a^j(x) + \mathcal{O}(s),\ (s-1) \cdot j \cdot c_a^x + j \cdot d_a^x + \mathcal{O}(s) \,\} \\
=\ & \max\{\, s \cdot j \cdot c_a^x + \max\{\, j \cdot c_a^x + \sigma^{q_a}(x) + \mathcal{O}(1),\ j \cdot d_a^x + \mathcal{O}(1) \,\} + \mathcal{O}(s), \\
& \qquad (s-1) \cdot j \cdot c_a^x + j \cdot d_a^x + \mathcal{O}(s) \,\} && \text{(by (16))} \\
=\ & \max\{\, (s+1) \cdot j \cdot c_a^x + \sigma^{q_a}(x) + \mathcal{O}(s),\ s \cdot j \cdot c_a^x + j \cdot d_a^x + \mathcal{O}(s), \\
& \qquad (s-1) \cdot j \cdot c_a^x + j \cdot d_a^x + \mathcal{O}(s) \,\} \\
=\ & \max\{\, (s+1) \cdot j \cdot c_a^x + \sigma^{q_a}(x) + \mathcal{O}(s),\ s \cdot j \cdot c_a^x + j \cdot d_a^x + \mathcal{O}(s) \,\} && \text{(by dominance)}
\end{aligned}$$

We are ready to define the word for which we prove that $\mathcal{A}$ gives the wrong output. Recall that w_0 is a word such that $\delta^*(q_0, w_0) = (q_0', \sigma_0)$, where q_0' is in the BSCC of $\mathcal{A}$. Without loss of generality we assume that $q_0' = q_a$. Let also w^{q_a, q_b} and

w^{q_b,q_a} be the reset words between q_a and q_b from Proposition 2 (given that we are inside a BSCC of $\mathcal{A}$, we can assume that q_b is accessible from q_a and viceversa). For all $j \geq 0$ we define the sequence of words:

$$w(s, j) \;=\; w_0 \cdot w_a(s, j) \cdot w^{q_a,q_b} \cdot w_b(j^2) \cdot w^{q_b,q_a} \cdot w_a(j) \tag{19}$$

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

$$\begin{tikzcd} q_a \arrow[loop left, "w_a(s,j)"] \arrow[r, bend left, "w^{q_a,q_b}"] & q_b \arrow[l, bend left, "w^{q_b,q_a}"] \arrow[loop right, "w_b(j^2)"] \end{tikzcd}$$

The automaton $\mathcal{A}$ starts from reading w_0 to reach the state q_a. Then it cycles in state q_a reading $w_a(s, j)$. After that it moves to q_b reading the reset word w^{q_a,q_b}. Then it cycles in state q_b reading $w_b(j^2)$. Finally, it comes back to the state q_a, where it cycles again. We show that $\mathcal{A}$ outputs wrong value over $w(s, j)$ for some fixed s (to be determined later) and for j big enough.

First we estimate the correct value $f^R_{\mathcal{B}}(w(s, j))$ for any $s, j \geq 0$. By definition the words $w_a(s, j)$, $w_a(j)$ and $w_b(j^2)$ contain blocks of a's and b's separated by reset words. By Proposition 2 each reset word contains the letter #, thus the cycles of a's and b's in the words $w_a(s, j)$, $w_a(j)$, and $w_b(j^2)$ are all separated by #. From the definition of these cycles, one can easily check that the number of a's in $w_a(j)$ is $j \cdot n_a + \mathcal{O}(1)$ and the number of b's in $w_b(j^2)$ is $j^2 \cdot n_b + \mathcal{O}(1)$. Furthermore, the number of a's in $w_a(s, j)$ is equal to $s \cdot j \cdot n_a + \mathcal{O}(1)$. Notice that the only fragments in $w(s, j)$ whose size depends on j are these cycles. The reset words between the cycles are of constant size and there is $\mathcal{O}(s)$ of them. Recall that by (15) $f^R_{\mathcal{B}}(w) = \max_j \left\{ \sum_{i=0}^{j-1} n^a_i + n^b_j \right\}$, where n^a_i and n^b_i are the numbers of a's and b's, respectively, between the #-letters. It is easy to see that for j big enough:

$$f^R_{\mathcal{B}}(w(s, j)) \;=\; n_b \cdot j^2 + n_a \cdot s \cdot j + \mathcal{O}(s) \tag{20}$$

where $\mathcal{O}(s)$ represents the fixed number of a's (their number does not depend on j) that are presented in w_0, w^{q_a}, w^{q_a,q_b} or w^{q_b}. Notice that the last suffix $w_a(j)$ of a's is not contributing into the sum. This is because the sequence $w_b(j^2)$ is overshadowing the last sequence $w_a(j)$, i.e., the max-operator considers the number of b's in $w_b(j^2)$ instead of the number of a's in $w_a(j)$. The rest of the proof is to show that $\mathcal{A}$ does not output the right value on $w(s, j)$. The intuition behind this misbehavior of $\mathcal{A}$ with respect to $w(s, j)$ is that if $\mathcal{A}$ is summing the sequence of a's before $w_b(j^2)$ then it will also add the sequence of a's after $w_b(j^2)$, which by the previous calculations should not happen.

We estimate now the values in the registers of $\mathcal{A}$ after reading the word $w(s, j)$ for j big enough. By the construction of $w(s, j)$ we know that $\delta^*(q_0, w(s, j)) = (q_a, \sigma_{w(s,j)})$, and the final value in the registers of $\mathcal{A}$ after reading $w(s, j)$ is given by composing substitutions corresponding to the composition of the words in (19):

$$\nu_0 \circ \sigma_{w(s,j)} = \nu_0 \circ \sigma_0 \circ \sigma_a^{s,j} \circ \sigma^{q_a,q_b} \circ \sigma_b^{j^2} \circ \sigma^{q_b,q_a} \circ \sigma_a^{j}.$$

For all non-stable registers y this expression is equivalent to $\sigma_a^j(y)$, which is estimated by $\mathcal{O}(1)$ by (16) (where λ is the identity substitution). For a stable register x, the story is much more complicated. We evaluate the expression $\sigma_{w(s,j)}(x)$ step-by-step to estimate its value. First, by (16):

$$\sigma_a^j(x) \;=\; \max\{\, j \cdot c_a^x + \sigma^{q_a}(x) + \mathcal{O}(1),\; j \cdot d_a^x + \mathcal{O}(1) \,\}.$$

Then by composing $\sigma_b^{j^2} \circ \sigma^{q_b,q_a}$ with $\sigma_a^j(x)$ we get:

$$\begin{aligned}
\sigma_b^{j^2} \circ \sigma^{q_b,q_a} \circ \sigma_a^j(x) \;&=\; \sigma_b^{j^2} \circ \sigma^{q_b,q_a} \circ \big(\max\{\, j \cdot c_a^x + \sigma^{q_a}(x) + \mathcal{O}(1),\; j \cdot d_a^x + \mathcal{O}(1) \,\}\big) \\
&=\; \max\{\, j \cdot c_a^x + \sigma_b^{j^2} \circ \sigma^{q_b,q_a} \circ \sigma^{q_a}(x) + \mathcal{O}(1),\; j \cdot d_a^x + \mathcal{O}(1) \,\} \\
&=\; \max\{\, j \cdot c_a^x + \max\{\, j^2 \cdot c_b^x + \sigma^{q_b}(x) + \mathcal{O}(1),\; j^2 \cdot d_b^x + \mathcal{O}(1) \,\} + \mathcal{O}(1),\; j \cdot d_a^x + \mathcal{O}(1) \,\} \qquad (\heartsuit) \\
&=\; \max\left\{\, j \cdot c_a^x + j^2 \cdot c_b^x + \sigma^{q_b}(x) + \mathcal{O}(1),\; j \cdot c_a^x + j^2 \cdot d_b^x + \mathcal{O}(1),\; j \cdot d_a^x + \mathcal{O}(1) \right\}
\end{aligned}$$

There is only one nontrivial equality ($\heartsuit$). It holds by (17) and by considering $\lambda = \sigma^{q_b,q_a} \circ \sigma^{q_a}$. The next step is to compose $\sigma_a^{s,j} \circ \sigma^{q_a,q_b}$ with $\sigma_b^{j^2} \circ \sigma^{q_b,q_a} \circ \sigma_a^j$. We denote this composition by $\sigma^{-w_0}_{w(s,j)}$:

$$\begin{aligned}
\sigma^{-w_0}_{w(s,j)}(x) \;&=\; \sigma_a^{s,j} \circ \sigma^{q_a,q_b} \circ \big(\sigma_b^{j^2} \circ \sigma^{q_b,q_a} \circ \sigma_a^j(x)\big) \\
&=\; \sigma_a^{s,j} \circ \sigma^{q_a,q_b} \circ \max\big\{\, j \cdot c_a^x + j^2 \cdot c_b^x + \sigma^{q_b}(x) + \mathcal{O}(1),
\end{aligned}$$

$$
\begin{aligned}
& \qquad j \cdot c_a^x + j^2 \cdot d_b^x + \mathcal{O}(1),\ j \cdot d_a^x + \mathcal{O}(1) \big\} \\
= \ & \max \big\{ \ j \cdot c_a^x + j^2 \cdot c_b^x + \sigma_a^{s,j} \circ \sigma^{q_a,q_b} \circ \sigma^{q_b}(x) + \mathcal{O}(1), \\
& \qquad j \cdot c_a^x + j^2 \cdot d_b^x + \mathcal{O}(1),\ j \cdot d_a^x + \mathcal{O}(1) \big\} \\
= \ & \max \big\{ \ j \cdot c_a^x + j^2 \cdot c_b^x + \max \big\{ \ s \cdot j \cdot c_a^x + \sigma^{q_a}(x) + \mathcal{O}(s),\ (s-1) \cdot j \cdot c_a^x + j \cdot d_a^x + \mathcal{O}(s) \ \big\} + \mathcal{O}(1), \quad (\diamond) \\
& \qquad j \cdot c_a^x + j^2 \cdot d_b^x + \mathcal{O}(1),\ j \cdot d_a^x + \mathcal{O}(1) \big\} \\
= \ & \max \big\{ \ j \cdot c_a^x + j^2 \cdot c_b^x + s \cdot j \cdot c_a^x + \sigma^{q_a}(x) + \mathcal{O}(s),\ j \cdot c_a^x + j^2 \cdot c_b^x + (s-1) \cdot j \cdot c_a^x + j \cdot d_a^x + \mathcal{O}(s), \\
& \qquad j \cdot c_a^x + j^2 \cdot d_b^x + \mathcal{O}(1),\ j \cdot d_a^x + \mathcal{O}(1) \big\} \\
= \ & \max \big\{ \ (s+1) \cdot j \cdot c_a^x + j^2 \cdot c_b^x + \sigma^{q_a}(x) + \mathcal{O}(s),\ j \cdot (s \cdot c_a^x + d_a^x) + j^2 \cdot c_b^x + \mathcal{O}(s), \\
& \qquad j \cdot c_a^x + j^2 \cdot d_b^x + \mathcal{O}(1),\ j \cdot d_a^x + \mathcal{O}(1) \big\}
\end{aligned}
$$

Like before all equalities are routine except for ($\diamond$). It holds by (18) and by considering $\lambda = \sigma^{q_a,q_b} \circ \sigma^{q_b}$. We compose $\sigma_{w(s,j)}^{-w_0}$ with $\nu_0 \circ \sigma_0(x)$, Given that $\nu_0 \circ \sigma_0(x)$ is a constant with respect to j (i.e. $\nu_0 \circ \sigma_0(x) = \mathcal{O}(1)$), we have that $\nu_0 \circ \sigma_{w(s,j)}$ over a stable register x in $\mathcal{A}$ is equal to:

$$
\begin{aligned}
\nu_0 \circ \sigma_{w(s,j)}(x) \ = \ & \max \big\{ \ (s+1) \cdot j \cdot c_a^x + j^2 \cdot c_b^x + \mathcal{O}(s),\ j \cdot (s \cdot c_a^x + d_a^x) + j^2 \cdot c_b^x + \mathcal{O}(s), \\
& \qquad j \cdot c_a^x + j^2 \cdot d_b^x + \mathcal{O}(1),\ j \cdot d_a^x + \mathcal{O}(1) \big\} \qquad (21)
\end{aligned}
$$

For the rest of the proof, let us fix the value for s that does not depend on j (the exact value of s will be defined at the very end of the proof). For a fixed s we denote the quadratic functions on j by:

$$
\begin{aligned}
g_1^x(j) &= c_b^x \cdot j^2 + c_a^x \cdot (s+1) \cdot j + \mathcal{O}(1) \\
g_2^x(j) &= c_b^x \cdot j^2 + (c_a^x \cdot s + d_a^x) \cdot j + \mathcal{O}(1) \\
g_3^x(j) &= d_b^x \cdot j^2 + c_a^x \cdot j + \mathcal{O}(1) \\
g_4^x(j) &= d_a^x \cdot j + \mathcal{O}(1)
\end{aligned}
$$

where $\mathcal{O}(1)$ denotes some constant that does not depend on j (recall that s is fixed). Then for every stable register x we have

$$
\nu_0 \circ \sigma_{w(s,j)}(x) = \max_{i=1}^{4} \{ g_i^x(j) \}.
$$

Notice that if $d_a^x < c_a^x$ then for j big enough it holds that $g_2^x(j) \leq g_1^x(j)$ and $g_2^x(j)$ is dominated by $g_1^x(j)$. This motivates the definition of e_a^x, given by $e_a^x = d_a^x - c_a^x$ when $d_a^x \geq c_a^x$ and $e_a^x = 0$ otherwise. Furthermore, redefine $g_2^x(j)$ by:

$$
g_2^x(j) \ = \ c_b^x \cdot j^2 + (c_a^x \cdot (s+1) + e_a^x) \cdot j + \mathcal{O}(1)
$$

It is easy to see that $\nu_0 \circ \sigma_{w(s,j)}(x) = \max_{i=1}^4 \{ g_i^x(j) \}$ holds for the new definition of $g_2^x(j)$.

Towards the end of the proof, we study the output function $[\![\mathcal{A}]\!](w(s,j)) = \nu_0 \circ \sigma_{w(s,j)} \circ \mu(q_a)$. By Lemma 1, we know that the copyless expression $\mu(q_a)$ can be presented as an expression of the form:

$$
\mu(q_a) \ = \ \max_{i=1}^{k} \left\{ \ell_i + \sum_{x \in X_i} x \right\},
$$

where $X_1, \ldots, X_k$ is a sequence of different sets over $\mathcal{X}$ and $\ell_1, \ldots, \ell_k$ is a sequence of values over $\mathbb{N}$ for $k \geq 0$. Notice that in this representation the expression $\mu(q_a)$ might be not copyless. By the previous analysis we know that for j big enough $\nu_0 \circ \sigma_{w(s,j)}(x)$ is either estimated by $\mathcal{O}(1)$ when x is non-stable or equal to $\max_{i=1}^4 \{ g_i^x(j) \}$ when x is stable. Then by composing $\nu_0 \circ \sigma_{w(s,j)}$ with $\mu(q_a)$ we get:

$$
\begin{aligned}
[\![\mathcal{A}]\!](w(s,j)) \ &= \ \nu_0 \circ \sigma_{w(s,j)} \circ \mu(q_a) \\
&= \ \nu_0 \circ \sigma_{w(s,j)} \circ \max_{i=1}^{k} \Big\{ \ell_i + \sum_{x \in X_i} x \Big\} \\
&= \ \max_{i=1}^{k} \Big\{ \ \ell_i + \sum_{x \in X_i} \nu_0 \circ \sigma_{w(s,j)}(x) \ \Big\}
\end{aligned}
$$

$$= \max_{i=1}^{h} \left\{ n_i^b + \sum_{x \in Y_i} \max_{i=1}^{4} \{ g_i^x(j) \} \right\} \tag{22}$$

where $Y_1, \dots, Y_h$ is a new sequence of subsets of stable registers and $m_1, \dots, m_h$ is a sequence of values over $\mathbb{N}$ for $k \geq 0$. The last equality holds since $\nu_0 \circ \sigma_{w(s,j)}(x)$ are constants for non-stable registers. Thus we can sum the constants with ℓ_i and get new constants n_i^b; and regroup the sets X_i to form new sets of stable registers Y_i. Notice that we might need multiple copies of the same Y_i.

The next step is to further simplify the output $[\![\mathcal{A}]\!](w(s,j))$. For this, note that $[\![\mathcal{A}]\!](w(s,j))$ is the max and sum of quadratic functions over j. Then for j big enough there exists a constant $i^* \leq h$ and a partition of $Y_{i^*} = Z_1 \uplus Z_2 \uplus Z_3 \uplus Z_4$ such that:

$$[\![\mathcal{A}]\!](w(s,j)) \; = m_{i^*} + \sum_{x \in Z_1} g_1^x(j) + \sum_{x \in Z_2} g_2^x(j) + \sum_{x \in Z_3} g_3^x(j) + \sum_{x \in Z_4} g_4^x(j).$$

The index i^* is the one where (22) is maximized for j big enough and the partition $Y_{i^*} = Z_1 \uplus Z_2 \uplus Z_3 \uplus Z_4$ is the division of Y_{i^*} where each function g_i^x dominates for j big enough. Thus, $[\![\mathcal{A}]\!](w(s,j))$ is a sum of quadratic functions and by summing common j-terms we can reduce $[\![\mathcal{A}]\!](w(s,j))$ to a polynomial of the form $B \cdot j^2 + A \cdot j + C$. Intuitively, the value $B \cdot j^2$ should correspond to the number of b's in $w(s,j)$ and $A \cdot j$ to the number of a's in $w(s,j)$. In the last part of this proof, we analyze the A-coefficient and compare it with the corresponding coefficient in $f_{\mathcal{B}}^R(w(s,j))$.

Recall from (20) that the output of $f_{\mathcal{B}}^R$ over $w(s,j)$ is equal to $n_b \cdot j^2 + n_a \cdot s \cdot j + \mathcal{O}(s)$. If the output $[\![\mathcal{A}]\!](w(s,j))$ is correct then we should have $A = n_a \cdot s$. By adding the linear coefficients of $g_i^x(j)$ for $i \in \{1,2,3,4\}$ and $x \in Z_i$ we get that

$$\begin{aligned} A \;&=\; \sum_{x \in Z_1} c_a^x \cdot (s+1) \;+\; \sum_{x \in Z_2} \left(c_a^x \cdot (s+1) \;+\; e_a^x\right) \;+\; \sum_{x \in Z_3} c_a^x \;+\; \sum_{x \in Z_4} d_a^x \\ &=\; (s+1) \cdot \sum_{x \in Z_1 \cup Z_2} c_a^x \;+\; \sum_{x \in Z_2} e_a^x \;+\; \sum_{x \in Z_3} c_a^x \;+\; \sum_{x \in Z_4} d_a^x. \end{aligned}$$

Given that A should be equal to $n_a \cdot s$ this implies that $\sum_{x \in Z_1 \cup Z_2} c_a^x < n_a$. Otherwise, $A > n_a \cdot s$. Therefore, $\sum_{x \in Z_1 \cup Z_2} c_a^x \leq n_a - 1$ and thus

$$(s+1) \cdot \sum_{x \in Z_1 \cup Z_2} c_a^x \;\leq\; (s+1) \cdot (n_a - 1)$$

By replacing this fact in the definition of A, we can over-approximate the A-coefficient as follows

$$\begin{aligned} A \;&\leq\; (s+1) \cdot (n_a - 1) \;+\; \sum_{x \in Z_2} e_a^x \;+\; \sum_{x \in Z_3} c_a^x \;+\; \sum_{x \in Z_4} d_a^x \\ &\leq\; s \cdot n_a \;-\; s \;+\; n_a \;-\; 1 \;+\; \sum_{x \in \mathcal{X}} (e_a^x + c_a^x + d_a^x). \end{aligned} \tag{23}$$

For the last bound, recall that Z_i are pairwise disjoint subsets. It is important to notice that the inequality (23) does not depend on how we choose s.

We are ready to show that, by suitably choosing a fix value for s, A will not be equal to $n_a \cdot s$. It suffices to choose $s > n_a - 1 + \sum_{x \in \mathcal{X}} (e_a^x + c_a^x + d_a^x)$. This with (23) implies that $A < n_a \cdot s$ which is a contradiction. In other words, $\mathcal{A}$ is not counting all the a's in $w(s,j)$ for j big enough. □

From Theorem 2 we get that copyless CRA over max-plus semiring is strictly less expressive than WA: we know that copyless CRA are contained in WA [18] and by Fig. 4 the function $f_{\mathcal{B}}^R$ is definable by a linear CRA and thus by a WA. It is also well-known [24,10] that the class of functions defined by WA is closed under reverse.

Corollary 2. *The class of functions defined by copyless CRA over the max-plus semiring is strictly contained in the class of functions defined by WA over the max-plus semiring.*

Moreover, from Theorem 2 we immediately get the following corollary.

Corollary 3. *The class of functions recognizable by copyless CRA over the max-plus semiring is not closed under reverse.*

6. 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 2 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 [18] 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 $\mathrm{Expr}_N(\mathcal{X}) = \mathrm{Expr}_N^{\oplus}(\mathcal{X}) \cup \mathrm{Expr}_N^{\odot}(\mathcal{X})$.

We say that a copyless CRA $\mathcal{A}$ has bounded alternation if 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 NLogSpacein the size of a copyless CRA.

Proposition 4. *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.

Proof. Let $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \Delta, q_0, \nu_0, \mu)$ be a copyless CRA. We define the graph $\mathcal{G}_{\mathcal{A}} = (V_{\mathcal{A}}, E_{\mathcal{A}})$ in which we look for cycles that produce unbounded alternation in $\mathcal{A}$. The set of vertices $V_{\mathcal{A}}$ of $\mathcal{G}_{\mathcal{A}}$ are triples in $Q \times \mathcal{X} \times \{\oplus, \odot\}$. Each vertex $(q, x, \otimes) \in V_{\mathcal{A}}$ keeps track of the current state q, a register x, and the last operation $\otimes$ seen in the last transition. We define $E_{\mathcal{A}} \subseteq (Q \times \mathcal{X} \times \{\oplus, \odot\})^2$ such that $((q_1, x_1, \otimes_1), (q_2, x_2, \otimes_2)) \in E_{\mathcal{A}}$ if, and only if: (1) $\delta(q_1, a) = (q_2, \sigma)$ for some $a \in \Sigma$, (2) $x_1 \in \mathrm{Var}(\sigma(x_2))$, and (3) $\otimes_2$ is equal to $\otimes$ whenever $\sigma(x_2) = e_1 \otimes e_2$ for some expressions e_1 and e_2, or equal to $\otimes_1$ otherwise. Intuitively, $\otimes_2$ keeps track of the last operation seen when x_1 passes its value to x_2 in the expression $\sigma(x_2)$.

Furthermore, we define $E_{\mathcal{A}}^{\mathrm{alt}} \subseteq E_{\mathcal{A}}$ that indicates edges that produce an alternation of operations. Precisely, $((q_1, x_1, \otimes_1), (q_2, x_2, \otimes_2)) \in E_{\mathcal{A}}^{\mathrm{alt}}$ if, and only if, $((q_1, x_1, \otimes_1), (q_2, x_2, \otimes_2)) \in E_{\mathcal{A}}$ and there exists a subexpression $e_1 \otimes e_2$ in $\sigma(x_2)$ with $x_1 \in \mathrm{Var}(e_1 \otimes e_2)$ and $\otimes \neq \otimes_1$. In other words, there exists an alternation with respect to the last operation $\otimes_1$ in the transition from q_1 to q_2.

Let $V_{\mathcal{A}}^{\mathrm{alt}}$ be the set of all $(q, x, \otimes) \in V_{\mathcal{A}}$ such that $x \in \mathrm{Var}(\mu(q))$ and let $\mathcal{G}'_{\mathcal{A}} = (V'_{\mathcal{A}}, E'_{\mathcal{A}})$ be the subgraph of $\mathcal{G}_{\mathcal{A}}$ induced by vertices that can reach $V_{\mathcal{A}}^{\mathrm{alt}}$ in $\mathcal{G}_{\mathcal{A}}$. It is straightforward to prove that if $\mathcal{G}'_{\mathcal{A}}$ has a cycle with an $E_{\mathcal{A}}^{\mathrm{alt}}$-edge then $\mathcal{A}$ has unbounded alternation. Conversely, note that if $\mathcal{G}'_{\mathcal{A}}$ do not have a cycle with an $E_{\mathcal{A}}^{\mathrm{alt}}$-edge, then any path can cross at most $|Q|$ $E_{\mathcal{A}}^{\mathrm{alt}}$-edges, each with at most $\max\{\mathrm{alt}(\sigma) \mid \exists. p, q \in Q\,.\ \delta(q,a) = (p,\sigma)\}$ alternations. These properties can be checked in NLogSpace since $\mathcal{G}_{\mathcal{A}}$ can be generated on the fly in logarithmic space. $\square$

6.1. 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 require 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.

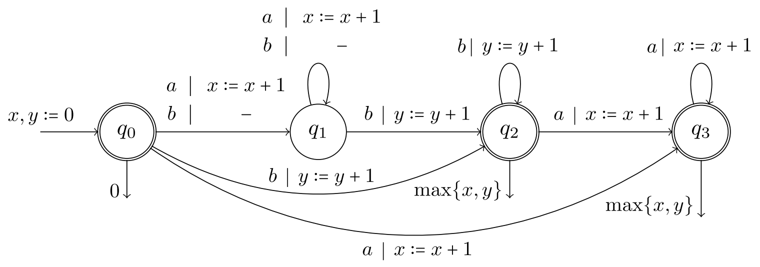


Fig. 5. Unambiguous BAC recognizing f from Example 4.

Example 4. Fix $\Sigma = \{a, b\}$. For every $w \in \Sigma^*$ let $\#_a(w)$ denote the number of a's in w and $\#_b^{last}(w)$ denote the length of the last contiguous subsequence of b's (e.g. $\#_b^{last}(ab^2ab^4ab^3aa) = 3$). Consider the function $f : \Sigma^* \to \mathbb{N}$ defined as $f(w) = \max(\#_a(w), \#_b^{last}(w))$. This function can be easily defined by a nondeterministic unambiguous BAC in Fig. 5. The automaton keeps the number of a's in register x and the number of b's in the last subsequence in y. To do that the automaton guesses that the word reached the last contiguous subsequence of b's (i.e. state q_2) and then it continues only if the remaining letters are a (i.e. state q_3). For simplicity, assignments keeping the value ($x := x$) were omitted in the picture. All transitions are labeled with the alphabet letter and the register assignments. If both assignments keep the value of the register this is denoted by "$-$".

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. In particular, one can define f from Example 4 without unambiguous nondeterminism.

Theorem 3. *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 of Theorem 3. To start we need to introduce some notation for trees, expressions, and substitutions that will be the main objects during the proof.

Trees. Let Σ be a set of labels. An (unordered) labeled Σ-tree t is a finite function $t : \text{nodes}(t) \to \Sigma$ such that nodes(t) is a finite prefix-closed subset of $\mathbb{N}^*$ (i.e. $w \in \text{nodes}(t)$ whenever $w \cdot i \in \text{nodes}(t)$ for some $i \in \mathbb{N}$). We say that ϵ is the root of t and $w \cdot i \in \text{nodes}(t)$ is a child of w. For any $w \in \text{nodes}(t)$, we denote by $t[w]$ the subtree rooted at w, i.e. $t[w](i) = t(w \cdot i)$ for every $i \in \text{nodes}(t[w])$. For every $a \in \Sigma$ we write $a\{t_1, \dots, t_k\}$ to denote a tree whose root is labeled by a and $t_1, \dots, t_k$ are the subtrees hanging from the root. We say that $w \in \text{nodes}(t)$ is an internal node of t if $w \cdot i \in \text{nodes}(t)$ for some $i \in \mathbb{N}$. Otherwise, w is called a leaf of t and the set of all leaves of t is denoted by leaves(t). We say that a tree is *complete* if every internal nodes has at least two children. One can easily check that if t is a complete tree, then $|\text{nodes}(t)| \leq 2 \cdot |\text{leaves}(t)|$. Finally, we denote by Trees(Σ) the set of all Σ-trees.

Expressions and substitutions. From now on, we see expressions as unordered labeled trees by exploiting the commutativity and associativity of the semiring (recall that in this paper we always assume that semirings are commutative). Formally, for $\otimes \in \{\oplus, \odot\}$ we define $\text{Expr}^{\otimes}(\mathcal{X})$ as the minimal set such that $S \cup \mathcal{X} \subseteq \text{Expr}^{\otimes}(\mathcal{X})$ and $\otimes\{e_1, \dots, e_k\} \in \text{Expr}^{\otimes}(\mathcal{X})$ for every $e_1, \dots, e_k \in \text{Expr}^{\bar{\otimes}}(\mathcal{X})$ with $k \geq 1$ (recall that $\bar{\otimes}$ is the dual operation of $\otimes$ in $\mathbb{S}$). Indeed, any expression can be represented by a unique unordered tree in $\text{Expr}^{\oplus}(\mathcal{X})$ or $\text{Expr}^{\oplus}(\mathcal{X})$. For example, the expression $((x \odot (y \odot 2)) \oplus 3) \oplus (z \odot 4)$ can be represented by:

$$\oplus\Big\{ \odot\{x, y, 2\},\ 3,\ \odot\{z, 4\} \Big\} \tag{24}$$

Intuitively, the unordered tree encodes the expression by removing the parenthesis and the order of multiplication and addition. For the sake of simplification, in the sequel we represent every expression with its canonical representation in $\text{Expr}^{\oplus}(\mathcal{X}) \cup \text{Expr}^{\odot}(\mathcal{X})$ and we associate $\text{Expr}(\mathcal{X})$ with $\text{Expr}^{\oplus}(\mathcal{X}) \cup \text{Expr}^{\odot}(\mathcal{X})$.

For two disjoint set of variables $\mathcal{X}_1$ and $\mathcal{X}_2$, we define $\text{Subs}(\mathcal{X}_1, \mathcal{X}_2)$ to be the set of all copyless substitutions $\sigma : \mathcal{X}_1 \to \text{Expr}(\mathcal{X}_1 \cup \mathcal{X}_2)$, that is, copyless substitutions where the domain contains only variables in $\mathcal{X}_1$. Here, composition between substitutions in $\text{Subs}(\mathcal{X}_1, \mathcal{X}_2)$ is defined in a straightforward way where $\mathcal{X}_2$-variables are treated as constants. Notice that the composition of two substitutions is copyless for the registers $\mathcal{X}_1$ but not necessarily for the registers $\mathcal{X}_2$. This is because the registers $\mathcal{X}_2$ are not in the domain of these substitutions.

Substitution trees. We denote by $\text{Trees}(\mathcal{X}_1, \mathcal{X}_2)$ the set of all trees labeled by substitutions in $\text{Subs}(\mathcal{X}_1, \mathcal{X}_2)$. Furthermore, we say that $t \in \text{Trees}(\mathcal{X}_1, \mathcal{X}_2)$ is *copyless* if $t(u)$ is copyless for every $u \in \text{nodes}(t)$ and:

$$\text{Var}(t(u)) \cap \text{Var}(t(v)) \subseteq \mathcal{X}_1 \tag{25}$$

for every $u, v \in \text{nodes}(t)$ where, for any substitution σ, $\text{Var}(\sigma)$ are the variables mentioned in any expression of σ. In other words, each $\mathcal{X}_2$-variable is used at most once in a substitution tree t (note that there is no restriction in $\mathcal{X}_1$). With the condition (25), the composition of substitutions between different nodes is also copyless for the set $\mathcal{X}_2$. Indeed, for every two different nodes u and v the substitutions $t(u), t(v) : \mathcal{X}_1 \to \text{Expr}(\mathcal{X}_1 \cup \mathcal{X}_2)$ are copyless and do not share variables from $\mathcal{X}_2$. Thus, $t(u) \circ t(v)$ will still contain each variable from $\mathcal{X}_2$ at most once. We say that $t \in \text{Trees}(\mathcal{X}_1, \mathcal{X}_2)$ is *constant-free* if, for every $u \in \text{nodes}(t)$ the substitution $t(u)$ does not use elements from $\mathcal{S}$. Finally, for any node $u \in \text{nodes}(t)$ we define the *collapse operation* $t^{\downarrow}(u)$ such that:

$$t^{\downarrow}(u) = t(\epsilon) \circ t(u[\cdot, 1]) \circ \ldots \circ t(u[\cdot, k])$$

where $k = |u|$ and $u[\cdot, i]$ is the prefix of u until position i. In other words, $t^{\downarrow}(u)$ is the composition of all substitution along the branch from the root until u. Note that, by condition (25), this composition always produces a copyless substitution.

Before going into the details of the proof we recall the following lemma which is a well-known property of unambiguous finite automata and, in particular, of unambiguous CRA.

Lemma 7. *[24,26] For every different runs ρ and ρ' of an unambiguous finite automaton $\mathcal{A}$ over $w \in \Sigma^*$, the last states of ρ and ρ' are different, that is, $\rho(|w|) \neq \rho'(|w|)$.*

Let $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \delta, I_0, V_0, F, \mu)$ be an unambiguous BAC whose alternation is bounded by N. To construct a deterministic BAC $\mathcal{A}'$ from $\mathcal{A}$, the idea is to execute all runs of $\mathcal{A}$ over a word in parallel, simulating some sort of subset construction [14]. The problem here is that we cannot make arbitrary copies of registers (recall that $\mathcal{A}'$ must be copyless) and then we cannot simulate directly with a deterministic automaton the arbitrary branching of runs, namely, non-deterministic transitions. To solve this, we keep in states a tree of runs that encodes how runs are branching when the word is read. Of course, if we keep the branching of all runs in memory, the tree would be unbounded. A characteristic property of unambiguous CRA (Lemma 7) is that, for each prefix, there are at most $|Q|$ partial runs and, moreover, all leading states are different. These two facts suggest a tree structure of the runs where each branch is a run and where the number of branches is bounded by $|Q|$. Although we can keep in finite memory the branching structure of the partial runs of $\mathcal{A}$, we cannot do the same trick for registers and naively keep copies of registers for each run (recall again that $\mathcal{A}'$ must be copyless). To overcome this problem, $\mathcal{A}'$ will postpone the evaluation of registers by keeping substitutions inside the internal tree structure of runs (i.e. substitutions trees). Clearly, $\mathcal{A}'$ cannot postpone these substitutions forever and store a long sequence of these objects with finite memory. The key idea here is to prune and reduce the tree structure by doing partial evaluation of the substitutions whenever is possible. We show that by exploiting the bounded alternation of the output and the copyless restriction of $\mathcal{A}$, we need only a finite amount of memory to remember the tree structure and the substitutions of all runs. Finally, for the sake of simplification we present the main construction for the restricted case when $\mathcal{A}$ has only one initial state, namely, $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \delta, q_0, \nu_0, F, \mu)$ where q_0 is the only initial state and the initialization valuation is just $\nu_0 : \mathcal{X} \to \mathbb{S}$.

Let $\mathcal{X}$ be the set of registers in $\mathcal{A}$ and $\hat{\mathcal{X}} = \{\hat{x} \mid x \in \mathcal{X}\}$ a disjoint copy of $\mathcal{X}$. We construct a deterministic BAC $\mathcal{A}' = (Q', \Sigma, \mathcal{Y}, \delta', q_0', \nu_0', \mu')$ as follows.

- Q' is the set of all pairs (t, B) where $t \in \text{Trees}(\mathcal{X}, \mathcal{Y})$ is a complete, copyless, constant-free substitution tree and $B : \text{leaves}(t) \to Q$ is an injective function.
- $\mathcal{Y}$ is a set of registers of size $|\mathcal{Y}| = 2 \cdot |Q| \cdot |\mathcal{X}| \cdot (|\mathcal{X}| \cdot N + 1)$ satisfying $\hat{\mathcal{X}} \subseteq \mathcal{Y}$ and $\mathcal{X} \cap \mathcal{Y} = \emptyset$.
- $q_0' = (t_0, B_0)$ is the initial state of $\mathcal{A}'$ where t_0 is a single-node tree labeled with $\sigma_0 \in \text{Subs}(\mathcal{X}, \mathcal{Y})$ and $B_0(\epsilon) = q_0$ such that $\sigma_0(x) = \hat{x}$ for every $x \in \mathcal{X}$.
- $\nu_0' : \mathcal{Y} \to \mathbb{S}$ is the initial substitution such that $\nu_0'(\hat{x}) = \nu_0(x)$ for all $x \in \mathcal{X}$ and $\nu_0'(y) = \mathbb{0}$ for all $y \in \mathcal{Y} \setminus \hat{\mathcal{X}}$.
- μ' is the final substitution such that $\mu'((t, B)) = t^{\downarrow}(u) \circ \mu(q)$ for every $(t, B) \in Q'$ whenever $u \in \text{leaves}(t)$ is the only leaf satisfying $B(u) \in F$ and $q = B(u)$.
- δ' is the transition function defined below.

We start explaining the relation between the set of registers $\hat{\mathcal{X}}$, $\mathcal{X}$ and $\mathcal{Y}$. The $\mathcal{X}$-variables in the construction are used inside the internal structure of $\mathcal{A}'$ (i.e. states). In fact, $\mathcal{X}$-variables will never be used as real variables by $\mathcal{A}'$. They will just be used to keep track of temporary substitutions. Regarding $\mathcal{Y}$, the decision of taking the size of $\mathcal{Y}$ equal to $2 \cdot |Q| \cdot |\mathcal{X}| \cdot (|\mathcal{X}| \cdot N + 1)$ is technical and will be clear later in the proof. Regarding $\hat{\mathcal{X}}$, the property $\hat{\mathcal{X}} \subseteq \mathcal{Y}$ is needed for the definition of the initial substitution. Besides that we will not use the fact that $\hat{\mathcal{X}} \subseteq \mathcal{Y}$.

Recall that each state in Q' is composed by a complete, copyless, and constants free substitution tree t and an injective function $B : \text{leaves}(t) \to Q$. As was suggested before, t keeps track of the branching history of the partial runs of $\mathcal{A}$ and B labels leaves of t with states in Q. The plan here is that each partial run of $\mathcal{A}$ is represented by a different branch of t and the states assigned by B represent the current state of each run.

Next, we show that Q' is a finite set. Indeed, we can bound the size of any state $(t, B) \in Q'$ in terms of $|Q|$, $|\mathcal{X}|$, and $|\mathcal{Y}|$. First, the number of nodes in t is bounded by $2 \cdot |Q|$, that is, $|\text{nodes}(t)| \leq 2 \cdot |\text{leaves}(t)| \leq 2 \cdot |Q|$ since t is a complete tree and B is an injective function over leaves(t). Second, each copyless and constants-free expression that one can write with $|\mathcal{X} \cup \mathcal{Y}|$ variables is of size at most $2 \cdot (|\mathcal{X}| + |\mathcal{Y}|)$ (i.e. the expression has no constant, and each variable can be used at most once). Third, the size of any copyless and constants-free substitution $\sigma \in \text{Subs}(\mathcal{X}, \mathcal{Y})$ is bounded by $|\sigma| = \mathcal{O}(2 \cdot |\mathcal{X}| \cdot (|\mathcal{X}| + |\mathcal{Y}|))$ and thus the number of possible labels for t is finite, which implies that $|t|$ is of bounded size. Fourth, B is an injective function from leaves(t) into Q and, given that t is of bounded size, then B is also bounded. Finally, given that the size of t and B is bounded by $|Q|$, $|\mathcal{X}|$, and $|\mathcal{Y}|$, then we can conclude that Q' is a finite set. Furthermore, by composing and counting the previous arguments, one can show that $|Q'| = 2^{\mathcal{O}(|Q| \cdot |\mathcal{X}| \cdot |Y|^2)} = 2^{\mathcal{O}(|Q|^3 \cdot |\mathcal{X}|^5 \cdot N^2)}$.

For a given word w, consider the set of partial runs in $\mathcal{A}$. By Lemma 7 every partial run on w is determined by its last state. Then, we define $Q(w) \subseteq Q$ as the set of states in the partial runs of $\mathcal{A}$ over w. To understand the main purpose of (t, B), we state the following Lemma. Its proof is postponed to the end of this section.

Lemma 8. *Let ρ be a run of $\mathcal{A}$ over w and (q, ν) be the last configuration of ρ. Then the run of $\mathcal{A}'$ over w reaches configuration $((t, B), \xi)$, where ξ is a valuation over $\mathcal{Y}$. Moreover $Q(w) = range(B)$ and for every $q \in Q(w)$ there exists $u \in$ leaves(t) such that $B(u) = q$ and $\nu = \xi \circ t^{\downarrow}(u)$.*

In other words, from a state (t, B) we can recover the configuration (q, ν) from a leaf u, by applying B over u and composing the substitutions from the root to u.

A key notion for defining the transition function δ' is the notion of an $\mathcal{X}$-reduction. An $\mathcal{X}$-reduction of an expression $e \in \text{Expr}(\mathcal{X} \cup \mathcal{Y})$ is a tuple (r, σ) where $r \in \text{Expr}(\mathcal{X} \cup \mathcal{Y})$ is a copyless expression without constants and $\sigma : \mathcal{Y} \rightharpoonup \text{Expr}(\mathcal{Y})$ is a partial substitution (i.e. partial function) such that $e = \hat{\sigma}(r)$. Notice that $\text{dom}(\sigma) \cap \mathcal{X} = \emptyset$ since we assumed $\mathcal{X} \cap \mathcal{Y} = \emptyset$. The goal of an $\mathcal{X}$-reduction is to factorize constants into $\mathcal{Y}$-variables. Of course, an $\mathcal{X}$-reduction could increment the number of $\mathcal{Y}$-variables by trying to remove constants. The trick is to define for every expression an $\mathcal{X}$-reduction (r, σ) such that the size of $\text{dom}(\sigma)$ depends only on N and $|\mathcal{X}|$.

We define an $\mathcal{X}$-reduction by induction over expressions by using a function $\text{red}_{\mathcal{X}}(\cdot)$. For this definition we assume that $\mathcal{Y}$ is possibly infinite. Later, in Lemma 9, we show that we only need a finite number of variables to define $\text{red}_{\mathcal{X}}(\cdot)$. For the base case, if $e = x \in \mathcal{X}$, then we define $\text{red}_{\mathcal{X}}(e) = (x, \sigma_\emptyset)$ where $\sigma_\emptyset$ is the empty function. Otherwise, if $e = s \in \mathcal{S}$ or $e = y \in \mathcal{Y}$ then we choose a fresh variable $y' \in \mathcal{Y}$ and define $\text{red}_{\mathcal{X}}(e) = (y', \sigma)$, where $\sigma(y') = e$ and $\text{dom}(\sigma) = \{y'\}$. For the inductive step, suppose that e is of the form:

$$e = \otimes\{e_1, \ldots, e_n, f_1, \ldots, f_m\} \tag{26}$$

where each $e_i \in \text{Expr}(\mathcal{X} \cup \mathcal{Y})$ contains at least one variable in $\mathcal{X}$ and each $f_i \in \text{Expr}(\mathcal{Y})$ contains no variables in $\mathcal{X}$. Furthermore, suppose that $\text{red}_{\mathcal{X}}(e_1) = (r_1, \sigma_1), \ldots, \text{red}_{\mathcal{X}}(e_n) = (r_n, \sigma_n)$ are already defined and $\text{dom}(\sigma_i) \cap \text{dom}(\sigma_j) = \emptyset$ (without lost of generality, we can relabel the variables). Then we define $\text{red}_{\mathcal{X}}(e) = (r, \sigma)$ recursively as follows:

$$\begin{aligned} r &= \otimes\{r_1, \ldots, r_n, y\} \\ \sigma &= \big(\sigma_1 \cup \ldots \cup \sigma_n\big)[y \to \otimes\{f_1, \ldots, f_m\}] \end{aligned} \tag{27}$$

where $y \in \mathcal{Y}$ is a fresh variable not used in $\sigma_1, \ldots, \sigma_n$ and $\big(\sigma_1 \cup \ldots \cup \sigma_n\big)$. If $m = 0$, then we do not add the additional variable y.

In the recursive definition of $\text{red}_{\mathcal{X}}(e)$, the subexpression $f_1, \ldots, f_m$ that do not use $\mathcal{X}$ registers are replaced by a new fresh variable y and its content $\otimes\{f_1, \ldots, f_m\}$ is assigned into y. It is clear from the definition that (r, σ) is an $\mathcal{X}$-reduction for e. Since $\sigma_1, \ldots, \sigma_n$ have disjoint domains and y was chosen as a fresh variable then r is copyless and the domain of σ is equal to $\bigcup_i \text{dom}(\sigma_i) \cup \{y\} = \text{Var}(r) \cap \mathcal{Y}$. For example, recall the expression $e = \oplus\{\odot\{x, y, 2\}, 3, \odot\{z, 4\}\}$ in (24). Suppose that $x \in \mathcal{X}$ and $y, z \in \mathcal{Y}$. Then the $\mathcal{X}$-reduction $\text{red}_{\mathcal{X}}(e) = (r, \sigma)$ is equal to:

$$r := \oplus\big\{ \odot\{x, u\},\ v \big\} \qquad \sigma : \begin{array}{l} u := y \odot 2 \\ v := 3 \oplus (z \odot 4) \end{array}$$

where u and v are fresh variables in $\mathcal{Y}$. One can easily check that $e = \hat{\sigma}(r)$ and, thus, the expression defined by e is preserved in (r, σ). In the next lemma, we show that we need at most $|\mathcal{X}| \cdot N + 1$ fresh variables from $\mathcal{Y}$ for the above definition of $\text{red}_{\mathcal{X}}(\cdot)$.

Lemma 9. *Let $e \in \text{Expr}(\mathcal{X} \cup \mathcal{Y})$ be copyless and with alternation bounded by N. If $\text{red}_{\mathcal{X}}(e) = (r, \sigma)$, then r is copyless with alternation bounded by N, σ is copyless with respect to $\text{dom}(\sigma)$, and $|\text{Var}(r) \cap \mathcal{Y}| \leq |\mathcal{X}| \cdot N + 1$.*

Proof. Suppose that $\text{red}_{\mathcal{X}}(e) = (r, \sigma)$. By definition it is straightforward to check that r is copyless and has alternation bounded by N. In fact, each time that a subexpression is replaced, we use a new fresh variable and since e is copyless this proves that r and σ are also copyless. The most interesting part is to bound the number of $\mathcal{Y}$-variables in r. For this, we prove a slightly stronger bound: $|\mathsf{Var}(r) \cap \mathcal{Y}| \leq |\mathsf{Var}(r) \cap \mathcal{X}| \cdot N + 1$. Since $|\mathsf{Var}(r) \cap \mathcal{X}| \leq |\mathcal{X}|$ this will prove the lemma.

The proof goes by induction over the alternation of e. For the base case we have $|\mathsf{Var}(r)| \leq 1$ thus the bound is trivially true. For the inductive step suppose that the statement holds for expression with alternation at most N and consider an expression e like (26) with alternation $N+1$. By definition of $\mathsf{Expr}^{\oplus}(\mathcal{X})$ and $\mathsf{Expr}^{\oplus}(\mathcal{X})$ each e_i has alternation at most N and the inductive hypothesis applies: $|\mathsf{Var}(r_i) \cap \mathcal{Y}| \leq |\mathsf{Var}(r_i) \cap \mathcal{X}| \cdot N + 1$ where $(r_i, \sigma_i) = \mathsf{red}_{\mathcal{X}}(e_i)$. Given that $\mathsf{Var}(r_i) \subseteq \mathsf{Var}(r)$

$$\sum_{i=1}^{n} |\mathsf{Var}(r_i) \cap \mathcal{Y}| \quad \leq \quad \sum_{i=1}^{n} (|\mathsf{Var}(r_i) \cap \mathcal{X}| \cdot N + 1) \quad \leq \quad |\mathsf{Var}(r) \cap \mathcal{X}| \cdot N + n.$$

Also, the set $\mathsf{Var}(r) \cap \mathcal{Y}$ is partitioned by the sets $\mathsf{Var}(r_i) \cap \mathcal{Y}$ and the fresh variable y. Therefore, we derive the following bound:

$$|\mathsf{Var}(r) \cap \mathcal{Y}| \quad \leq \quad \sum_{i=1}^{n} |\mathsf{Var}(r_i) \cap \mathcal{Y}| + 1 \quad \leq \quad |\mathsf{Var}(r) \cap \mathcal{X}| \cdot N + n + 1$$

Finally, given that r is copyless, we cannot have more subexpressions e_i than variables in $\mathcal{X}$ and, thus, $n \leq |\mathsf{Var}(r) \cap \mathcal{X}|$ which leads to the desire conclusion: $|\mathsf{Var}(r) \cap \mathcal{X}| \cdot N + n + 1 \leq |\mathsf{Var}(r) \cap \mathcal{X}| \cdot (N+1) + 1$. $\square$

We can naturally extend the function $\mathsf{red}_{\mathcal{X}}(\cdot)$ from $\mathsf{Expr}(\mathcal{X} \cup \mathcal{Y})$ to $\mathsf{Subs}(\mathcal{X}, \mathcal{Y})$. More precisely, for any $\alpha \in \mathsf{Subs}(\mathcal{X}, \mathcal{Y})$ define $\mathsf{red}_{\mathcal{X}}(\alpha) = (\beta, \sigma_\beta)$ such that $\mathsf{red}_{\mathcal{X}}(\alpha(x)) = (\beta(x), \sigma_x)$ for every $x \in \mathcal{X}$ and $\sigma_\beta = \bigcup_{x \in \mathcal{X}} \sigma_x$ (similar to $\mathcal{X}$-reduction of expressions we can assume that $\mathsf{dom}(\sigma_x) \cap \mathsf{dom}(\sigma_y) = \emptyset$ by relabeling σ_x and σ_y if necessary). Further, we extend $\mathsf{red}_{\mathcal{X}}(\cdot)$ from substitutions to substitution trees such that $\mathsf{red}_{\mathcal{X}}(t) = (t', \sigma_{t'})$ where $\mathsf{nodes}(t) = \mathsf{nodes}(t')$, $\mathsf{red}_{\mathcal{X}}(t(u)) = (t'(u), \sigma_u)$ for each $u \in \mathsf{nodes}(t)$, and $\sigma_{t'}$ is the disjoint union of all substitutions σ_u (again, we assume that domains of σ_u are disjoint). The following lemma, similar to Lemma 9 but for substitution trees, will be useful later in the correctness proof of δ'. We omit the proof since it is straightforward from Lemma 9 and the copyless restriction.

Lemma 10. *For any copyless substitution tree $t \in \mathsf{Trees}(\mathcal{X}, \mathcal{Y})$, if $\mathsf{red}_{\mathcal{X}}(t) = (t', \sigma_{t'})$, then t' is a copyless and constants-free substitution tree and $\sigma_{t'}$ is a copyless partial substitution. Furthermore, the number of $\mathcal{Y}$-variables used in t' is bounded by $|\mathsf{nodes}(t)| \cdot |\mathcal{X}| \cdot (|\mathcal{X}| \cdot N + 1)$.*

Recall that states of $\mathcal{A}'$ are of the form (t, B) where t is a complete, copyless, and constants-free substitution tree and $B : \mathsf{leaves}(t) \to Q$ is an injective function. To define the transition $\delta'(t, B) = ((t', B'), \sigma)$ we show how to convert t into t' and how to update B into B' through the composition of four different processes: extend, prune, shrink, and reduce. In the sequel, we explain each procedure in detail.

Extend. The first step is to extend branches in (t, B) to the next states when reading $a \in \Sigma$. We define this process formally by the function $\mathsf{extend}((t, B), a) = (t_1, B_1)$ that receives a state $(t, B) \in Q'$ and a letter $a \in \Sigma$ and outputs the pair (t_1, B_1), where t_1 is a substitution tree and $B_1 : \mathsf{leaves}(t_1) \rightharpoonup Q$ is a partial injective function. The substitution tree t_1 is defined as an extension of t (i.e., $\mathsf{nodes}(t) \subseteq \mathsf{nodes}(t_1)$) such that $t_1(u) = t(u)$ whenever $u \in \mathsf{nodes}(t)$ and for every $v \in \mathsf{leaves}(t)$, if there exists a transition $(B(v), a, q, \sigma) \in \delta$, then there exists $i \in \mathbb{N}$ such that $t_1(v \cdot i) = \sigma$. The function B_1 is defined only on the new leaves such that $B_1(v \cdot i) = q$. Intuitively, $\mathsf{extend}((t, B), a) = (t_1, B_1)$ extends t whenever the state on a leaf of t can evolve to a new state by reading a. Notice that δ is not deterministic and a leaf $v \in \mathsf{leaves}(t)$ could be extended with more than one node. Since trees are unordered, extend is a deterministic procedure.

Prune. The problem with (t_1, B_1) is that there could exist leaves in t_1 that are not marked by the function B_1 and therefore $(t_1, B_1) \notin Q'$. This happens when for a leaf v there is no transition $(B(v), a, q, \sigma)$ and this branch of the tree becomes a "dead run". The purpose of the function $\mathsf{prune}(t_1, B_1) = (t_2, B_2)$ is to prune branches that are dead and to update B_1 into a total function B_2. Formally, t_2 is a subset of t_1 (i.e., $\mathsf{nodes}(t_2) \subseteq \mathsf{nodes}(t_1)$ and $t_2(u) = t_1(u)$ for every $u \in \mathsf{nodes}(t_2)$) such that $v \in \mathsf{nodes}(t_2)$ iff $v \cdot u \in \mathsf{dom}(B_1)$ for some $u \in \mathbb{N}^*$. In other words, we keep only nodes that are ancestors of leaves that are marked by B_1. Finally, we define $B_2 = B_1$. Note that $\mathsf{dom}(B_2) = \mathsf{leaves}(t_2)$ since we did not remove any node from the domain of B_1. Moreover, paths from the root to leaves were not modified and, in particular, it holds that $t_2^{\downarrow}(u) = t_1^{\downarrow}(u)$ for every $u \in \mathsf{leaves}(t_2)$.

Shrink. By adding and removing branches with the procedures extend and prune it could happen that t_2 is not a complete tree and $(t_2, B_2) \notin Q'$ (i.e., t_2 could contain internal nodes with just one child). These nodes are redundant and they can be easily removed by shrinking the tree. For this purpose, we define the procedure $\mathsf{shrink}(t_2, B_2) = (t_3, B_3)$ recursively. We define shrink by induction on the depth of t_2, maintaining the following properties: t_3 is a complete tree; B_3 is an injective function from $\mathsf{leaves}(t_3)$ into Q; $\mathit{range}(B_3) = \mathit{range}(B_2)$ and that for every $u \in \mathsf{leaves}(t_2)$ there is $u' \in \mathsf{leaves}(t_3)$ such that $B_2(u) = B_3(u')$ and $t_2^{\downarrow}(u) = t_3^{\downarrow}(u')$. In particular this means that the size of t_3 is bounded by $2 \cdot |Q|$.

For trees that have only one node we define shrink as the identity function and the properties are kept trivially. Suppose $t_2 = \sigma\{r_1, \ldots, r_n\}$ for some $\sigma \in \mathsf{Subs}(\mathcal{X}, \mathcal{Y})$. For every $j \in \{1, \ldots, n\}$ let $i_j \in \mathbb{N}$ be the node in t_2 corresponding to the root of r_i. Then for every leaf $u \in \text{leaves}(r_j)$ the node $i_j \cdot u$ is a leaf in t_2. Moreover $\text{leaves}(t_2) = \bigcup_j \{i_j \cdot u \mid u \in \text{leaves}(r_j)\}$. We define the functions $C_j : \text{leaves}(r_j) \to Q$ by $C_j(u) = B_2(i_j \cdot u)$. Notice that $range(B_2) = \bigcup_j range(C_j)$.

If $n > 1$ then $\text{shrink}(t_2, B_2) = (\sigma\{r'_1, \ldots, r'_n\}, B_3)$, where $\text{shrink}(r_j, C_j) = (r'_j, C'_j)$ and $B_3(i_j \cdot u) = C'_j(u)$ for every j and $u \in \text{dom}(C_j)$. By induction the properties are kept in $\text{shrink}(r_j, C_j)$ for all j. Then it is easy to see that they are also kept for $\text{shrink}(t_2, B_2)$. The remaining case is for $n = 1$, for simplicity we skip the indexes and write $t_2 = \sigma\{r\}$ and $C : \text{leaves}(r) \to Q$. Let r' be a tree such that $\text{nodes}(r') = \text{nodes}(r)$, $r'(u) = r(u)$ for every node $u \neq \epsilon$ and $r'(\epsilon) = \sigma \circ r(\epsilon)$. Then we define $\text{shrink}(t_2, B_2) = \text{shrink}(r', C)$. That is, the edge between the root of $\sigma\{r\}$ and its unique child r is removed. By induction the properties are kept in the step from (r', C) to $(t_3, B_3) = \text{shrink}((r', C))$. Thus we only have to prove that $range(B_3) = range(B_2)$ and that for every $u \in \text{leaves}(t_2)$ there is $u' \in \text{leaves}(t_3)$ such that $t_2^{\downarrow}(u) = t_3^{\downarrow}(u')$. The first property follows from the fact that $range(B_2) = range(C) = range(B_3)$. To prove the second property let $u \in \text{leaves}(t_2)$. By definition there exists an i such that $u = i \cdot v$ and $B_2(i \cdot v) = C(v)$. Let $|u| = k$ then:

$$\begin{aligned}
t_2^{\downarrow}(u) &= t_2(\epsilon) \circ t_2(u[\cdot,1]) \circ t_2(u[\cdot,2]) \circ \ldots \circ t_2(u[\cdot,k]) \\
&= \underbrace{\sigma \circ t_2(u[\cdot,1])}_{r'(\epsilon)} \circ \underbrace{t_2(u[\cdot,2])}_{r'(v[\cdot,1])} \circ \ldots \circ \underbrace{t_2(u[\cdot,k])}_{r'(v[\cdot,k-1])} \\
&= r'^{\downarrow}(v).
\end{aligned}$$

By the induction assumption there is $u' \in \text{leaves}(t_3)$ such that $r'^{\downarrow}(v) = t_3^{\downarrow}(u')$.

Reduce. The pair (t_3, B_3) is almost ready after shrinking single-child internal nodes, except that substitutions inside t_3 could have constants (e.g. $\text{extend}(\cdot)$ could have introduced constants). To solve this issue, we apply the $\mathcal{X}$-reduction procedure $\text{red}_{\mathcal{X}}(\cdot)$ to reduce all substitutions in t_3. Formally, we define the procedure $\text{reduce}(t_3, B_3) = ((t_4, B_4), \sigma)$ where $\text{red}_{\mathcal{X}}(t_3) = (t_4, \sigma)$ and $B_3 = B_4$. The function $\text{red}_{\mathcal{X}}$ changes only the labels (i.e. substitutions of the nodes) and, in particular, it does not change its tree structure, namely, $\text{nodes}(t_3) = \text{nodes}(t_4)$. Thus $|\text{nodes}(t_4)| = |\text{nodes}(t_3)| \leq 2 \cdot |Q|$. By Lemma 10, this implies that the number of fresh $\mathcal{Y}$-variables needed to apply the procedure $\text{red}_{\mathcal{X}}(\cdot)$ is at most $|\text{nodes}(t_3)| \cdot |\mathcal{X}| \cdot (|\mathcal{X}| \cdot N + 1) = 2 \cdot |Q| \cdot |\mathcal{X}| \cdot (|\mathcal{X}| \cdot N + 1)$ which is exactly the size of $\mathcal{Y}$. The remaining issue is that the function $\text{red}_{\mathcal{X}}$ is not deterministic since it has to choose "fresh variables" to apply the $\mathcal{X}$-reduction. To make it deterministic we can fix a deterministic choice of every fresh variable inside $\text{red}_{\mathcal{X}}(\cdot)$ (e.g. by following an arbitrary total order over $\mathcal{Y}$). With this change the procedure reduce is deterministic.

With the definitions of the procedures extend, prune, shrink and reduce, we are ready to define formally the transition function δ' of $\mathcal{A}'$. Specifically, for every state $(t, B) \in Q'$ and every $a \in \Sigma^*$ we define:

$$\delta'((t, B), a) = \text{reduce}(\,\text{shrink}(\,\text{prune}(\,\text{extend}((t, B), a)\,)\,)\,) = ((t_4, B_4), \sigma)$$

Note that all procedures (reduce, shrink, prune, extend) are deterministic so δ' is also deterministic. In the next lemma, we show that the definition of δ' is correct.

Lemma 11. *For any $(t, B) \in Q'$ and $a \in \Sigma$ let:*

$$\text{reduce}(\,\text{shrink}(\,\text{prune}(\,\text{extend}((t, B), a)\,)\,)\,) = ((t_4, B_4), \sigma).$$

Then $(t_4, B_4) \in Q'$ and σ is a copyless substitution over $\mathcal{Y}$.

Proof. Assume that we have $(t_1, B_1) = \text{extend}((t, B), a)$, $(t_2, B_2) = \text{prune}(t_1, B_1)$, $(t_3, B_3) = \text{shrink}(t_2, B_2)$ and $(t_4, B_4) = \text{reduce}(t_3, B_3)$. We first check that t_4 is a complete, copyless, and constants-free substitution tree in $\text{Trees}(\mathcal{X}, \mathcal{Y})$. We start from showing that t_4 is complete. The tree t_3 is a result of $\text{prune}(t_2)$, which by definition is a complete tree. This proves completeness because t_4 has the same structure as t_3.

We show that t_4 is copyless and constant-free. The procedure $\text{extend}(\cdot)$ introduces new variables in t only from $\mathcal{X}$ in new nodes. We label the new nodes with substitutions σ from $\mathcal{A}$, which by definition are substitutions from $\text{Subs}(\mathcal{X}, \mathcal{Y})$. Moreover it does not introduce new variables from $\mathcal{Y}$ thus the tree t_1 is copyless. The procedures $\text{prune}(\cdot)$ and $\text{shrink}(\cdot)$ do not introduce new variables in t_1, and t_2. This shows that the tree t_3 is copyless. By Lemma 10 we conclude that t_4 is copyless, constants-free, and σ is a copyless substitution. □

Now we have all the ingredients to prove Lemma 8.

Proof of Lemma 8. The lemma is proved by induction over the size of $w \in \Sigma^*$. For the base case $w = \epsilon$ by definition $q'_0 = (t_0, B_0)$ where t_0 is a single-node tree labeled with $\sigma_0 \in \text{Subs}(\mathcal{X}, \mathcal{Y})$ and $B_0(\epsilon) = q_0$, where $\sigma_0(x) = \hat{x}$ for every $x \in \mathcal{X}$, where $\hat{x}$ is the copy of x in $\hat{\mathcal{X}}$. The initial function ν'_0 is defined as $\nu'_0(\hat{x}) = \nu_0(x)$ for every $\hat{x} \in \hat{\mathcal{X}}$ and $\nu'_0(y) = \mathbb{0}$ for every $y \in \mathcal{Y} \setminus \hat{\mathcal{X}}$. In this setting

$$\xi \circ t^{\downarrow}(u)(x) = \nu'_0 \circ \sigma_0(x) = \nu'_0(\hat{x}) = \nu_0(x).$$

In the initial configuration there is only one run which ends in q_0 and $range(B_0) = \{q_0\}$, which finishes the proof of the base case.

For the induction step, assume that the lemma holds for $w \in \Sigma^*$ and we show that it also holds for $w \cdot a$ for any $a \in \Sigma$. Let (q', ν') be a configuration reached by a run of $\mathcal{A}$ over $w \cdot a$. By Lemma 7, there is a unique run of $\mathcal{A}$ on w that ends in a configuration (q, ν) such that $(q, a, q', \sigma) \in \delta$ and $\nu' = \nu \circ \sigma$. By the induction assumption we have that there is a run of $\mathcal{A}'$ on w reaching configuration $((t, B), \xi)$ such that $Q(w) = range(B)$ and for every $x \in \mathcal{X}$ that there exists $u \in \text{nodes}(t)$ such that $B(u) = q$ and

$$\nu = \xi \circ t^{\downarrow}(u). \tag{28}$$

Fix $(t_1, B_1) = \text{extend}((t, B), a)$, $(t_2, B_2) = \text{prune}(t_1, B_1)$, $(t_3, B_3) = \text{shrink}(t_2, B_2)$ and $(t_4, B_4, \tau) = \text{reduce}(t_3, B_3)$. By definition of extend the set $range(B_1)$ is the set of all p such that $(B(v), a, p, \sigma_p) \in \delta$ for some σ_p and $v \in \text{leaves}(t)$. Since $Q(w) = range(B)$ then $Q(w \cdot a) = range(B_1)$. Clearly by definition of the procedures prune, shrink, reduce we have $range(B_1) = range(B_2) = range(B_3) = range(B_4)$. By the unambiguity of $\mathcal{A}$ we get that $|range(B_4) \cap F| = |Q(w \cdot a) \cap F| = 1$. Similarly to prove that B_4 is injective it suffices to show that B_1 is injective. Suppose contrary that B_1 maps two different leaves to the same state p. Then by definition of B_1 there are at least two runs of $\mathcal{A}$ on $w \cdot a$ that end in q, which contradicts Lemma 7. We conclude by Lemma 11 that $\delta'((t, B), a) = (t_4, B_4)$ and $\mathcal{A}'$ is in configuration $((t_4, B_4), \xi \circ \tau)$ after reading $w \cdot a$. Since $range(B_4) = Q(w \cdot a)$ then there exists $i \in \mathbb{N}$ such that $ui \in \text{leaves}(t_1)$, $B_1(u \cdot i) = q'$, and $t_1(ui) = \sigma$. If we compose both sides of (28) with σ then we get:

$$\underbrace{\nu \circ \sigma}_{\nu'} = \xi \circ \underbrace{t^{\downarrow}(u) \circ \sigma}_{t_1^{\downarrow}(u \cdot i)} \quad \text{and} \quad \nu' = \xi \circ t_1^{\downarrow}(u \cdot i).$$

We know that the procedure prune$(\cdot)$ and shrink$(\cdot)$ preserve the outputs of the collapse operation. Thus there exists $v \in \text{nodes}(t_3)$ such that $B_3(v) = B'(v) = q'$ and $\nu' = \xi \circ t_3^{\downarrow}(v)$. Given that $(t_4, B_4, \tau) = \text{reduce}(t_3, B_3)$ is an $\mathcal{X}$-reduction, we know that $\tau \circ t_4^{\downarrow}(v) = t_3^{\downarrow}(v)$. By replacing the last equation in the above formula, we get $\nu' = \xi \circ \tau \circ t_4^{\downarrow}(v)$, which concludes the induction step. □

Lemma 8 in particular proves that δ' is a total function when Q' is restricted to the reachable states Q'_r. For this reason we restrict the set of states to Q'_r. With Q'_r as the set of states the automaton $\mathcal{A}'$ is a deterministic CRA.

To conclude the proof, we show that $\mathcal{A}$ and $\mathcal{A}'$ have the same output on every word w. Let (q, ν) be the configuration of the unique accepting run of $\mathcal{A}$ on w and let $((t, B), \xi)$ be the configuration of $\mathcal{A}'$ of the run on w. By Lemma 8 there exists a node u such that $u \in \text{leaves}(t)$ such that $B(u) = q$ and $\nu = \xi \circ t^{\downarrow}(u)$. The output of $\mathcal{A}'$ on w is defined as $\xi \circ \mu'(t, B)$. By definition $q \in F$ and $|range(B) \cap F| = 1$ thus q is the unique accepting state in leaves(t). By definition $\mu'(t, B) = t^{\downarrow}(u) \circ \mu(q)$. Thus we get the following equalities for the output of $\mathcal{A}'$ on w:

$$\xi \circ \mu'(t, B) = \xi \circ t^{\downarrow}(u) \circ \mu(q) = \nu \circ \mu(q).$$

This finishes the proof given that $\nu \circ \mu(q)$ is exactly the output of $\mathcal{A}$ on w. □

6.2. 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 [11,12] and has been stated as a key property of a model for computing non-boolean functions [3,4]. 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 $\Delta : Q \times \text{REG}_\Sigma \rightharpoonup Q \times \text{Subs}(\mathcal{X})$ is a partial transition function with finite domain. 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 $1 \le i \le n$, $\Delta(q_{i-1}, L_i) = (q_i, \sigma_i)$, $a_i \dots a_n \in L_i$ 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 $L_1, \dots, L_k \in \text{REG}_\Sigma$. Then the languages $L_1, \dots, L_k$ are pairwise disjoint (i.e. $L_i \cap L_j = \emptyset$). 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.

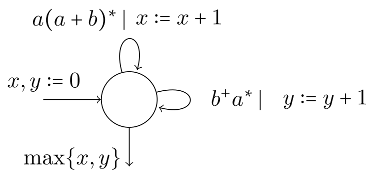


Fig. 6. BAC extended with regular look-ahead recognizing f from Example 5.

Example 5. Consider the function $f : \Sigma^* \to \mathbb{N}$ from Example 4. This function can be defined by a BAC extended with regular look-ahead in Fig. 6. The automaton has two registers x and y keeping the number of all a's and the number of b's in the last block, respectively. When updating register y, to verify if the automaton is in the last block of b's it checks whether the suffix belongs to b^+a^*. For simplicity, the transition $b^+a^+b(a+b)^*$, which does not change the value of the registers, is omitted.

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 4. *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. Let $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \Delta, q_0, \nu_0, \mu)$ be a BAC with regular look-ahead where $\Delta : Q \times \mathrm{REG}_\Sigma \rightharpoonup Q \times \mathrm{Subs}(\mathcal{X})$ is the partial transition function (recall that Δ has finite domain). To represent Δ finitely, let $L_1, \ldots, L_N$ be all the regular languages in the finite domain of Δ and, for each $i \le N$, let $\mathcal{A}_i = (P_i, \Sigma, \delta_i, p_i^0, F_i)$ be a finite state automaton such that $L_i = \mathcal{L}(\mathcal{A}_i)$. To unify the structure of each $\mathcal{A}_i$, define the set of states $P = \biguplus_{i=1}^N P_i$, the transition function $\delta = \biguplus_{i=1}^N \delta_i$, and the set of final states $F = \biguplus_{i=1}^N, F_i$, that is, P, δ and F are the disjoint union of states, transitions and final states, respectively, of the automata $\mathcal{A}_1, \ldots, \mathcal{A}_N$. Then for every $i \le N$, define the automaton $\mathcal{R}_i = (P, \Sigma, \delta, p_i^0, F)$. By definition of P, δ and F, one can easily see that $L_i = \mathcal{L}(\mathcal{R}_i)$ for every $i \le N$.

We are ready to show an unambiguous BAC equivalent to $\mathcal{A}$. By Theorem 3 this will show that $\mathcal{A}$ can be defined by a deterministic BAC. Let $\mathcal{A}' = (Q', \Sigma, \mathcal{X}, \Delta', I_0', \nu_0, F', \mu')$ be a BAC such that:

- $Q' = Q \times 2^P$ is the set of states,
- $\Delta' \subseteq Q' \times \Sigma \times Q' \times \mathrm{Subs}(\mathcal{X})$ where $((q, S), a, (q', S'), \sigma) \in \Delta'$ iff there exist $(q, L_i, q', \sigma) \in \Delta$ for some $i \le N$, and a surjective function $f : S \cup \{p_i^0\} \to S'$ such that $\delta(s, a) = f(s)$ for every $s \in S \cup \{p_i^0\}$,
- $I_0' = \{(q_0, \emptyset)\}$,
- $F' = \{(q, S) \mid S \subseteq F\}$, and
- $\mu' : Q' \to \mathrm{Expr}(\mathcal{X})$ where $\mu'(q, S) = \mu(q)$ for every $(q, S) \in Q'$.

The idea behind $\mathcal{A}'$ is to guess, at each letter, all transitions in $\mathcal{A}$ that are satisfied by the remaining suffix. In each state (q, S) of $\mathcal{A}'$ we keep the current state q of a run and a subset S of P that includes all regular transitions that have been taken so far by the run on q. Then we have a transition $((q, S), a, (q', S'), \sigma) \in \Delta'$ if there exists a transition $(q, L_i, q', \sigma) \in \Delta$ (i.e. a transition from q to q') such that we can extend each state in $S \cup \{p_i^0\}$ to a state in S'. Note that, in order to start simulating the finite automaton $\mathcal{R}_i$ over the suffix, p_i^0 is also included on the set of states that are updated. Finally, if the last state (q, S) of a run satisfies $S \subseteq F$, then we know that all suffixes during the run satisfies the regular look-ahead of the transitions and, then, the state (q, S) is final.

We show first that $\mathcal{A}'$ is unambiguous. By contradiction, suppose that $\mathcal{A}'$ is not unambiguous, that is, there exist $w = a_1 \ldots a_n \in \Sigma^*$ and two different accepting runs ρ and ρ' of $\mathcal{A}'$ over w. Let $i \le n$ be the least position such that $\rho(i) = \rho'(i) = (q, S)$ but $\rho(i+1) = (q_1, S_1) \ne (q_2, S_2) = \rho'(i+1)$. We know that this position exists since, by construction, it holds that $\rho(0) = \rho'(0)$. Let $(q, L, q_1, \sigma_1) \in \Delta$ and $(q, L', q_2, \sigma_2) \in \Delta$ be the transitions that witness the transitions $((q, S), a_{i+1}, (q_1, S_1), \sigma_1) \in \Delta'$ and $((q, S), a_{i+1}, (q_2, S_2), \sigma_2) \in \Delta'$. Since both runs are accepting, then it is straightforward to show by induction that $w[i, \cdot] \in L$ and $w[i, \cdot] \in L'$ for the suffix $w[i, \cdot]$ of w at position i. Then we have a contradiction since, by definition of CRA-RLA, we know that $L \cap L' = \emptyset$. We conclude that $\mathcal{A}'$ must be unambiguous. Note that during the construction we did not change the assignments σ in the transitions. For this reason, we can also conclude that $\mathcal{A}'$ is a copyless CRA with bounded alternation.

For the last part of the proof, we have to show that $[\![\mathcal{A}]\!](w) = [\![\mathcal{A}']\!](w)$ for every $w \in \Sigma^*$. It is easy to verify that for every run $(q_0, \nu_0) \xrightarrow{L_1} \ldots \xrightarrow{L_n} (q_n, \nu_n)$ of $\mathcal{A}$ over $w \in \Sigma^*$, there exist sets S_i such that the sequence $((q_0, S_0), \nu_0) \xrightarrow{a_1} \ldots \xrightarrow{a_n} ((q_n, S_n), \nu_n)$ is an accepting run of $\mathcal{A}'$ over w. For a given word w the sets S_i are uniquely determined by the transitions Δ and δ. Thus, $[\![\mathcal{A}]\!](w) = [\![\hat{\nu}_n(\mu(q_n))]\!] = [\![\mathcal{A}']\!](w)$ for every $w \in \Sigma^*$. $\square$

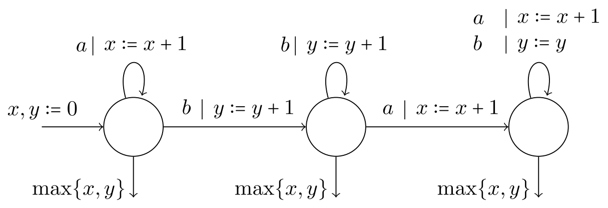


Fig. 7. BAC recognizing f^r from Example 6.

6.3. Closure under reverse

We finish this section proving that, in contrast to copyless CRA (see Section 5), 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 $w \in \Sigma^*$. First, we show an example that sometimes it is more convenient to define the reverse of the function.

Example 6. Consider the function $f : \Sigma^* \to \mathbb{N}$ from Example 4. We define the function f^r by a BAC in Fig. 7. The automaton has two registers x and y keeping the number of all a's and the number of b's in the last block, respectively. By adding extra two states the automaton updates the register y only in the first block of b's, and then it keeps its value.

Theorem 5. *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 $\mathcal{A}$.*

Proof. Let $\mathcal{A} = (Q, \Sigma, \mathcal{X}, \Delta, q_0, \nu_0, \mu)$ be a (deterministic) BAC with alternation bounded by N. The main idea of this construction is to run $\mathcal{A}$ backwardly over an input. For this purpose, we start from any state in Q where $\mathcal{A}$ could have potentially finished and follow the transitions in the other direction. Of course, by starting from any state and going back there are many non-deterministic choices that the reverse automaton must take to find the run of $\mathcal{A}$ over the input. Fortunately, $\mathcal{A}$ is a deterministic automaton which implies that there is at most one run that starts in q_0 and ends in some state in Q. That is, the reverse automaton constructed from $\mathcal{A}$ will be unambiguous and, by Theorem 3, we know that there is an equivalent deterministic BAC cost-register automaton. The only remaining issue is to construct the output expression of $\mathcal{A}$ over the input from the back. For this, we exploit the bounded alternation of $\mathcal{A}$ and $\mathcal{X}$-reductions (see below) to extend and reduce expressions starting from the final output.

Let $\mathcal{Y}$ be a non-empty set of variables such that $\mathcal{X} \cap \mathcal{Y} = \emptyset$. We recall here the idea of an $\mathcal{X}$-reduction introduced in the proof of Theorem 3. An $\mathcal{X}$-reduction for $e \in \mathrm{Expr}(\mathcal{X} \cup \mathcal{Y})$ is a pair $\mathrm{red}_{\mathcal{X}}(e) = (r, \sigma)$ where $r \in \mathrm{Expr}(\mathcal{X} \cup \mathcal{Y})$ is an expression without constants and $\sigma : \mathcal{Y} \to \mathrm{Expr}(\mathcal{Y})$ is a substitution such that $e = \hat{\sigma}(r)$. Moreover, r is copyless whenever e is copyless. By Lemma 9, we know that if e has alternation bounded by N, then r has alternation bounded by N, σ is a copyless substitution, and the number of $\mathcal{Y}$-variables in r is bounded by $|\mathcal{X}| \cdot N + 1$.

We have now all the ingredients to define a reverse BAC cost register automaton from $\mathcal{A}$. Formally, we construct an unambiguous BAC $\mathcal{A}' = (Q', \Sigma, \mathcal{Y}, \Delta', I'_0, V'_0, F', \mu')$ from $\mathcal{A}$ as follows.

- Q' is the set of all pairs (q, r) where $q \in Q$ and $r \in \mathrm{Expr}(\mathcal{X} \cup \mathcal{Y})$ is a copyless and constants-free expression, in particular, the size of r is bounded by $|\mathcal{X}| + |\mathcal{Y}|$.
- $\mathcal{Y}$ is a set of registers such that $\mathcal{X} \cap \mathcal{Y} = \emptyset$ and $|\mathcal{Y}| = |\mathcal{X}| \cdot N + 1$.
- I'_0 is the set of all pairs (q, r) such that $\mathrm{red}_{\mathcal{X}}(\mu(q)) = (r, \sigma)$ for some substitution σ.
- $V'_0 : I'_0 \to \mathrm{Val}(\mathcal{Y})$ is the initial substitution function such that $V'_0((q, r)) = \sigma$ for every $(q, r) \in I'_0$ and $\mathrm{red}_{\mathcal{X}}(\mu(q)) = (r, \sigma)$.
- F' is the set of all pairs $(q_0, r) \in Q'$ such that q_0 is the initial state of $\mathcal{A}$.
- μ' is the final substitution such that $\mu'((q, r)) = \nu_0 \circ r$ for every $(q, r) \in Q'$.
- Δ' is the transition relation where $((q, r), a, (q', r'), \sigma') \in \Delta'$ if, and only if, $\delta(q', a) = (q, \sigma)$ and $\mathrm{red}_{\mathcal{X}}(\sigma \circ r) = (r', \sigma')$.

We start by explaining the definition of the initial substitution function V'_0 and the transition relation Δ'. First, note that V'_0 is well defined, namely, $V'_0((q, r)) = \sigma$ is a valuation over $\mathcal{Y}$ for every $(q, r) \in I_0$ satisfying $\mathrm{red}_{\mathcal{X}}(\mu(q)) = (r, \sigma)$. Indeed, the expressions $\mu(q)$ contains $\mathcal{X}$-variables and constants which implies that σ is a substitution over $\mathcal{Y}$, where $\sigma(y)$ is a ground expression for every $y \in \mathrm{dom}(\sigma)$. We consider $\sigma(y)$ as a constant because as an expression without variables it can be evaluated into a constant. Regarding the transition relation Δ', notice that for $((q, r), a, (q', r'), \sigma') \in \Delta'$ we are following the transition $\delta(q', a) = (q, \sigma)$ backwardly and "extending and reducing" r by applying σ. This is similar to the extend function used in the transition function of Theorem 3. The result of this "extension" is reduced by the $\mathrm{red}_{\mathcal{X}}$ procedure into an expression that has bounded size by Lemma 9. In other words, we are constructing the output of $\mathcal{A}$ over the input backwardly by extending the final output $\mu(q)$, reducing the tree whenever it is possible and storing this in the states. Finally, one can easily show that $\mathcal{A}'$ is unambiguous given that for each transition $((q, r), a, (q', r'), \sigma') \in \Delta'$ we are following the transition $\delta(q', a) = (q, \sigma)$ backwardly and given (q, t), a, and q' one can easily check by definition that r' and σ' are uniquely determined.

It is left to show that for every word $w = a_1 a_2 \ldots a_n \in \Sigma^*$ we have $[\![\mathcal{A}]\!](a_1 a_2 \ldots a_n) = [\![\mathcal{A}']\!](a_n a_{n-1} \ldots a_1)$. Let $\rho : (q_0, \nu_0) \xrightarrow{a_1} \ldots \xrightarrow{a_n} (q_n, \nu_n)$ be a run of $\mathcal{A}$ over w 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 $x \in \mathcal{X}$. Recall that the output of $\mathcal{A}$ over w is defined by $[\![\mathcal{A}]\!](w) = [\![\nu_n \circ \mu(q_n)]\!]$. By the construction of $\mathcal{A}'$ and Δ', let

$$\rho' : ((q_n, r_n), \chi_n) \xrightarrow{a_n} ((q_{n-1}, r_{n-1}), \chi_{n-1}) \xrightarrow{a_{n-1}} \ldots \xrightarrow{a_1} ((q_0, r_0), \chi_0)$$

be a run of $\mathcal{A}'$ over $a_n a_{n-1} \ldots a_1$ such that $\text{red}_{\mathcal{X}}(\mu(q_n)) = (r_n, \chi_n)$ and, for every $1 \le i \le n$, it holds that $((q_i, r_i), a_i, (q_{i-1}, r_{i-1}), \tau_i) \in \Delta'$ where $\text{red}_{\mathcal{X}}(\sigma_i \circ r_i) = (r_{i-1}, \tau_i)$ and $\chi_{i-1}(y) = [\![\chi_i \circ \tau_i(y)]\!]$ for each $y \in \mathcal{Y}$. One can easily check that ρ' is the only run of $\mathcal{A}'$ over $a_n a_{n-1} \ldots a_1$. Indeed, r_n is determined by q_n and each pair (q_{i-1}, r_{i-1}) is determined by q_i, r_i and ρ. Therefore, it is left to show that the output of ρ' is equal to the output of ρ. For this, we prove by backward induction (i.e. starting from n and ending in 0) that:

$$[\![\nu_i \circ \chi_i \circ r_i]\!] = [\![\mathcal{A}]\!](w[1, i]). \tag{29}$$

This concludes the proof, since ν_i and χ_i are valuations over disjoint sets of variables

$$[\![\mathcal{A}]\!](w) = [\![\nu_0 \circ \chi_0 \circ r_0]\!] = [\![\chi_0 \circ \nu_0 \circ r_0]\!] = [\![\chi_0 \circ \mu'((q_0, r_0))]\!] = [\![\mathcal{A}']\!](a_n a_{n-1} \ldots a_1).$$

To prove the base of induction in (29) notice that $[\![\mathcal{A}]\!](w) = [\![\nu_n \circ \mu(q_n)]\!] = [\![\nu_n \circ \chi_n \circ r_n]\!]$, since by definition $\chi_n \circ r_n = \mu(q_n)$. For the inductive step suppose that (29) holds for $i+1$ and we prove it for i. By definition of Δ' we have $\text{red}_{\mathcal{X}}(\sigma_{i+1} \circ r_{i+1}) = (r_i, \tau_{i+1})$, furthermore

$$\begin{aligned}
\tau_{i+1} \circ r_i &= \sigma_{i+1} \circ r_{i+1} && \text{(by definition of } \mathcal{X}\text{-reduction)} \\
\nu_i \circ \tau_{i+1} \circ r_i &= \nu_i \circ \sigma_{i+1} \circ r_{i+1} && \\
\nu_i \circ \tau_{i+1} \circ r_i &= \nu_{i+1} \circ r_{i+1} && \text{(by definition of } \nu_i\text{)} \\
\chi_{i+1} \circ \nu_i \circ \tau_{i+1} \circ r_i &= \chi_{i+1} \circ \nu_{i+1} \circ r_{i+1} && \\
\nu_i \circ \chi_i \circ r_i &= \nu_{i+1} \circ \chi_{i+1} \circ r_{i+1} && \text{(by definition of } \chi_i\text{)} \\
\nu_i \circ \chi_i \circ r_i &= [\![\mathcal{A}]\!](w) && \text{(by inductive hyphotesis)} \quad \square
\end{aligned}$$

Notice that, as proved in Theorem 2, this construction fails for the function $f_{\mathcal{B}}^R$, where $f_{\mathcal{B}}$ is given by the copyless CRA $\mathcal{B}$. This is because while $\mathcal{B}$ is copyless its alternation is not bounded. In the above construction when the output is read backwards we encode in the states the tree expression obtained by composing the substitutions from backwards. In the constructions we ensure that one can store such a tree in a succinct way by using the $\mathcal{X}$-reductions. The $\mathcal{X}$-reductions deal with constants and registers that are on the same 'level' of alternation but it does not lower its alternation level. Since we do not have any reductions to drop the alternation of the tree, keeping such trees would require unbounded memory.

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.

7. 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. Due to this result we proved that copyless CRA are strictly contained in the class of weighted automata. In [4] Alur et al. introduced the class of regular cost functions defined in terms of streaming string-to-tree transducers. 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, we can conclude that copyless CRA are strictly contained in the class of regular cost functions.

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. An open problem here is to show whether these constructions are optimal or not. In particular, the construction to show that BAC are closed under regular look-ahead is double exponential: first constructing an unambiguous BAC; and second by determinizing it. Each step takes exponential time, and one can envision a construction in one step that requires only a single exponential. 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 4.

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 Corentin Barloy for repairing an error in Lemma 5. 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 grant 821320001, FONDECYT grant 11150653, and partially supported by Millennium Institute for Foundational Research on Data (IMFD).

References

[1] S. Almagor, U. Boker, O. Kupferman, What's decidable about weighted automata?, in: ATVA, 2011, pp. 482–491.
[2] R. Alur, P. Cerný, Streaming transducers for algorithmic verification of single-pass list-processing programs, in: POPL, 2011, pp. 599–610.
[3] R. Alur, L. D'Antoni, Streaming tree transducers, in: Automata, Languages, and Programming, Springer, 2012, pp. 42–53.
[4] R. Alur, L. D'Antoni, J. Deshmukh, M. Raghothaman, Y. Yuan, Regular functions and cost register automata, in: LICS, IEEE Computer Society, 2013, pp. 13–22.
[5] R. Alur, L. D'Antoni, J.V. Deshmukh, M. Raghothaman, Y. Yuan, Regular functions, cost register automata, and generalized min-cost problems, CoRR, 2011, arXiv:1111.0670.
[6] R. Alur, M. Raghothaman, Decision problems for additive regular functions, in: Automata, Languages, and Programming, Springer, 2013, pp. 37–48.
[7] J.R. Buchi, Weak second-order arithmetic and finite automata, Math. Log. Q. 6 (1–6) (1960) 66–92.
[8] K. Culik II, J. Kari, Image compression using weighted finite automata, in: Mathematical Foundations of Computer Science 1993, Springer, 1993, pp. 392–402.
[9] M. Droste, P. Gastin, Weighted automata and weighted logics, Theor. Comput. Sci. 380 (1–2) (2007) 69–86.
[10] M. Droste, W. Kuich, H. Vogler, Handbook of Weighted Automata, 1st edition, Springer, 2009.
[11] J. Engelfriet, Top-down tree transducers with regular look-ahead, Math. Syst. Theory 10 (1) (1976) 289–303.
[12] J. Engelfriet, S. Maneth, Macro tree transducers, attribute grammars, and mso definable tree translations, Inf. Comput. 154 (1) (1999) 34–91.
[13] R.L. Graham, D.E. Knuth, O. Patashnik, Concrete Mathematics – A Foundation for Computer Science, 2nd ed., Addison-Wesley, 1994.
[14] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages and Computation, Addison-Wesley, 1979.
[15] S. Kreutzer, C. Riveros, Quantitative monadic second-order logic, in: LICS, IEEE Computer Society, 2013, pp. 113–122.
[16] D. Krob, The equality problem for rational series with multiplicities in the tropical semiring is undecidable, in: ICALP, 1992, pp. 101–112.
[17] S. Lombardy, Sequentialization and unambiguity of (max,+) rational series over one letter, in: G. Stéphane, L. Jean-Jacques (Eds.), Workshop on Max-Plus Algebras and Their Applications to Discrete-Event Systems, Theoretical Computer Science, and Optimization, Prague, Czech Republic, IFAC, Elsevier Sciences, 2001, p. 6.
[18] F. Mazowiecki, C. Riveros, Maximal partition logic: towards a logical characterization of copyless cost register automata, in: CSL, 2015.
[19] F. Mazowiecki, C. Riveros, Copyless cost-register automata: structure, expressiveness, and closure properties, in: 33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17–20, 2016, Orléans, France, 2016, pp. 53:1–53:13.
[20] M. Mohri, Finite-state transducers in language and speech processing, Comput. Linguist. 23 (2) (1997) 269–311.
[21] C.H. Papadimitriou, Computational Complexity, Addison-Wesley, 1993.
[22] M.O. Rabin, D. Scott, Finite automata and their decision problems, IBM J. Res. Dev. 3 (2) (Apr. 1959) 114–125.
[23] S. Safra, On the complexity of omega-automata, in: FOCS, 1988, pp. 319–327.
[24] J. Sakarovitch, Elements of Automata Theory, Cambridge University Press, 2009.
[25] M.P. Schützenberger, On the definition of a family of automata, Inf. Control 4 (1961) 245–270.
[26] A. Weber, H. Seidl, On the degree of ambiguity of finite automata, Theor. Comput. Sci. 88 (2) (1991) 325–349.