Register complexity and determinisation of max-plus automata

Laure Daviaud

Abstract

We survey some results about the sequentiality problem for max-plus automata and its generalisation, the register complexity problem for cost register automata. We compare classes of functions computed by max-plus automata and by cost register automata with respect to the notion of ambiguity. The two models are introduced gently, so the novice reader is welcome!

1 Introduction

Weighted automata (introduced in [Sch61]) are a well-studied quantitative generalisation of finite non deterministic automata, computing functions which map words to values. This allows to model the computations of probabilities, costs, running time... Each transition carries a weight and, on reading a word, these weights are combined using two operations: one to aggregate the weights along a given run; one to put together computations of the runs labelled by a given word. Max-plus automata are a special case of weighted automata where these two operations are addition and maximum.

They have been applied in natural language processing [Moh97, MPR02], automatic analysis of complexity of programs [CDZ14], study of discrete event systems [GM99], termination of rewriting systems [Wal07] and to prove results in tropical algebra [DGM17] and automata theory [Has88, Sim78, Sim88].

Unlike for non determinisitc finite automata, given a max-plus automaton, there does not necessarily exist an equivalent deterministic one, i.e. one computing the same function. The sequentiality problem asks whether, given a max-plus automaton, there exist an equivalent deterministic one. Whether the sequentiality problem is decidable or not is a prominent long-standing open problem, solved only in particular cases (see Section 2.3).

While this paper focuses on max-plus automata on words, they have also been studied on trees [BR82, Sei94, Kui97] and in particular, generalisations of some of the results given in this paper can be found in [Pau17, Pau19].

More recently, Alur et al. introduced an equivalent model: the cost register automata [ADD$^+$13, AR13]. A cost register automaton is a deterministic machine which computes a function mapping words to values. A finite set of

so-called registers are used to store values and operations can be performed on these values (in our case, maxima and sums). The register complexity of a cost register automaton computing a function f is the minimum number of registers that are necessary to compute f. Computing the register complexity of any cost register automaton is a highly challenging open problem.

2 Max-plus automata and ambiguity

In this section, we present the model of max-plus automata and the notions of ambiguity and sequentiality.

Notation An alphabet Σ is a finite set of symbols, called letters and Σ^* denotes the set of (possibly empty) finite words over Σ. The length of a word w is denoted by $|w|$. The set $\mathbb{N} \cup \{-\infty\}$ is denoted by $\mathbb{N}_{\max}$. The operations maximum and addition are extended to $\mathbb{N}_{\max}$ as usual.

We chose to use the set of natural numbers $\mathbb{N}$ as a base set in this paper and to present max-plus automata. However, we could have considered $\mathbb{Z}$ or $\mathbb{Q}$ and min-plus automata and presented exactly the same results. See Remark 1 for more details.

2.1 Max-plus automata

Before giving the formal definition of a max-plus automaton, we present a simple example in Figure 1. A max-plus automaton is an automaton whose transitions carry weights. In Figure 1, the automaton has two states p and q. Initial states are denoted by ingoing arrows and final states by outgoing arrows. Both p and q are initial and final. Transitions between states are labelled by a letter (in this example, from the alphabet $\{a, b\}$) and a non negative integer, called the weight of the transition. Along a fixed path (or run), the weights on the transitions are added. The weight of a path is then defined as the sum of the weights of its transitions. The automaton is said to compute a value on every word: for a given word w, this is the maximum of the weights of all paths labelled by w going from an initial state to a final state. For example, in Figure 1, the word *aaabbaba* labels exactly two runs: one looping around p, which has weight $1+1+1+0+0+1+0+1 = 5$ and one looping around q which has weight $0+0+0+1+1+0+1+0 = 3$. The weight computed on w is then the maximum of 5 and 3, so 5. More generally speaking, the max-plus automaton represented in Figure 1 computes the maximum of the number of occurrences of the letter a and the number of occurrences of the letter b in a word. Indeed, every word w labels exactly two runs going from an initial state to a final state: one looping around p with weight the number of a's in w and one looping around q with weight the number of b's in w.

We give now the formal definition.

Definition 1 (Max-plus automata) *A max-plus automaton over an alphabet Σ is a tuple (Q, Q_I, Q_F, T) where:*

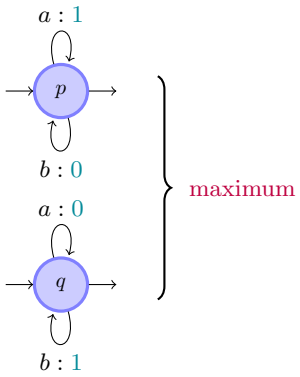


Figure 1: Max-plus automaton computing the maximum of the numbers of occurrences of the letters a and b

- Q *is a finite set of states,*
- $Q_I \subseteq Q$ *is a set of initial states,*
- $Q_F \subseteq Q$ *is a set of final states,*
- T *is a transition function* $Q \times \Sigma \times Q \to \mathbb{N}_{\max}$.

Given two states p and q and a letter a, whenever $T(p, a, q) \neq -\infty$, we say that (p, a, q) is a transition, $T(p, a, q)$ is its weight and we write:

$$p \xrightarrow{a:T(p,a,q)} q$$

A run on a word $w = w_1 w_2 \dots w_n$ where $w_i \in \Sigma$ for all $i = 1, \dots, n$, is a sequence of compatible transitions:

$$q_0 \xrightarrow{w_1:m_1} q_1 \xrightarrow{w_2:m_2} q_2 \xrightarrow{w_3:m_3} \cdots \xrightarrow{w_n:m_n} q_n$$

In this case, we can say that w labels the run. The weight of a run is the sum of the weights of the transitions in the run i.e. $\sum_{i=1}^{n} m_i$. The run is said to be accepting if $q_0 \in Q_I$ and $q_n \in Q_F$. The weight of a word w in the automaton is the maximum of the weights of the accepting runs labelled by w (with the usual convention that the maximum of the empty set is $-\infty$).

The *semantics* or *function computed* by a max-plus automaton $\mathcal{A}$ over alphabet Σ is the function which maps every word of Σ^* to its weight in $\mathbb{N}_{\max}$. It is denoted by $[\![\mathcal{A}]\!]$.

2.2 Sequentiality and Ambiguity

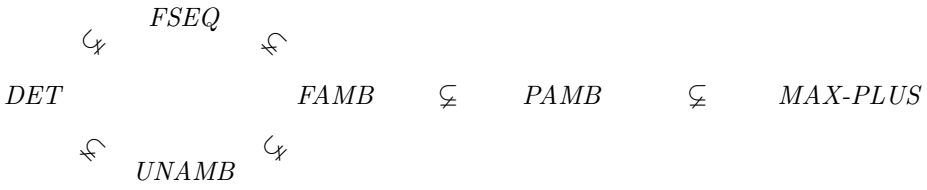
A max-plus automaton is said to be *deterministic* or *sequential* if for all states p and letters a, there is at most one state q such that $T(p, a, q) \neq -\infty$. It is said

to be *finitely sequential* if it is the union (maximum) of finitely many sequential max-plus automata.

It is said to be *k-ambiguous* for some positive integer k, if every word labels at most k accepting runs. It is said to be *unambiguous* if it is 1-ambiguous and *finitely ambiguous* if it is k-ambiguous for some k. Finally, it is said to be *polynomially ambiguous* if there is a polynomial P such that the number of accepting runs labelled by any word w is bounded by $P(|w|)$. Note that these notions do not depend on the weights on each transition, but only on the underlying automaton. They can be checked in polynomial time [BH77, WS91, AMR08].

The class of functions computed by a deterministic (resp. finitely sequential, unambiguous, finitely ambiguous, polynomially ambiguous, any) max-plus automaton is denoted by DET (resp. FSEQ, UNAMB, FAMB, PAMB, MAX-PLUS).

Proposition 1 *We have the following inclusions:*



Moreover, all the inclusions are strict and FSEQ and UNAMB are uncomparable.

Examples of max-plus automata in each class, proving the strict inclusions, can be found (or derived from examples) in [KLMP04, Kir08, MR18, CMMR20].

2.3 The sequentiality problem

Sequentiality problem

Input: A max-plus automaton $\mathcal{A}$ over an alphabet Σ.
Output: "Yes" if and only if there exist a max-plus automaton $\mathcal{B}$ over Σ such that:

1. $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$, and
2. $\mathcal{B}$ is deterministic.

In this case, $\mathcal{A}$ is said to be determinisable.

Theorem 1 ([KL09]) *The sequentiality problem is decidable for the class of polynomially ambiguous max-plus automata.*

This result emerged after a long history of research. To make the long story short, it was first proved for the class of unambiguous max-plus automata [Moh97] and finitely ambiguous max-plus automata [KLMP04]. The generalisation to the class of polynomially ambiguous automata is very intricate and demonstrated new deep insights in the problem.

The question whether the sequentiality problem is decidable for the whole class of max-plus automata is wide open.

Another linked problem is the one of finite sequentiality: given a max-plus automaton $\mathcal{A}$, is $[\![\mathcal{A}]\!]$ in FSEQ? This has been proved to be decidable for unambiguous [BK13] and finitely ambiguous max-plus automata [Bal13].

Remark 1 *As stated earlier, we have similar results when the weights are considered in $\mathbb{Z}$ or $\mathbb{Q}$ or when the minimum is used instead of the maximum operation (and $+\infty$ instead of $-\infty$). The ambiguity and sequentiality notions are not really affected by these choices (the situation is different when looking at problems such as equivalence and containment, for which one has to pay attention to the set in which the weights are considered).*

The reason for this is that there are transformations of max-plus automata which preserve ambiguity without "modifying the semantics too much".

From $\mathbb{N}$ to $\mathbb{Z}$ *Consider a max-plus automaton $\mathcal{A}$ with weights in $\mathbb{Z}$ and suppose that $-n$ is the smallest weight on a transition for some positive integer n. Consider the max-plus automaton $\overline{\mathcal{A}}$ constructed from $\mathcal{A}$ by adding n to all the weights on transitions of $\mathcal{A}$. Then $\overline{\mathcal{A}}$ has weights in $\mathbb{N}$, its ambiguity/sequentiality is the same as $\mathcal{A}$ and for all w, $[\![\overline{\mathcal{A}}]\!](w) = [\![\mathcal{A}]\!](w) + n|w|$.*

From $\mathbb{Z}$ to $\mathbb{Q}$ *Consider a max-plus automaton $\mathcal{A}$ with weights in $\mathbb{Q}$ and let n be a common denominator of all the weights on the transitions, with n in $\mathbb{N}$. Consider the max-plus automaton $\overline{\mathcal{A}}$ constructed from $\mathcal{A}$ by multiplying all the weights on the transitions of $\mathcal{A}$ by n. Then $\overline{\mathcal{A}}$ has weights in $\mathbb{Z}$, its ambiguity/sequentiality is the same as $\mathcal{A}$ and for all w, $[\![\overline{\mathcal{A}}]\!](w) = n[\![\mathcal{A}]\!](w)$.*

Results. *Using these transformations, one can easily prove that all the results stated in this section are also correct in $\mathbb{Z}$ and in $\mathbb{Q}$.*

Min and Max *Moreover, the sets $\mathbb{Z} \cup \{-\infty\}$ with operations* max *and* $+$ *and the set $\mathbb{Z} \cup \{+\infty\}$ with operations* min *and* $+$ *are isomorphic (taking the opposite). Using this observation and the transformations given above (from $\mathbb{N}$ to $\mathbb{Z}$ and conversely), the results in this section can be stated in terms of min-plus automata, i.e. automata defined with weights in $\mathbb{N}$ and substituting operation* max *by* min *and $-\infty$ by $+\infty$ in the definition of max-plus automata.*

3 Register complexity

In this section, we give an equivalent model to max-plus automata: cost register automata. Introduced in a more general setting by Alur et al. [ADD+13, AR13], they have been intensively studied and give models equivalent to weighted automata and transducers. We give here a definition which is not necessarily exactly the standard one (in particular regarding the final function) but is made to match the model of max-plus automata given above. More details about this can be found in Remark 2.

Notations For any positive integer ℓ, given two tuples $a = (a_1, a_2, \ldots, a_\ell)$ and $b = (b_1, b_2, \ldots, b_\ell)$ of elements in $\mathbb{N}_{\max}$, we denote by $\overline{(a, b)}$ the element of $\mathbb{N}_{\max}$ equal to $\max(a_1 + b_1, a_2 + b_2, \ldots, a_\ell + b_\ell)$.

Given a tuple $a = (a_1, a_2, \ldots, a_\ell)$, and a set of indices $S \subseteq \{1, \ldots, \ell\}$, we denote by $a_{|S}$ the set $\{a_i \mid i \in S\}$.

3.1 Cost register automata

We first introduce an example. Consider the picture in Figure 2. A cost register automaton is an automaton with a unique initial state (denoted by an ingoing arrow). The cost register automaton in Figure 2 has only one state which is the initial one. The automaton has a finite set of registers aimed at storing values in $\mathbb{N}_{\max}$ (initially 0). In the example, the automaton has two registers X_a and X_b. Each transition is labelled by a letter (in the example, from the alphabet $\{a, b\}$) and an update of the registers: the value in each register can be updated using the operation max and the current values in the registers added with some constants. For example in Figure 2, when reading an a, the value of the register X_a is updated by adding 1 to it and the value of the register X_b is unchanged. In this example, any update of the form $X_a := \max(X_a + c, X_b + d)$ for any c, d in $\mathbb{N}_{\max}$ would be a valid update for the register X_a; and similarly for X_b. For every letter and every state, only one transition labelled by the former can go out of the latter, making a computation in the automaton deterministic. Given a word w, the run on w starts from the initial state, and performs the updates of the registers on each transitions that are taken while reading w. Finally, each state has an output: a subset of the set of registers. In the example in Figure 2, the only state has the set of all the registers $\{X_a, X_b\}$ as output and the value computed is the maximum of the values contained in these registers. It is easy to see that in Figure 2, the register X_a stores the number of a's seen so far, and the register X_b the number of b's. As the automaton outputs the maximum of the values contained in X_a and X_b, this cost register automaton computes exactly the same function as the one computed by the max-plus automaton in Figure 1.

Let us now give the formal definition. Given a finite set of registers $\mathcal{X}$, an *update* on $\mathcal{X}$ is given by a function $\mathcal{X} \to \mathbb{N}_{\max}^{\mathcal{X}}$, i.e. a function mapping every register to a tuple of elements in $\mathbb{N}_{\max}$, one per each register. The set of updates on $\mathcal{X}$ are denoted by $\mathcal{UP}(\mathcal{X})$.

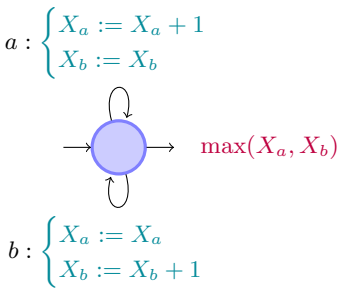


Figure 2: Cost register automaton computing the maximum of the numbers of occurrences of the letters a and b

Definition 2 (Cost register automata) *A cost register automaton over an alphabet Σ is a tuple $(Q, q_0, \mathcal{X}, \delta, \mu)$ where:*

- *Q is a finite set of states,*
- *$q_0 \in Q$ is the initial state,*
- *$\mathcal{X}$ is a finite set of registers,*
- *δ is a transition function $\delta : (Q \times \Sigma) \to (Q \times \mathcal{UP}(\mathcal{X}))$,*
- *μ is the final function which maps every state to a (possibly empty) set of registers.*

The semantics of such an automaton is given by means of valuations of the registers. From now on, let us write $\mathcal{X}$ as the set of ℓ registers $\{X_1, X_2, \ldots, X_\ell\}$ for some positive integer ℓ. A valuation of the registers is a tuple in $\mathbb{N}_{\max}^{\mathcal{X}}$. The initial valuation is $(0, 0, \ldots, 0)$. Update functions act on valuations as follows: given a valuation $v = (v_1, v_2, \ldots, v_\ell)$ and an update function f, we denote (extending notation) $f(v) = (\overline{(v, f(v_1))}, \overline{(v, f(v_2))}, \ldots, \overline{(v, f(v_\ell))})$ which is itself a valuation. A configuration is an element of $Q \times \mathbb{N}_{\max}^{\mathcal{X}}$. The initial configuration is the pair formed with the initial state and the initial valuation. A run on a word $w = w_1 w_2 \ldots w_n$ where $w_i \in \Sigma$ for all $i = 1, \ldots, n$ is a sequence of compatible configurations:

$$(q_0, v^0) \xrightarrow{w_1} (q_1, v^1) \xrightarrow{w_2} (q_2, v^2) \xrightarrow{w_3} \cdots \xrightarrow{w_n} (q_n, v^n)$$

such that (q_0, v^0) is the initial configuration, $\delta(q_{i-1}, w_i) = (q_i, f_i)$ and $v^i = f_i(v^{i-1})$ for all $i = 1, \ldots, n$.

Note that by definition of the model, every word labels exactly one run. We denote by (q_w, v^w) the uniquely defined configuration in which the run labelled by w ends. A cost register automaton $\mathcal{A}$ maps each word w to an element of $\mathbb{N}_{\max}$ defined by:

$$[\![\mathcal{A}]\!](w) = \max((v^w)_{|\mu(q_w)})$$

It is standard that max-plus automata and cost register automata are equivalent models [ADD$^+$13, Theorem 9]. More precisely, given a max-plus automaton $\mathcal{A}$ with n states, one can construct a cost register automaton $\mathcal{B}$ with n registers such that $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$. Conversely, given a cost register automaton $\mathcal{B}$ with n states and ℓ registers, one can construct a max-plus automaton $\mathcal{A}$ with $n\ell$ registers such that $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$.

3.2 Register complexity

Cost register automata are deterministic by definition. The challenge for this model is to minimise the number of registers that are needed to compute a function. The *register complexity* of a cost register automaton $\mathcal{A}$ is the least positive integer k such that there exists a cost register automaton $\mathcal{B}$ with k registers and $[\![\mathcal{B}]\!] = [\![\mathcal{A}]\!]$.

Register complexity minimisation

Input: A cost register automaton $\mathcal{A}$ over an alphabet Σ and an integer k.
Output: "yes" if and only if there exist a cost register automaton $\mathcal{B}$ over Σ such that:

1. $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$, and
2. $\mathcal{B}$ has at most k registers.

It is easy to see, using the constructions from [ADD$^+$13, Theorem 9], that a max-plus automaton is determinisable if and only if there exist a cost register automaton of register complexity 1 computing the same function, making the register complexity minimisation problem for cost register automata at least as difficult as the sequentiality problem for max-plus automata.

Proposition 2 *A max-plus automaton $\mathcal{A}$ is determinisable if and only if there exist a cost register automaton $\mathcal{B}$ of register complexity 1 such that $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$.*

Remark 2 *Alternative definitions of max-plus automata and cost register automata involve final weights: states in a max-plus automata are associated with weights ($-\infty$ if the state is not final) that are added to the weight of any run ending in the state. The equivalent adaptation for cost register automata is to modify the final function μ to allow adding non negative integers to the valuation of the registers in $\mu(q)$ for any state q (just before computing their maximum at the end of a run). Similar results as the ones presented in this paper hold. These modifications do not increase the expressive power of the models, and the class of functions computed whether allowing final weights or not is the same. However, the notion of sequentiality is impacted by these modifications: in order to transform an max-plus automaton with final weights into a max-plus automaton as in the definition presented in this paper, one has to "guess" the last letter of the word, thus introducing non determinism.*

3.3 Characterisation of the register complexity in a special case

As seen above, characterising the register complexity is a highly challenging problem. We can at least state one result that answers this question in a particular case.

Consider the class of cost register automata for which the only updates that are allowed are of the form $X := Y + c$ where X and Y are registers and c an element of $\mathbb{N}_{\max}$; and the final function μ maps every state to a single register. Essentially, we disallow the use of the operation max.

It can be proved that this class of cost register automata computes exactly the same functions as the class of unambiguous max-plus automata. The register complexity can be computed in this particular case [AR13], and is characterised by means of a twinning property [DRT16].

Theorem 2 ([AR13, DRT16]) *There is an algorithm which outputs the register complexity of any unambiguous max-plus automaton taken as input.*

4 Classes of max-plus automata and cost register automata

As already mentioned the class of deterministic max-plus automata and the class of cost register automata with 1 register compute exactly the same functions.

Similarly, other restrictions on cost register automata coincide with classes of max-plus automata.

An important notion, introduced in the study of cost register automata is the notion of *copyless*. Essentially, a cost register automaton is copyless if a register cannot be used twice in an update on a transition. We present here a particular case of copyless, that suits our framework, though this notion is usually more general.

A cost register automaton is said to be copyless if for all states p, q, letter a and update function f such that $\delta(p, a) = (q, f)$, for all registers X, Y and Z such that X is distinct from Y, if the Z-component of $f(X)$ is finite then the Z-component of $f(Y)$ is not.

Let $CRA^c(\max, +)$ be the class of functions computed by copyless cost register automata. Let $CRA^c(\max, +, := c)$ be the class of functions computed by copyless cost register automata with an additional update rule that is of the form $X := c$ for any non negative integer c. Let $CRA(+)_{\max}$ be the class of functions computed by cost register automata where the only update rules are of the form $X := Y + c$ and $CRA(+)$ the subclass of $CRA(+)_{\max}$ with the further restriction that $\mu(q)$ is a singleton for all states q.

Proposition 3 *We have the following equalities of classes:*

- $UNAMB = CRA(+)$

- $FSEQ = CRA^c(\max, +)$
- $FAMB = CRA(+)_{\max}$

We obtain the following (strict) hierarchy. The proofs of inclusions and equalities of classes are routine by adapting the constructions given in [ADD+13, Theorem 9] to go from max-plus automata to cost register automata and conversely; and using the fact that any function computed by a finitely ambiguous automaton is computed by the union (maximum) of unambiguous ones. Some examples justifying strictness are given below.

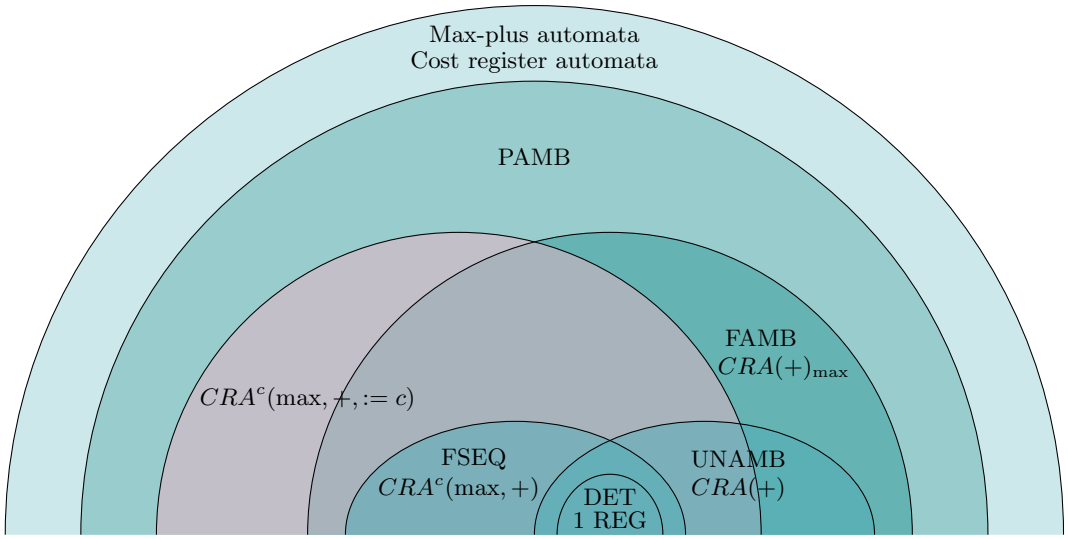


Figure 3: Classes of max-plus automata and cost register automata

Function in UNAMB, not in $CRA^c(\max, +, := c)$ Consider the alphabet $\{a, b, c\}$. Every word w can be written as $w_1 c w_2 c \cdots c w_n$ with $w_i \in \{a, b\}^*$. Let ℓ_i be the number of a's in w_i if the last letter of w_i is an a and the number of b's in w_i otherwise (if w_i's last letter is a b or w_i is empty). The function which maps w to the sum $\ell_1 + \ell_2 + \cdots + \ell_n$ is computed by an unambiguous max-plus automaton but is not in $CRA^c(\max, +, := c)$.

Function in $CRA^c(\max, +, := c)$, not in FAMB Consider the alphabet $\{a, b\}$. Every word w can be written as $a^{\ell_0} b a^{\ell_1} b \cdots b a^{\ell_n}$ for non negative integers $\ell_0, \ldots, \ell_n$. The function which maps w to $\max(\ell_0, \ldots, \ell_n)$ is in $CRA^c(\max, +, := c)$ but not computed by a finitely ambiguous max-plus automaton.

Other classes Other classes of cost register automata can be defined when relaxing/modifying the update rules and the notion of copyless. However, for now, there exist no characterisation of the polynomially ambiguous max-plus automata in terms of cost register automata.

References

[ADD+13] Rajeev Alur, Loris D'Antoni, Jyotirmoy V. Deshmukh, Mukund Raghothaman, and Yifei Yuan. Regular functions and cost register automata. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 13–22. IEEE Computer Society, 2013.

[AMR08] Cyril Allauzen, Mehryar Mohri, and Ashish Rastogi. General algorithms for testing the ambiguity of finite automata. In Masami Ito and Masafumi Toyama, editors, *Developments in Language Theory, 12th International Conference, DLT 2008, Kyoto, Japan, September 16-19, 2008. Proceedings*, volume 5257 of *Lecture Notes in Computer Science*, pages 108–120. Springer, 2008.

[AR13] Rajeev Alur and Mukund Raghothaman. Decision problems for additive regular functions. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, volume 7966 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2013.

[Bal13] Sebastian Bala. Which finitely ambiguous automata recognize finitely sequential functions? - (extended abstract). In Krishnendu Chatterjee and Jirí Sgall, editors, *Mathematical Foundations of Computer Science 2013 - 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings*, volume 8087 of *Lecture Notes in Computer Science*, pages 86–97. Springer, 2013.

[BH77] Meera Blattner and Tom Head. Automata that recognize intersections of free submonoids. *Information and Control*, 35(3):173–176, 1977.

[BK13] Sebastian Bala and Artur Koninski. Unambiguous automata denoting finitely sequential functions. In Adrian-Horia Dediu, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications - 7th International Conference, LATA 2013, Bilbao, Spain, April 2-5, 2013. Proceedings*, volume 7810 of *Lecture Notes in Computer Science*, pages 104–115. Springer, 2013.

[BR82] Jean Berstel and Christophe Reutenauer. Recognizable formal power series on trees. *Theor. Comput. Sci.*, 18:115–148, 1982.

[CDZ14] Thomas Colcombet, Laure Daviaud, and Florian Zuleger. Size-change abstraction and max-plus automata. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part I*, volume 8634 of *Lecture Notes in Computer Science*, pages 208–219. Springer, 2014.

[CMMR20] Agnishom Chattopadhyay, Filip Mazowiecki, Anca Muscholl, and Cristian Riveros. Pumping lemmas for weighted automata. *CoRR*, abs/2001.06272, 2020.

[DGM17] Laure Daviaud, Pierre Guillon, and Glenn Merlet. Comparison of max-plus automata and joint spectral radius of tropical matrices. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, volume 83 of *LIPIcs*, pages 19:1–19:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[DRT16] Laure Daviaud, Pierre-Alain Reynier, and Jean-Marc Talbot. A generalised twinning property for minimisation of cost register automata. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 857–866. ACM, 2016.

[GM99] Stéphane Gaubert and Jean Mairesse. Modeling and analysis of timed petri nets using heaps of pieces. *IEEE Trans. Automat. Contr.*, 44(4):683–697, 1999.

[Has88] Kosaburo Hashiguchi. Algorithms for determining relative star height and star height. *Inf. Comput.*, 78(2):124–169, 1988.

[Kir08] Daniel Kirsten. A burnside approach to the termination of mohri's algorithm for polynomially ambiguous min-plus-automata. *ITA*, 42(3):553–581, 2008.

[KL09] Daniel Kirsten and Sylvain Lombardy. Deciding unambiguity and sequentiality of polynomially ambiguous min-plus automata. In Susanne Albers and Jean-Yves Marion, editors, *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, volume 3 of *LIPIcs*, pages 589–600. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.

[KLMP04] Ines Klimann, Sylvain Lombardy, Jean Mairesse, and Christophe Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theor. Comput. Sci.*, 327(3):349–373, 2004.

[Kui97] Werner Kuich. Formal power series over trees. In Symeon Bozapalidis, editor, *Proceedings of the 3rd International Conference Developments in Language Theory, DLT 1997, Thessaloniki, Greece, July 20-23, 1997*, pages 61–101. Aristotle University of Thessaloniki, 1997.

[Moh97] Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

[MPR02] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

[MR18] Filip Mazowiecki and Cristian Riveros. Pumping lemmas for weighted automata. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, pages 50:1–50:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[Pau17] Erik Paul. The equivalence, unambiguity and sequentiality problems of finitely ambiguous max-plus tree automata are decidable. In Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin, editors, *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, volume 83 of *LIPIcs*, pages 53:1–53:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[Pau19] Erik Paul. Finite sequentiality of unambiguous max-plus tree automata. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPIcs*, pages 55:1–55:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[Sch61] Marcel Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2-3):245–270, 1961.

[Sei94] Helmut Seidl. Finite tree automata with cost functions. *Theor. Comput. Sci.*, 126(1):113–142, 1994.

[Sim78] Imre Simon. Limited subsets of a free monoid. In *19th Annual Symposium on Foundations of Computer Science, Ann Arbor, Michigan, USA, 16-18 October 1978*, pages 143–150. IEEE Computer Society, 1978.

[Sim88] Imre Simon. Recognizable sets with multiplicities in the tropical semiring. In Michal Chytil, Ladislav Janiga, and Václav Koubek, editors, *Mathematical Foundations of Computer Science 1988, MFCS'88, Carlsbad, Czechoslovakia, August 29 - September 2, 1988, Proceedings*, volume 324 of *Lecture Notes in Computer Science*, pages 107–120. Springer, 1988.

[Wal07] Johannes Waldmann. Weighted automata for proving termination of string rewriting. *Journal of Automata, Languages and Combinatorics*, 12(4):545–570, 2007.

[WS91] Andreas Weber and Helmut Seidl. On the degree of ambiguity of finite automata. *Theor. Comput. Sci.*, 88(2):325–349, 1991.