A Generalised Twinning Property for Minimisation of Cost Register Automata

Laure Daviaud, Pierre-Alain Reynier, Jean-Marc Talbot

A Generalised Twinning Property for Minimisation of Cost Register Automata

Laure Daviaud, Pierre-Alain Reynier, and Jean-Marc Talbot

Aix Marseille Université, CNRS, LIF UMR 7279

Abstract

Weighted automata (WA) extend finite-state automata by associating weights with transitions. Unambiguous WA are such that for each word, there is at most one accepting run. Unambiguous WA are equivalent to functional WA, *i.e.* WA such every two runs on the same input have the same weight. Recently, cost register automata have been introduced as an alternative model to describe any function realised by a WA by means of a deterministic machine. Given a monoid $(M, \otimes)$, we denote by $\mathsf{CRA}_{\otimes c}(M)$ the cost register automata whose registers take their values in M, and are updated by operations of the form $x := y \otimes c$, with $c \in M$.

We introduce a twinning property and a bounded variation property parameterised by an integer k, such that the corresponding notions introduced by Choffrut are obtained for $k = 1$. Given an unambiguous weighted automaton W over an infinitary group $(G, \otimes)$, we prove that the three following properties are equivalent: *i*) W satisfies the twinning property of order k, *ii*) the function it realises satisfies the k-bounded variation property, and *iii*) this function can be described by a $\mathsf{CRA}_{\otimes c}(G)$ with at most k registers. We actually prove this result in the more general setting of finite-valued weighted automata. We show that if the operation of the group is computable, then one can decide whether a WA satisfies the twinning property of order k. As a corollary, this allows to decide the register minimisation problem for the class $\mathsf{CRA}_{\otimes c}(G)$.

Last, we prove that a similar result holds for finite-valued finite-state transducers, and that the register minimisation problem for the class $\mathsf{CRA}_{.c}(B^*)$ is PSPACE-complete.

1 Introduction

Finite state automata can be viewed as functions from words to Booleans and, thus, describe languages. Such automata have been extended to define functions from words to various structures yielding a very rich literature, with recent applications in quantitative verification [5]. Weighted automata [13] (WA) is the oldest of such formalisms. They are defined on semirings $(S, \oplus, \otimes)$ by adding weights from S on transitions; the weight of a run is the product of the weights of the transitions, and the weight of a word w is the sum of the weights of the accepting runs on w. Many problems have been considered for such machines such as equivalence [11], determinisation [12, 4, 10] and minimisation [12]. Most of the results highly depend on the considered semiring and are often undecidable.

Recently, Alur et al. have introduced a new model of machine, named cost register automata (CRA) [2]. These automata are deterministic but use registers that aim to store along the computation computed values from a given set S. A final output function associates with each final state a register. Hence, a step of computation boils down to register update: for each register a new value is computed from the stored values and a collection of operations defined on S. Considering a semiring $\mathbb{S} = (S, \oplus, \otimes)$, Alur et al. showed that WA over $\mathbb{S}$ and CRA defined over S with operations $\oplus$ and $\otimes c$ (*ie* the function $x \mapsto x \otimes c$ for c in S) compute the same functions [2]. Moreover, they showed that when unambiguous WA are considered (*ie* automata having at most one successful computation per input, thus making the additive law of the semiring useless), they turn out to be equivalent to CRA over S with $\otimes c$ as unique operation, denoted by $\mathsf{CRA}_{\otimes c}(S)$. In the particular case $\mathbb{S} = (\mathbb{Z}, +, \times)$, we obtain the model called "additive cost register automata" (ACRA) in [3].

Transducers [7] generalise automata by associating finite words to transitions. This way, they define rational relations over words. They can be viewed as weighted automata on the semiring of finite sets of words (thus, built over the free monoid); product is the set union and sum is the concatenation extended to sets. A transducer is functional (resp. finite-valued) if it associates a singleton with each input (resp. if, for some $k \in \mathbb{N}$, it associates at most k output words with each input). Deterministic (or sequential) transducers are those such that the underlying automaton is deterministic; they are thus functional. Determinisability is the decision problem asking whether for some transducer, there exists an equivalent deterministic one. In [6], Choffrut introduced two properties: first, a property of transducers named "twinning property" (TP) and second, a property of string functions named "bounded variation". He showed that a transducer T is determinisable iff T satisfies the twinning property iff the function f computed by T satisfies the bounded variation property. It has been first shown in [14] that the twinning property is decidable in Ptime.

Finite-valued weighted automata. As several problems are undecidable for general weighted automata, it is important to identify large subclasses with decidability results. While equivalence is undecidable for non-deterministic transducers, it is decidable for finite-valued ones. This class also enjoys other interesting properties, such as equivalence with finitely ambiguous transducers, and decomposition as finite unions of unambiguous transducers. These positive results motivated the study of WA with a "set" semantics in order to obtain decidability results for a large and expressive subclass. Instead of aggregating the weights of the different runs on the same input by using the first operation of the semi-ring, the semantics is defined as the set of these weights. A *functional* (resp. k-valued) WA is such that all accepting runs on the same input word have same weight (resp. such that, for every input word w, the set of values computed by all the accepting runs on w has cardinality at most k). It is finite valued if it is k-valued for some k. For instance, unambiguous and functional WA are equivalent, and determinisability is proven to be decidable for functional weighted automata using a twinning property under some hypotheses on the semiring (see [12, 10]). More recently, determinisability has been studied for functional weighted automata over groups satisfying the so-called infinitary condition [8]. Results on decomposition and inclusion have also been obtained in [9] for the class of finite-valued weighted automata.

For automata, a very important problem is to simplify the models. For instance, determinisation is essential in order to derive efficient evaluation algorithms. Similarly, reducing the size of the models allows to reduce the computation time of most algorithms, and thus minimisation has been extensively studied. For CRA, the most important problem concerns the minimisation of the number of registers: can we compute the minimal number of registers necessary to realise a given function? This highly challenging problem has been addressed in a recent paper for the particular case of ACRA [3].

One can notice that the class of $\mathsf{CRA}_{\otimes c}(S)$ with a single register coincides with the class of deterministic weighted automata on S. Hence, when $(S, \otimes)$ is an infinitary group, [8] entails that the twinning property characterises functional weighted automata on S that can be expressed by a $\mathsf{CRA}_{\otimes c}(S)$ with a single register.

Contributions. We start with the framework of infinitary groups and generalise this characterisation to finite-valued weighted automata and to an arbitrary number of registers using a twinning property of order k (TP_k). As a first step, and in order to capture the expressiveness of finite-valued weighted automata, we allow the final output function of $\mathsf{CRA}_{\otimes c}(S)$ to produce a subset of the register's values, and denote by $\mathsf{CRA}^{+}_{\otimes c}(S)$ the resulting class. Then,

our main result states that a finite-valued weighted automaton satisfies the TP_k iff it can be expressed by a $\mathsf{CRA}^{+}_{\otimes c}(S)$ with k registers. In addition, we also provide a generalisation of the bounded variation property that is also equivalent to the previously described class of functions. Provided that the group $(S, \otimes)$ is commutative and that the operation $\otimes$ and the equality on S are computable, we describe a procedure to decide whether a finite-valued WA satisfies the TP_k. As a corollary, we obtain the decidability of the problem of register minimisation for the class $\mathsf{CRA}^{+}_{\otimes c}(S)$. In particular, we obtain a PSPACE procedure for the group $(\mathbb{Z}, +)$, hence a slight generalisation of the result obtained in [3] for ACRA. It is worth noticing that our result relies on a general methodology, while the proof of [3] was tailored to the setting of ACRA.

As a complement to the previous results, we study the framework of transducers from A^* to B^*. It is known that streaming string transducers (*i.e.* CRA with a special set of updates) are expressively equivalent to regular string functions [1], and that the class $\mathsf{CRA}_{\cdot c}(B^*)$ coincides with the class of rational functions. In order to apply the above results, and as the set of words equipped with concatenation is not a group, we consider the free group over B. We prove that if a CRA over the free group only produces as output words in B^*, then there exists an equivalent CRA over B^*. The above results yield: a finite-valued transducer satisfies the TP_k iff it can be expressed by a $\mathsf{CRA}^{+}_{\cdot c}(B^*)$ with at most k registers. In addition, we also prove that the twinning property of order k can be decided in PSPACE, hence so register minimisation in the class $\mathsf{CRA}^{+}_{\cdot c}(B^*)$.

Organisation of the paper. We start with definitions in Section 2. In Section 3, we introduce the generalised twinning and bounded variation properties.We state our main result, in the context of infinitary groups in Section 4 and sketch its proof. We turn to transducers in Section 5. Last we present our decidability results and their application to the register minimisation problem in Section 6. Omitted proofs can be found in the Appendix.

2 Preliminaries

Prerequisites and notations. We denote by A a finite alphabet, by A^* the set of finite words on A, by ε the empty word and by $|w|$ the length of a word w. Given a set S, the set of the finite subsets of S is denoted by $\mathcal{P}_f(S)$.

A *monoid* $\mathbb{M} = (M, \otimes, \mathbf{1})$ is a set M equipped with an associative binary operation $\otimes$ with $\mathbf{1}$ as neutral element; the product $\alpha \otimes \beta$ in M may be simply denoted by $\alpha\beta$. Given $O, O' \subseteq M$, $O \otimes O'$ (or simply OO') is the set $\{\alpha\beta \mid \alpha \in O,\ \beta \in O'\}$, O^k denotes the set $\underbrace{OO\cdots O}_{k \text{ times}}$, $O^{<k} = \cup_{0 \leqslant i < k} O^i$ and $O^{\leqslant k} = O^{<k} \cup O^k$. If every element of a monoid possesses an inverse - for all $\alpha \in S$, there exists β such that $\alpha\beta = \beta\alpha = \mathbf{1}$ (such a β is unique and is denoted by α^{-1}) - then M is called a group. In this case, given $O \subseteq M$, O^{-1} denotes the set $\{\alpha^{-1} \mid \alpha \in O\}$. The monoid (*resp.* group) is said *commutative* when $\cdot$ is commutative.

A *semiring* $\mathbb{S}$ is a set S equipped with two binary operations $\oplus$ (sum) and $\otimes$ (product) such that $(S, \oplus, \mathbf{0})$ is a commutative monoid of neutral element $\mathbf{0}$, $(S, \otimes, \mathbf{1})$ is a monoid of neutral element $\mathbf{1}$, $\mathbf{0}$ is absorbing for $\otimes$ (*i.e.* $\alpha \otimes \mathbf{0} = \mathbf{0} \otimes \alpha = \mathbf{0}$) and $\otimes$ distributes over $\oplus$ (*i.e.* $\alpha \otimes (\beta \oplus \gamma) = (\alpha \otimes \beta) \oplus (\alpha \otimes \gamma)$ and $(\alpha \oplus \beta) \otimes \gamma = (\alpha \otimes \gamma) \oplus (\beta \otimes \gamma)$).

For a monoid $\mathbb{M}$, the set $\mathcal{P}_f(\mathbb{M})$ equipped with the two operations $\cup$ (union of two sets) and $\cdot$ is a semiring denoted $\mathbb{P}_f(\mathbb{M})$.

From now on, we may confuse algebraic structures (monoid, group, semiring) with the set they are defined on when the operations are clear from the context.

Delay and infinitary group. There exists a classical notion of *distance* on words (*i.e.* on the free monoid) measuring their difference: *dist* is defined for any two words u, v as $dist(u, v) = |u| + |v| - 2 * |lcp(u, v)|$ where $lcp(u, v)$ is the longest common prefix of u and v.

When considering groups, this notion is similar to the notion of delay:

▶ **Definition 1** (delay)**.** Let $\mathbb{G}$ be a group. Given $\alpha,\ \beta \in \mathbb{G}$, the *delay* between α and β is $\alpha^{-1}\beta$. It is denoted by $d(\alpha, \beta)$.

▶ **Lemma 2.** *Given a group $\mathbb{G}$, for all $\alpha, \alpha', \beta, \beta', \gamma, \gamma' \in \mathbb{G}$,*

1. *$d(\alpha, \beta) = 1$ if and only if $\alpha = \beta$,*
2. *if $d(\alpha, \alpha') = d(\beta, \beta')$ then $d(\alpha\gamma, \alpha'\gamma') = d(\beta\gamma, \beta'\gamma')$.*

▶ **Definition 3.** A group $\mathbb{G}$ is said to be *infinitary* if for all $\alpha, \alpha', \beta, \beta' \in \mathbb{G}$ such that $d(\alpha, \alpha') \neq d(\alpha\beta, \alpha'\beta')$:

$$|\{d(\alpha\beta^n, \alpha'\beta'^n) \mid n \in \mathbb{N}\}| = +\infty$$

Classical examples of infinite groups such as $(\mathbb{Z}, +, 0)$, $(\mathbb{Q}, \times, 1)$ and the free group generated by a finite alphabet are all infinitary. Other examples are given in [8].

Weighted automata. Given a semiring $\mathbb{S}$, weighted automata are non-deterministic finite automata in which transitions have weights as elements of $\mathbb{S}$. They compute functions from the set of words to $\mathbb{S}$. The weight of a run is the product of the weights of the transitions along the run and the weight of a word w is the sum of the weights of the accepting runs labelled by w.

We will consider, for some monoid $\mathbb{M}$, weighted automata over the semiring $(\mathcal{P}_f(\mathbb{M}), \cup, \otimes)$. Wlog, we assume that transitions are labelled by elements in $\mathbb{M}$.

A *weighted automaton* W over the alphabet A with weights in the monoid $\mathbb{M}$ is a tuple $(Q, Q_{init}, Q_{final}, t, T)$ where Q is a finite set of states, $Q_{init} \subseteq Q$ is the set of initial states, $Q_{final} \subseteq Q$ is the set of final states, $t : Q_{final} \to \mathbb{M}$ is the output function and $T \subseteq Q \times A \times \mathbb{M} \times Q$ is the finite set of transitions.

A run ρ on a word $w = w_1 \cdots w_k \in A^*$ where for all i, $w_i \in A$, is a sequence of transitions $(q_1, w_1, \alpha_1, q_2), (q_2, w_2, \alpha_2, q_3), \ldots, (q_k, w_k, \alpha_k, q_{k+1})$. The *output* (or *value*) of such a run is the element of $\mathbb{M}$ $\alpha_1\alpha_2 \cdots \alpha_k$. A state q is *accessible* (resp. *co-accessible*) if there exists such a run with $q_1 \in Q_{init}$ and $q = q_k$ (resp. $q_{k+1} \in Q_{final}$ and $q = q_1$). The run ρ is said to be accepting if $q_1 \in Q_{init}$ and $q_{k+1} \in Q_{final}$. Moreover we say that the value of such an accepting run is $\alpha = \alpha_1\alpha_2 \cdots \alpha_k t(q_{k+1})$. We depict this situation as $q_1 \xrightarrow{w|\alpha} q_{k+1}$.

This automaton W computes the (total) function $[\![W]\!]$ which associates with any word w the set of the values of the accepting runs on w (and so, $\varnothing$ if there is no such run).

A weighed automaton is unambiguous if it admits at most one accepting run for each input. For such automata, the additive operation of the semiring is somehow useless.

Given a weighted automaton W over a group $\mathbb{G}$, we denote by Δ_W the set of all the elements of $\mathbb{G}$ occurring on the transitions of W as well as their inverse.

▶ **Definition 4** (Valuedness / Ambiguity)**.** For any positive integer ℓ, a weighted automaton W is said to be:

- *ℓ-valued* if for all words w, the set $[\![W]\!](w)$ contains at most ℓ elements,
- *ℓ-ambiguous* if for all words w, there are at most ℓ accepting runs labelled by w.

A weighted automaton is said to be *finitely valued* (resp. *finitely ambiguous*) if it is ℓ-valued (resp. ℓ-ambiguous) for some positive integer ℓ.

$\mathcal{WA}_\ell$ denotes the set of functions $A^* \to \mathcal{P}_f(\mathbb{M})$ computed by ℓ-valued weighted automata.

▶ **Example 5.** Let us consider $A = \{a, b\}$ and $(\mathbb{M}, \otimes, \mathbf{1}) = (\mathbb{N}, +, 0)$. The weighted automaton given in Figure 1 (on the left) associates with a word wa (resp. wb) its number of occurrences of the letter a (resp. b). It is 1-valued and 1-ambiguous.

Cost register automata. Cost register automata (CRA) [2] are defined over a set $\mathbb{C}$ equipped with a collection of operations $\mathbb{O}$. A CRA is a deterministic automaton with registers containing values from $\mathbb{C}$ and that are updated through the transitions: for each register, its new value is computed from the old ones combined using an operation in $\mathbb{O}$. The output value is computed from the values taken by the registers at the end of the processing of the input. Hence, a CRA defines a function from words in A^* to elements of $\mathbb{C}$.

In this paper, we focus on a particular structure $(\mathbb{M}, \otimes c)$ defined over a monoid $(\mathbb{M}, \otimes, \mathbf{1})$ that we denote $\mathsf{CRA}_{\otimes c}(\mathbb{M})$. In such a structure, the only operations are unary and defined as $x \mapsto x \otimes c$ with c is an element from $\mathbb{M}$. When $\mathbb{M}$ is $(\mathbb{N}, +, 0)$, this class of automata is called additive cost register automata [3]. When $\mathbb{M}$ is the free monoid $(A^*, ., \epsilon)$, this class is a subclass of streaming string transducers [1] and turns out be to equivalent to the class of rational functions on words, *i.e.* realized by finite-state transducers.

Moreover, we want to extend $\mathsf{CRA}_{\otimes c}(\mathbb{M})$ to compute not only single values from $\mathbb{M}$ but also finite sets of such elements. For this, we slightly modify the definition of CRA, allowing to produce a set of values computed from register contains. The new class of machines denoted $\mathsf{CRA}^{+}_{\otimes c}(\mathbb{M})$ is defined formally as follows:

▶ **Definition 6.** A *cost register automaton* on the alphabet A over the monoid $(\mathbb{M}, \otimes, \mathbf{1})$ with updates $\otimes c$ is a tuple $(Q, q_{init}, \mathcal{X}, \delta, \mu)$ where Q is a finite set of states, $q_{init} \in Q$ is the initial state and $\mathcal{X}$ is a finite set of registers. The transitions are given by the function $\delta : Q \times A \to (Q \times \mathcal{UP}(\mathcal{X}))$ where $\mathcal{UP}(\mathcal{X})$ is the set of functions $\mathcal{X} \to \mathcal{X} \times \mathbb{M}$ that represents the updates on the registers. Finally, $\mu : Q \to \mathcal{P}_f(\mathcal{X} \times \mathbb{M})$ is the output function.

The semantic of such an automaton is as follows: if an update function f labelled a transition and $f(Y) = (X, \alpha)$, then the register Y after the transition will take the value $\beta\alpha$ where β is the value contained in the register X before the transition. More precisely, a valuation ν is a mapping from $\mathcal{X}$ to $\mathbb{M}$ and let $\mathcal{V}$ be the set of such valuations. The initial valuation ν_{init} is the function associating with each register the value $\mathbf{1}$. A configuration is an element of $Q \times \mathcal{V}$. The initial configuration is (q_{init}, ν_{init}). A run on a word $w = w_1 \cdots w_k \in A^*$ where for all i, $w_i \in A$, is a sequence of configurations $(q_1, \nu_1)(q_2, \nu_2) \dots (q_{k+1}, \nu_{k+1})$ satisfying that for all $1 \leqslant i \leqslant k$, $\delta(q_i, w_i) = (q_{i+1}, g_i)$ with for all registers X, if $g_i(Y) = (X, \alpha)$ then $\nu_{i+1}(Y) = \nu_i(X)\alpha$. Moreover, the run is said to be accepting if (q_1, ν_1) is the initial configuration.

A cost register automaton R defines a (total) function $[\![R]\!]$ from words to finite subsets of $\mathbb{M}$ such as for all w, $[\![R]\!](w)$ is equal to

$$\{\nu_{k+1}(X)\alpha \mid (q_1, \nu_1)(q_2, \nu_2) \dots (q_{k+1}, \nu_{k+1}) \text{ is an accepting run on } w \text{ and } (X, \alpha) \in \mu(q_{k+1})\}$$

▶ **Definition 7** (Output size)**.** The *output size* of a cost register automaton with output function μ is the integer $\max\{|\mu(q)| \mid q \in Q\}$.

The class of such cost register automata whose output size is at most ℓ is denoted $\mathsf{CRA}^{\ell}_{\otimes c}(\mathbb{M})$. $\mathsf{CRA}^{+}_{\otimes c}(\mathbb{M})$ is defined as the union of $\mathsf{CRA}^{\ell}_{\otimes c}(\mathbb{M})$ over $\ell \geqslant 1$.

Let $\mathcal{RA}_\ell(k)$ (resp. $\mathcal{RA}(k)$, $\mathcal{RA}_\ell$) denote the set of functions $A^* \to \mathcal{P}_f(\mathbb{M})$ computed by cost register automata with at most k registers and output size at most ℓ (resp. at most k registers, of output size at most ℓ).

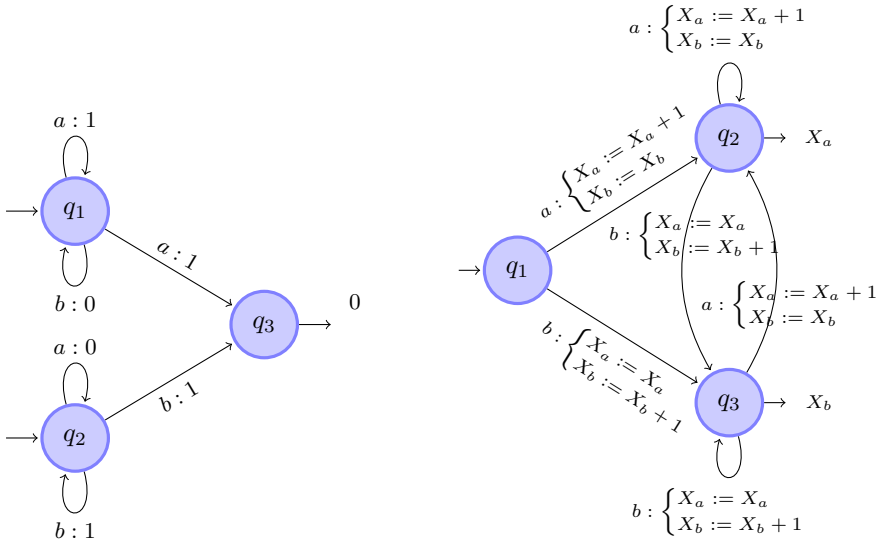

Figure 1 Examples of weighted automaton (left) and of cost register automaton (right).

▶ **Example 8.** Consider $A = \{a, b\}$ and $(\mathbb{M}, \otimes, \mathbf{1}) = (\mathbb{N}, +, 0)$. The cost register automaton given in Figure 1 (on the right) computes the same function as the one computed by the weighted automaton from Example 5. The register X_a (resp. X_b) stores the number of occurrences of the letter a (resp. b). It uses two registers and is of output size 1.

3 Generalised twinning and bounded variation properties

In this section, we present a twinning property and a bounded variation property parameterised by an integer k, such that the corresponding notions introduced by Choffrut in [6] are obtained for $k = 1$. Our properties are defined for the setting of groups.

3.1 Generalised twinning property (TP_k)

The idea behind the twinning property of order k is to consider $k+1$ runs labelled by the same word with k synchronized cycles. If the twinning property of order k is satisfied then there are two runs among these $k+1$ such that the values along these two runs remain close (*i.e.* the delay between these values stays within a fixed and finite subset of $\mathbb{G}$).

▶ **Definition 9.** A weighted automaton on a group $\mathbb{G}$ satisfies the *twinning property of order* k (denoted by TP_k) if:

- for all states $\{q_i^j \mid i, j \in \{0, \dots, k\}\}$ with q_0^j initial and q_k^j co-accessible for all j,
- for all words $u_1, \dots, u_k$, $v_1, \dots, v_k$ such that there are $k+1$ runs satisfying for all $0 \leq j \leq k$, for all $1 \leq i \leq k$, $q_{i-1}^j \xrightarrow{u_i|\alpha_i^j} q_i^j$ and $q_i^j \xrightarrow{v_i|\beta_i^j} q_i^j$ (see Figure 2),

there are $j \neq j'$ such that for all $i \in \{1, \dots, k\}$,

$$d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'}) = d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j \beta_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'} \beta_i^{j'})$$

We can now state some properties on runs of weighted automata depending whether they satisfy TP_k or not. The next lemma formalizes the intuition that when the TP_k is satisfied, there are two runs which are close at the end (Δ_W is defined in the preliminaries).

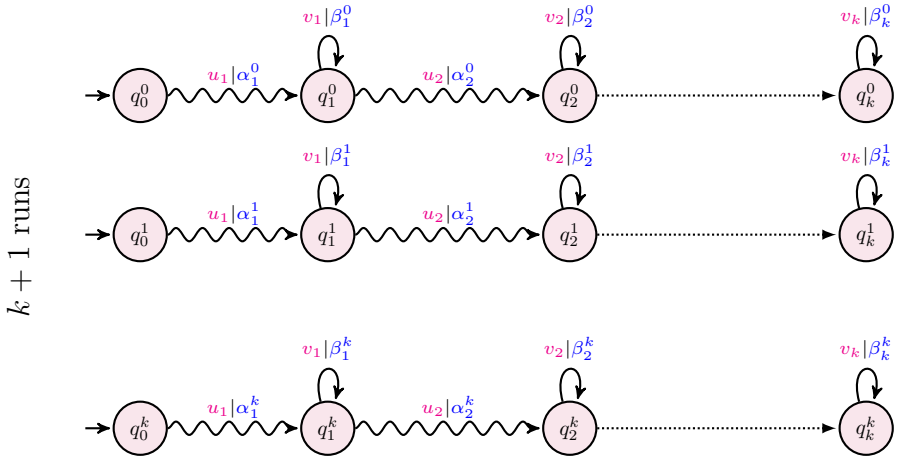

Figure 2 Twinning property of order k.

▶ **Lemma 10.** *Let W be a weighted automaton with n states satisfying TP_k. Then, for all words w, for all initial states $q^0, \dots, q^k$ and co-accessible states $p^0, \dots p^k$ such that there are $k+1$ runs*

$$q^j \xrightarrow{w|\alpha^j} p^j \quad \text{for all } j \in \{0, \dots, k\},$$

there are $j \neq j'$ such that $d(\alpha^j, \alpha^{j'}) \in \Delta_W^{\leqslant 2n^{k+1}}$.

This lemma relies on the fact that the group $\mathbb{G}$ is infinitary.

▶ **Lemma 11.** *Let W be a weighted automaton on an* infinitary group *that does not satisfy TP_k. Then, for all positive integers m, there is a word w, initial states $q^0, \dots, q^k$, co-accessible states $p^0, \dots, p^k$ and $k+1$ runs:*

$$q^j \xrightarrow{w|\alpha^j} p^j \text{ for all } j \in \{0, \dots, k\},$$

such that for all $j \neq j'$, $d(\alpha^j, \alpha^{j'}) \notin \Delta_W^{\leqslant m}$.

We introduce a definition expressing the fact that two runs are *always* close:

▶ **Definition 12.** Let Λ be a finite subset of $\mathbb{G}$, W be a weighted automaton on $\mathbb{G}$ and $w = w_1 \dots w_k$ be a word. Let ρ^1, ρ^2 be two runs on w, with $\rho^j = q_1 \xrightarrow{w_1|\beta_1^j} q_2 \cdots q_k \xrightarrow{w_k|\beta_k} q_{k+1}$, $j = 1, 2$. ρ^1, ρ^2 are said to be Λ-*close* if for all $i \geq 1$, $d(\beta_1^1 \dots \beta_i^1, \beta_1^2 \dots \beta_i^2) \in \Lambda$.

▶ **Lemma 13.** *Let W be a weighted automaton with n states satisfying TP_k. Then, for all r, for all words w, for all runs $\rho^1, \dots, \rho^r$ on w, from an initial state to a co-accessible state, there is a subset $P \subseteq \{1, \dots, r\}$ containing at most k elements such that for all $j' \in \{1, \dots, r\}$, there is $j \in P$ such that $\rho^j, \rho^{j'}$ are $(\Delta_W^{\leqslant 2n^{k+1}})$-close.*

3.2 Finite-valued weighted automata and TP_k

Based on the Lemmas from the previous subsection, we exhibit now the strong relationship between the twinning property and the finite-valuedness property for weighted automata.

▶ **Proposition 14.** *A weighted automaton over an infinitary group satisfies TP_k for some natural k if and only if it is finitely valued.*

First, if a weighted automaton is ℓ-valued then it satisfies $TP_{n\ell}$ where n is its number of states. This is proved by contradiction using Lemma 11. The converse follows from Lemma 13 applied on a set of accepting runs.

3.3 Bounded variation property

The bounded variation property is defined on functions and is thus a machine independent property: whenever two WA are equivalent, either both or none of them satisfy this property.

Given a partial mapping $f : A^* \rightharpoonup B^*$, the twinning property introduced by Choffrut in [6] states that for every $N \in \mathbb{N}$, there exists $n \in \mathbb{N}$ such that for all $w, w' \in A^*$ such that $f(w), f(w')$ are defined, if $dist(w, w') \leqslant N$, then $dist(f(w), f(w')) \leqslant N$. Intuitively, this property states that whenever two words only differ by a small suffix, so do their images by f. This corresponds to the intuition that the function can be expressed by means of a CRA with a single register (a behaviour can be deduced from the other one).

When lifting this property to functions that can be expressed using at most k registers, we consider $k+1$ input words pairwise close, and require that two of them must have close images by f. The extension to partial mappings $f : A^* \rightharpoonup \mathcal{P}_f(B^*)$ requires that for all $k+1$ pairwise close input words, and all $k+1$ output words chosen in the images of these input words, two of them should be close.

Last, our framework is that of infinitary groups. Instead of $dist(,)$, we use the delay $d(,)$ to compare the images. In order to quantify the "distance" between these outputs, we introduce a notion of generator that depends on the function under study. Given a mapping $f : A^* \to \mathcal{P}_f(\mathbb{G})$ computed by a weighted automaton, we say that a subset E of $\mathbb{G}$ is a *generator w.r.t. f* if $\{d(\alpha, \alpha') \mid \alpha \in f(w), \alpha' \in f(w'), w, w' \in A^*\} \subseteq \bigcup_{n \geqslant 0} E^n$. If this holds, we say that E satisfies the criterion (C_f). Note that if $\mathbb{G}$ is finitely generated, then any finite set of generators of E satisfies (C_f).

We present now our definition that generalises the original one of Choffrut [6]:

▶ **Definition 15.** A function $f : A^* \to \mathcal{P}_f(\mathbb{G})$ satisfies the *k-bounded variation property* if for all naturals $N > 0$ and all subsets E of $\mathbb{G}$ satisfying (C_f), there is a natural n such that for all words $w_0, \ldots, w_k \in A^*$, if for all $0 \leqslant i, j \leqslant k, dist(w_i, w_j) \leqslant N$ then for all $0 \leqslant i \leqslant k$, for all $\alpha_i \in f(w_i)$, there is $j \neq i$ such that $d(\alpha_i, \alpha_j) \in E^{\leqslant n}$.

4 Relating finite-valued weighted automata and cost-register automata

We present our main result stating the equivalence between the twinning property of order k, the k-bounded variation property, and the register complexity at most k.

▶ **Theorem 16.** *Let W be an ℓ-valued weighted automaton over the semiring $\mathbb{P}_f(\mathbb{G})$ where $(\mathbb{G}, \otimes)$ is an infinitary group, and k be a positive natural number. The following assertions are equivalent:*

1. *W satisfies the twinning property of order k,*
2. *$[\![W]\!]$ satisfies the k-bounded variation property,*
3. *$[\![W]\!]$ is computed by a $\mathsf{CRA}^{\ell}_{\otimes c}(\mathbb{G})$ with k registers.*

The proof of this theorem is obtained by showing the equivalence between 1 and 2 and then between 1 and 3. These proofs are sketched in Section 4.1 and Section 4.2.

This result allows to describe a hierarchy of functions as described in Figure 3 where $\mathcal{WA}(k)$ (resp. $\mathcal{WA}_\ell(k)$) denotes the set of functions computed by a weighted automaton (resp. ℓ-valued weighted automaton) satisfying TP_k.

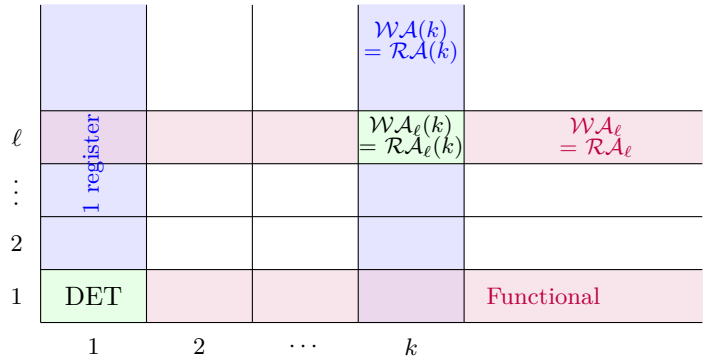

Figure 3 Hierarchy.

Consider an alphabet over k letters $A = \{a_1, \ldots, a_k\}$ and the function defined for all words w by:

$$f : wa_i \mapsto \{|w|_i + 1, |w|_i + 2, \ldots, |w|_i + \ell\}$$

where $|w|_i$ represents the number of occurrences of the letter a_i in w. One can prove that this function is in the class $\mathcal{RA}_\ell(k)$, but not in the classes $\mathcal{RA}_{\ell'}(k')$ for $k' < k$ or $\ell' < \ell$, showing that this hierarchy is strict.

4.1 Proof of the machine independent characterisation

In this section, we sketch the proof of the equivalence of assertions 1 and 2 of Theorem 16.

First, consider a weighted automaton W with n states satisfying TP_k. By Lemma 10, we can find a bound N such that among any $k+1$ runs, there always exists two runs with delay in $\Delta_W^{\leqslant N}$. Then, for all subset E satisfying criterion (C_f), we can prove that we can find a bound N' such that the lemma still holds for E. It allows us to prove that the function computed by W satisfies the k-bounded variation property.

Conversely, if the function computed by W satisfies the k-bounded variation property, it means that there is an integer N such that among any $k+1$ runs over a same word, there are two runs with delay in $\Delta_W^{\leqslant n}$. By using Lemma 11, we can prove that W satisfies TP_k.

4.2 Proof of the equivalence with cost register automata

In this section, we sketch the proof of the equivalence of assertions 1 and 3 of Theorem 16, which amounts to prove the equality $\mathcal{RA}_\ell(k) = \mathcal{WA}_\ell(k)$.

The inclusion $\mathcal{RA}_\ell(k) \subseteq \mathcal{WA}_\ell(k)$ is the easiest one. The construction is similar to the one given in [2] to construct a weighted automaton from a cost register automaton. One can verify that parameters k and ℓ are preserved by this construction.

Let us focus on the reverse inclusion and denote by $\mathcal{WA}_{\ell\text{-amb}}(k)$ the set of functions computed by a ℓ-ambiguous weighted automaton satisfying TP_k. It is proved in [9] that $\mathcal{WA}_\ell = \mathcal{WA}_{\ell\text{-amb}}$. Moreover, thanks to the equivalence of assertions 1 and 2 of Theorem 16 that we just proved, the twinning property of order k is a machine independent property. We can deduce that $\mathcal{WA}_\ell(k) = \mathcal{WA}_{\ell\text{-amb}}(k)$.

It is then sufficient to prove that $\mathcal{WA}_{\ell\text{-}amb}(k) \subseteq \mathcal{RA}_\ell(k)$. Consider a ℓ-ambiguous weighted automaton W with set of states Q satisfying TP_k, and let $\Gamma = \Delta_W^{2|Q|^{k+1}}$. We build $R \in \mathsf{CRA}^\ell_{\otimes c}(\mathbb{G})$ with k registers computing the same function.

The idea underlying the construction is to store in the states of R the delays between the values of the runs of W labelled by a given word. We will use the fact that there are at most k diverging behaviours (thanks to TP_k) to prove that we can store the delays up to a finite bound and use only k registers to capture the k diverging behaviours.

More precisely, let w be a word and $q \in Q$. We write $\rho_q^1, \rho_q^2, \dots, \rho_q^{\ell_q}$ the runs in W labelled by w from an initial state to q. Let $\alpha_q^1, \alpha_q^2, \dots, \alpha_q^{\ell_q}$ denote their respective weights. By ℓ-ambiguity of W, for all $q \in Q$, we have $\ell_q \leqslant \ell$.

Let f be the function from $(Q \times \{1, \dots, \ell\})^2$ to $\Gamma \cup \{\infty, \bot\}$ such that:

$$f((q,\kappa),(q',\kappa')) = \begin{cases} d(\alpha_q^\kappa, \alpha_{q'}^{\kappa'}) & \text{if } \kappa \leqslant \ell_q, \kappa' \leqslant \ell_{q'}, (\rho_q^\kappa, \rho_{q'}^{\kappa'}) \text{ } \Gamma\text{-close} \\ \infty & \text{if } \kappa \leqslant \ell_q, \kappa' \leqslant \ell_{q'}, (\rho_q^\kappa, \rho_{q'}^{\kappa'}) \text{ not } \Gamma\text{-close} \\ \bot & \text{otherwise (\textit{i.e.} one of the two runs doesn't exist)} \end{cases}$$

The cost register automaton R is constructed such that the unique run labelled by w in R from the initial state ends in a state representing the function f.

By Lemma 13 (that holds thanks to TP_k), one can choose a set $P \subseteq \cup_{q \in Q}(\{q\} \times \{1, \dots, \ell_q\})$ of k elements such that for all $q \in Q$ and $\kappa \in \{1, \dots, \ell_q\}$, there is $(q', \kappa') \in P$ such that the runs ρ_q^κ and $\rho_{q'}^{\kappa'}$ are Γ-close.

The k elements of P are mapped onto the k registers of R, and the updates of the registers are defined such that the values stored in the registers after reading w in R are the weights of these k runs. Thanks to the property of the set P stated above, the weight of any run along w can be obtained from one of the registers of R.

This allows to prove the correction of this construction.

5 The case of transducers

This section is devoted to prove that Theorem 16 still holds if we consider transducers from A^* to B^*. To this end, we view such transducers as weighted automata over the free semigroup on B, and the equivalence $1 \Leftrightarrow 2$ follows. The equivalence $1 \Leftrightarrow 3$ requires more attention.

More precisely, given a finite alphabet B, let $\mathcal{WA}_\ell^B(k)$ (resp. $\mathcal{RA}_\ell^B(k)$) denote the set of functions computed by a ℓ-valued weighted automaton satisfying TP_k (resp. a cost register automaton with k registers and output size ℓ) only using elements in B^*.

▶ **Theorem 17.** $\mathcal{WA}_\ell^B(k) = \mathcal{RA}_\ell^B(k)$

The proof of the inclusion $\mathcal{RA}_\ell^B(k) \subseteq \mathcal{WA}_\ell^B(k)$ is the same as the one done in the group case: in this proof, we never use group structure and the weights are preserved by the construction.

Let us now sketch the proof of the reverse inclusion. Let $\mathcal{F}$ denote the set of functions $A^* \to \mathcal{P}_f(B^*)$. By Theorem 16, we have the following sequence of inclusions:

$$\mathcal{WA}_\ell^B(k) \subseteq \mathcal{WA}_\ell(k) \cap \mathcal{F} \subseteq \mathcal{RA}_\ell(k) \cap \mathcal{F}$$

Thus, by proving Proposition 18, we will obtain the expected result.

▶ **Proposition 18.** $\mathcal{RA}_\ell(k) \cap \mathcal{F} \subseteq \mathcal{RA}_\ell^B(k)$

Consider a cost register automaton R that computes a function in $\mathcal{F}$. On can prove that there is a bound N such that along the runs of R, the value stored in the registers always belong to $B^*(B \cup B^{-1})^{\leqslant N}$. These values are thus of the form $\alpha_1\alpha_2$ with $\alpha_1 \in B^*$ and $\alpha_2 \in (B \cup B^{-1})^{\leqslant N}$. The idea is then to output the shortest α_1 satisfying these conditions and store the value α_2 in the states of the automaton. This construction preserves parameters k and ℓ.

6 Decidability and application to register minimisation

In this section, we prove the decidability of the twinning property, and as a consequence that of the register minimisation problem. We consider the two following problems:
Problem TP_k: given a weighted automaton W and a number k, does W satisfy the TP_k?
Problem Register Minimisation: given $R \in \mathsf{CRA}^+_{\otimes c}(S)$ and a number k, does there exist $R' \in \mathsf{CRA}^+_{\otimes c}(S)$ with k registers such that $[\![R]\!] = [\![R']\!]$?

We start with a preliminary result. Let us denote by TP'_k the property obtained from the TP_k by requiring the property not only for k cycles, but for m cycles, for every $m \geqslant k$.

▶ **Lemma 19.** *For all positive integer k, a weighted automaton satisfies TP_k if and only if it satisfies TP'_k.*

As a consequence, a witness of the violation of the TP_k can be identified as one of the violation of the TP'_k, *i.e.* a set of $k+1$ runs, with $m \geqslant k$ cycles, such that for each pair $i \neq j$, there exists a cycle that induces different delays between i-th and j-th runs.

Case of commutative groups. We write $W = (Q, Q_{init}, Q_{final}, t, T)$ and let $n = |W|$. In order to decide the twinning property, we will consider the $k+1$-th power of W, denoted W^{k+1}, which accepts the set of $k+1$ synchronised runs in W. We write its runs as $\vec{\rho} = (\rho_i)_{0 \leqslant i \leqslant k}$ and denote by α^i the weight of run ρ_i.

▶ **Theorem 20.** *Let $\mathbb{G} = (G, \otimes)$ be a commutative group such that the operation $\otimes$ and the equality check are computable. Then the TP_k problem is decidable.*

Sketch. It is easy to observe that for commutative groups, the constraint expressed on the delay in the twinning property boils down to checking that loops have different weights. The result follows from the two following facts:

- first, given two vectors of states $v, v' \in Q^{k+1}$, checking that there exists a path from v to v' in W^{k+1} is decidable,
- second, the following problem is decidable: given a vector of states $v \in Q^{k+1}$ and a pair $i \neq j$, check that there exists a cycle $\vec{\rho}$ around v in W^{k+1} such that $d(\alpha^i, \alpha^j) \neq \mathbf{1}$. The procedure non-deterministically guesses the cycle in W^{k+1} (its length can be bounded by $2n^{k+1}$) and computes incrementally the value of $d(\alpha^i, \alpha^j)$.

The overall procedure simply guesses a run in W^{k+1} with a cycle for each pair $i \neq j$, and checks that this cycle induces distinct delays between the i-th and j-th runs. ◀

If we consider the setting of ACRA, *i.e.* the group $(\mathbb{Z}, +)$, we can verify that the above procedure runs in Pspace if k is given in unary, yielding:

▶ **Theorem 21.** *Over the group $(\mathbb{Z}, +)$, the TP_k problem is in* Pspace *(k is given in unary).*

The construction from $\mathsf{CRA}^+_{\otimes c}(S)$ to WA over S is polynomial. Wlog, we suppose that k is given in unary. This is reasonable as k is smaller than the actual number of registers of R. As a consequence we deduce: (the hardness follows from the result of [3])

▶ **Corollary 22.** *The register minimisation problem for* $\mathsf{CRA}^{+}_{+c}(\mathbb{Z})$ *is* PSPACE*-complete.*

This result slightly generalises that of [3], as we allow more general output functions. In addition, it follows from a general framework, and similar results for other infinitary groups can be derived similarly.

Case of transducers. Let us first recall the procedure of [14] to decide the twinning property in PTIME for transducers. They prove that the *TP* is violated iff there exists a pair of runs such that either the output words on cycles v_1, v_2 are such that $|v_1| \neq |v_2|$, or the output words on paths leading to the cycle, say u_1, u_2, have a mismatch (*i.e.* a position on which they differ). Using a similar reasoning, we prove the following lemma:

▶ **Lemma 23.** *Let* T *be a transducer violating the* TP_k*. Then there exist:*

- *states* $\{q_i^j\}_{0\leqslant i\leqslant m, 0\leqslant j\leqslant k}$ *with* $k \leqslant m \leqslant k^2$*, and* q_0^j *initial and* q_k^j *co-accessible for all* j*,*
- *words* $u_1, \ldots, u_m$*,* $v_1, \ldots, v_m$ *such that there are* $k+1$ *runs satisfying for all* $0 \leq j \leq k$*, for all* $1 \leq i \leq m$*,* $q_{i-1}^j \xrightarrow{u_i|\alpha_i^j} q_i^j$ *and* $q_i^j \xrightarrow{v_i|\beta_i^j} q_i^j$

and such that for all $0 \leqslant j < j' \leqslant k$*:*

- *either there exists* $1 \leqslant i \leqslant m$ *such that* $|\beta_i^j| \neq |\beta_i^{j'}|$*,*
- *or there exists* $1 \leqslant i \leqslant m$ *such that* $|\beta_i^j| = |\beta_i^{j'}| \neq 0$*, the words* $\alpha_1^j \ldots \alpha_i^j$ *and* $\alpha_1^{j'} \ldots \alpha_i^{j'}$ *have a mismatch, and the runs* $q_0^j \xrightarrow{u_1 \ldots u_i} q_i^j$ *and* $q_0^{j'} \xrightarrow{u_1 \ldots u_i} q_i^{j'}$ *are* $\left(B \cup B^{-1}\right)^{n^{k+1}}$*-close.*

This allows to derive a non-deterministic procedure running in polynomial space (assuming k is given in unary): we first guess the vectors of states associated with cycles, and guess, for every $0 \leqslant j < j' \leqslant k$, which of the two cases holds. Using a procedure similar to the one described for the commutative case, one can check the existence of a cycle verifying the guessed property. Last, we verify the existence of the mismatches. Given a pair of runs (ρ, ρ'), we proceed as follows: one non-deterministically guess ρ and ρ' and stores the difference between the lengths of the outputs of the two runs. Non-deterministically, one can record the letter produced by the run that is ahead (say ρ). Then one continues the simulation until ρ' catches up ρ, and checks that the letter produced by ρ' is different. This can be achieved in polynomial space using the fact that ρ and ρ' are close.

▶ **Theorem 24.** *Over* $(B^*, \cdot)$*, the* TP_k *problem is in* PSPACE *(*k *is given in unary).*

▶ **Corollary 25.** *The register minimisation problem for* $\mathsf{CRA}^{+}_{.c}(B^*)$ *is* PSPACE*-complete.*

7 Conclusion

We have studied so-called finite-valued weighted automata on one side, and a class of cost register automata on the other side.

We have introduced a twinning property and a bounded-variation property that generalise the original properties introduced by Choffrut for transducers and obtained an elegant generalisation of a well-known result of Choffrut for transducers. In addition, this led to a decision procedure of a register minimisation problem for a large class of cost register automata.

Our setting includes both infinitary groups and transducers. It is worth observing that important classes of quantitative languages such as sum, discounted sum and average automata fit into the setting of infinitary groups (see [8]).

As a particular case, for the setting of additive cost regular functions, we obtain a generalization of the result of [3] on the minimisation of registers.

References

1. Rajeev Alur and Pavol Cerný. Expressiveness of streaming string transducers. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPIcs*, pages 1–12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.
2. 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.
3. Rajeev Alur and Mukund Raghothaman. Decision problems for additive regular functions. In *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.
4. Adam L. Buchsbaum, Raffaele Giancarlo, and Jeffery Westbrook. On the determinization of weighted finite automata. *SIAM J. Comput.*, 30(5):1502–1531, 2000.
5. Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Trans. Comput. Log.*, 11(4), 2010.
6. Christian Choffrut. Une caracterisation des fonctions sequentielles et des fonctions sous-sequentielles en tant que relations rationnelles. *Theor. Comput. Sci.*, 5(3):325–337, 1977.
7. Samuel Eilenberg. *Automata, Languages, and Machines*, volume vol. A. Academic Press, Inc., 1974.
8. Emmanuel Filiot, Raffaella Gentilini, and Jean-François Raskin. Quantitative languages defined by functional automata. *Logical Methods in Computer Science*, 11(3:14):1–32, 2015.
9. Emmanuel Filiot, Raffaella Gentilini, and Jean-François Raskin. Finite-valued weighted automata. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, volume 29 of *LIPIcs*, pages 133–145. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.
10. Daniel Kirsten and Ina Mäurer. On the determinization of weighted automata. *Journal of Automata, Languages and Combinatorics*, 10(2/3):287–312, 2005.
11. Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *IJAC*, 4(3):405–426, 1994.
12. Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.
13. Marcel-Paul Schützenberger. On the definition of a family of automata. *Information and Control*, 4, 1961.
14. Andreas Weber and Reinhard Klemm. Economy of description for single-valued transducers. *Inf. Comput.*, 118(2):327–340, 1995.

In all the results and proofs in the appendix, $W = (Q, Q_{init}, Q_{final}, T)$ denotes a weighted automaton computing a function f and k a positive integer. Let Θ denote the set of elements of $\mathbb{S}$ occurring on the transitions of W, $\Delta_W = \Theta \cup \Theta^{-1}$ and $\Gamma = \Delta_W^{\leqslant 2|Q|^{k+1}} = \Delta_W^{2|Q|^{k+1}}$.

A Delays

▶ **Lemma 2**. For all $\alpha, \alpha', \beta, \beta', \gamma, \gamma' \in \mathbb{S}$,
1. $d(\alpha, \beta) = 1$ if and only if $\alpha = \beta$,
2. if $d(\alpha, \alpha') = d(\beta, \beta')$ then $d(\alpha\gamma, \alpha'\gamma') = d(\beta\gamma, \beta'\gamma')$.

Proof. 1. $d(\alpha, \beta) = \alpha^{-1}\beta = 1$ if and only if $\alpha = \beta$.
2. $d(\alpha\gamma, \alpha'\gamma') = \gamma^{-1}\alpha^{-1}\alpha'\gamma' = \gamma^{-1}d(\alpha, \alpha')\gamma' = \gamma^{-1}d(\beta, \beta')\gamma' = d(\beta\gamma, \beta'\gamma')$

◀

B Infinitary group

▶ **Lemma 26.** *Let $\mathbb{S}$ be an infinitary group and Γ be a finite subset of S. For all $\alpha, \alpha', \beta, \beta' \in \mathbb{S}$ such that $d(\alpha, \alpha') \neq d(\alpha\beta, \alpha'\beta')$, for all non negative integers m, there is N such that for all $n \geqslant N$,*

$$d(\alpha\beta^n, \alpha'\beta'^n) \notin \Gamma^{\leqslant m}$$

Proof. Suppose that for some non negative integers $n \neq p$, $d(\alpha\beta^n, \alpha'\beta'^n) = d(\alpha\beta^p, \alpha'\beta'^p)$. Then by using the second item of Lemma 2, for all non negative integers i, $d(\alpha\beta^{n+i}, \alpha'\beta'^{n+i}) = d(\alpha\beta^{p+i}, \alpha'\beta'^{p+i})$, and thus $|\{d(\alpha\beta^n, \alpha'\beta'^n) \mid n \in \mathbb{N}\}|$ would be finite, that contradicts the hypothesis. Thus, for all non negative integers $n \neq p$, $d(\alpha\beta^n, \alpha'\beta'^n) \neq d(\alpha\beta^p, \alpha'\beta'^p)$, and finally, for all $m \geqslant 0$, since $\Gamma^{\leqslant m}$ is finite, there is N such that for all $n \geqslant N$,

$$d(\alpha\beta^n, \alpha'\beta'^n) \notin \Gamma^{\leqslant m}$$

◀

C Equivalent definition of TP_k

We give here another definition of the twinning property of order k and prove that the two definitions are equivalent.

A weighted automaton satisfies TP'_k if:
- for all $m \geqslant k$,
- for all states $\{q_i^j \mid i \in \{0, \dots, m\}, j \in \{0, \dots, k\}\}$ with q_0^j initial and q_m^j co-accessible for all j,
- for all words $u_1, \dots, u_m$, $v_1, \dots, v_m$ such that there are $k+1$ runs like described in the Figure 4,

then there are $j \neq j'$ such that for all $i \in \{1, \dots, m\}$:

$$d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'}) = d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j \beta_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'} \beta_i^{j'})$$

▶ **Lemma 27.** *For all positive integer k, a weighted automaton satisfies TP_k if and only if it satisfies TP'_k.*

Proof. By definition, TP'_k implies TP_k. For the converse implication, suppose that TP'_k is not satisfied. Then, there are:

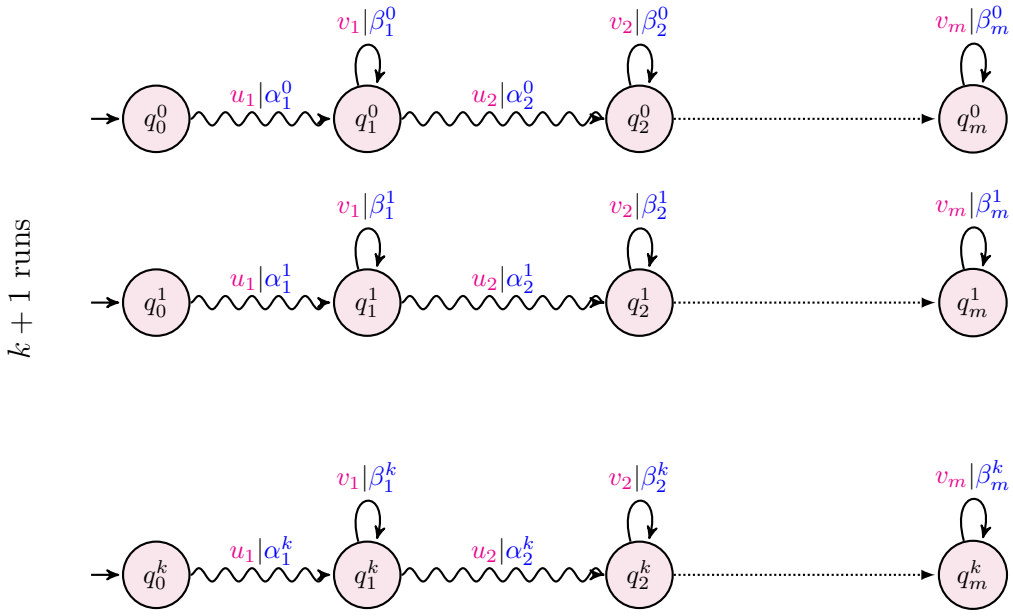

Figure 4 Equivalent definition of the twinning property of order k

- an integer $m \geqslant k$,
- words $u_1, \dots u_m$, $v_1, \dots, v_m$,
- states $\{q_i^j \mid i \in \{0, \dots, m\}, j \in \{0, \dots, k\}\}$ with q_0^j initial and q_m^j co-accessible for all j,
- $k+1$ runs like described in Figure 4,

such that for all $j \neq j'$, there is $i \in \{1, \dots, m\}$ satisfying:

$$d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'}) = d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j \beta_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'} \beta_i^{j'})$$

We prove now that we can only consider k loops and still preserve the property. For all $i \in \{0, \dots, m\}$, we construct a partition P_i of the set $\{0, \dots, k\}$.

- $P_0 = \{\{0, \dots, k\}\}$,
- P_{i+1} is a refinement of P_i such that j and j' remains in the same class if and only if $d(\alpha_1^j \alpha_2^j \cdots \alpha_{i+1}^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_{i+1}^{j'}) = d(\alpha_1^j \alpha_2^j \cdots \alpha_{i+1}^j \beta_{i+1}^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_{i+1}^{j'} \beta_{i+1}^{j'})$

By hypothesis, we know that P_m is the set of singleton sets. Moreover, since the partitioned set contains $k+1$ elements, there are at most k indices $i \in \{1, \dots, m\}$ such that $P_{i-1} \neq P_i$. Let us note them $i_1 < \ldots < i_k$. Then for all $j \neq j'$, consider i_s the smallest index such that j and j' are not is the same class in P_{i_s}. Then $d(\alpha_1^j \alpha_2^j \cdots \alpha_{i_s}^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_{i_s}^{j'}) \neq d(\alpha_1^j \alpha_2^j \cdots \alpha_{i_s}^j \beta_{i_s}^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_{i_s}^{j'} \beta_{i_s}^{j'})$, that proves that TP_k is not satisfied and concludes the proof. ◀

D Lemmas about TP_k

▶ **Lemma 10**. If W satisfies TP_k then for all words w, for all initial states $q^0, \dots, q^k$ and co-accessible states $p^0, \dots p^k$ such that there are $k+1$ runs:

$$q^j \xrightarrow{w|\alpha^j} p^j \quad \text{for all } j \in \{0, \dots, k\},$$

there are $j \neq j'$ such that $d(\alpha^j, \alpha^{j'}) \in \Gamma$.

Proof. If W satisfies TP_k then it also satisfies TP'_k. Let w be a word, $q^0, \dots, q^k$ initial states and $p^0, \dots p^k$ co-accessible states, such that:

$$q^j \xrightarrow{w|\alpha^j} p^j \quad \text{for all } j \in \{0, \dots, k\}$$

We will now extract synchronised loops in these runs. If $|w| \leqslant |Q|^{k+1}$ then for all $j \neq j'$, $d(\alpha^j, \alpha^{j'}) \in \Gamma$, otherwise there are:

- a positive integer $m \geqslant k$,
- words $u_1, \dots, u_m$, $v_1, \dots, v_m$ such that $w = u_1 v_1 \cdots u_m v_m$ and $|u_1 \cdots u_m| \leqslant |Q|^{k+1}$,
- states q_i^j for $i \in \{0, \dots, m\}$ and $j \in \{0, \dots, k\}$ such that $q_0^j = q^j$ and $q_k^j = p^j$,
- elements of $\mathbb{S}$, α_i^j, β_i^j for $i \in \{1, \dots, m\}$ and $j \in \{0, \dots, k\}$ such that $\alpha^j = \alpha_1^j \beta_1^j \cdots \alpha_m^j \beta_m^j$,

such that there are $k+1$ runs like described in Figure 4.

By TP'_k, there are $j \neq j'$ such that for all $i \in \{1, \dots, m\}$,

$$d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'}) = d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j \beta_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'} \beta_i^{j'})$$

Thus, by using at each step the item 2 of Lemma 2,

$$\begin{aligned} d(\alpha^j, \alpha^{j'}) &= d(\alpha_1^j \beta_1^j \alpha_2^j \cdots \alpha_m^j \beta_m^j, \alpha_1^{j'} \beta_1^{j'} \alpha_2^{j'} \cdots \alpha_m^{j'} \beta_m^{j'}) \\ &= d(\alpha_1^j \alpha_2^j \cdots \alpha_m^j \beta_m^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_m^{j'} \beta_m^{j'}) \qquad (\text{since } \; d(\alpha_1^j, \alpha_1^{j'}) = d(\alpha_1^j \beta_1^j, \alpha_1^{j'} \beta_1^{j'})) \\ &\vdots \\ &= d(\alpha_1^j \alpha_2^j \cdots \alpha_m^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_m^{j'}) \in \Gamma \qquad (\text{since } \; |u_1 \cdots u_m| \leqslant |Q|^{k+1}) \end{aligned}$$

◀

▶ **Lemma 11**. If W does not satisfy TP_k, then for all positive integers m, there is a word w, initial states $q^0, \dots, q^k$, co-accessible states $p^0, \dots, p^k$ and $k+1$ runs:

$$q^j \xrightarrow{w|\alpha^j} p^j \text{ for all } j \in \{0, \dots, k\},$$

such that for all $j \neq j'$, $d(\alpha^j, \alpha^{j'}) \notin \Delta_W^{\leqslant m}$.

Proof. Let m be a positive integer. Since W does not satisfy TP_k, then there are:

- states $\{q_i^j \mid i, j \in \{0, \dots, k\}\}$ with q_0^j initial and q_k^j co-accessible for all j,
- words $u_1, \dots, u_k$, $v_1, \dots, v_k$ such that there are $k+1$ runs like described in the Figure 2,

such that for all $j \neq j'$, there are $i \in \{1, \dots, k\}$ satisfying:

$$d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'}) \neq d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j \beta_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'} \beta_i^{j'})$$

We construct by induction (in decreasing order) a sequence of positive integers $t_k, \dots, t_1$. Let us give the construction of t_i. Let L be the length of the word $u_{i+1} v_{i+1}^{t_{i+1}} \cdots u_k v_k^{t_k}$. Consider all the pairs (j, j') such that:

$$d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'}) \neq d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j \beta_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'} \beta_i^{j'})$$

Thanks to Lemma 26, we can choose an integer n (the same for all such pairs (j, j')) such that $d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j (\beta_i^j)^n, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'} (\beta_i^{j'})^n) \notin \Delta_W^{\leqslant 2L+m}$. Let us set $t_i = n$.

Then, the word $w = u_1 v_1^{t_1} \cdots u_i v_i^{t_i} \cdots u_k v_k^{t_k}$ fulfils the conclusions of the Lemma: Let $j \neq j'$, and i the minimal index such that

$$d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'}) \neq d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j \beta_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'} \beta_i^{j'})$$

By using the second item of Lemma 2 at each steps before index i, one gets:

$$\begin{aligned}
& d(\alpha_1^j (\beta_1^j)^{t_1} \alpha_2^j \cdots \alpha_k^j (\beta_k^j)^{t_k}, \alpha_1^{j'} (\beta_1^{j'})^{t_1} \alpha_2^{j'} \cdots \alpha_k^{j'} (\beta_k^{j'})^{t_k}) \\
= \; & d(\alpha_1^j \alpha_2^j \cdots \alpha_{i-1}^j \alpha_i^j (\beta_i^j)^{t_i} \cdots \alpha_k^j (\beta_k^j)^{t_k}, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_{i-1}^{j'} \alpha_i^{j'} (\beta_i^{j'})^{t_i} \cdots \alpha_k^{j'} (\beta_k^{j'})^{t_k}) \\
= \; & (\alpha_{i+1}^j (\beta_{i+1}^j)^{t_{i+1}} \cdots \alpha_k^j (\beta_k^j)^{t_k})^{-1} \\
& \qquad d(\alpha_1^j \alpha_2^j \cdots \alpha_{i-1}^j \alpha_i^j (\beta_i^j)^{t_i}, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_{i-1}^{j'} \alpha_i^{j'} (\beta_i^{j'})^{t_i}) \\
& \qquad\qquad (\alpha_{i+1}^{j'} (\beta_{i+1}^{j'})^{t_{i+1}} \cdots \alpha_k^{j'} (\beta_k^{j'})^{t_k}) \\
\notin \; & \Delta_W^{\leqslant m}
\end{aligned}$$

◀

▶ **Lemma 28.** *If W satisfies TP_k then for all words w, for all runs $\rho^1, \ldots, \rho^{k+1}$ from an initial state to a co-accessible state, labelled by w, there are $j \neq j'$ such that $\rho^j, \rho^{j'}$ are Γ-close.*

Proof. The ideas are similar to the one given in Lemma 10. Consider $k+1$ runs $\rho^1, \ldots, \rho^{k+1}$ labelled by w from an initial state to a co-accessible state and suppose that for all $j \neq j'$, $\rho^j, \rho^{j'}$ are not Γ-close.

Then we can construct $k+1$ runs $\rho_1^j \overline{\rho}_1^j \rho_2^j \overline{\rho}_2^j \cdots \rho_n^j \overline{\rho}_n^j \rho_{n+1}^j$ where $n = k(k-1)$ such that:

- for all i, j, $\overline{\rho}_i^j$ is a loop (start and end in the same state),
- for all j, $\rho^j = \rho_1^j \rho_2^j \cdots \rho_n^j \rho_{n+1}^j$,
- for all $j \neq j'$, there is i such that

$$d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \alpha_i^{j'}) \neq d(\alpha_1^j \alpha_2^j \cdots \alpha_i^j \overline{\alpha}_i^j, \alpha_1^{j'} \alpha_2^{j'} \cdots \overline{\alpha}_i^{j'})$$

where α_j^i (resp. $\overline{\alpha}_j^i$) is the weight of the run ρ_j^i (resp. $\overline{\rho}_j^i$).

These $k+1$ runs give a counter example of TP_k. ◀

▶ **Lemma 13**. If W satisfies TP_k then for all n, for all words w, for all runs $\rho^1, \ldots, \rho^n$ from an initial state to a co-accessible state, labelled by w, there is a subset $P \subseteq \{1, \ldots, n\}$ containing k elements such that for all $j' \in \{1, \ldots, n\}$, there is $j \in P$ such that $\rho^j, \rho^{j'}$ are Γ-close.

Proof. It is a direct application of Lemma 28. ◀

E Relating TP_k and finite-valued weighted automata

▶ **Proposition 29.** *A ℓ-valued weighted automaton with n states satisfies $TP_{n\ell}$.*

Proof. Suppose that a weighted automaton W does not satisfy $TP_{n\ell}$, then from Lemma 11, for $m = n+1$, there is a word w, initial states $q^0, \ldots, q^{n\ell}$, co-accessible states $p^0, \ldots, p^{n\ell}$ and $n\ell + 1$ runs:

$$q^j \xrightarrow{w|\alpha^j} p^j \text{ for all } j \in \{0, \ldots, n\ell\}$$

such that for all $j \neq j'$, $d(\alpha^j, \alpha^{j'}) \notin \Delta_W^m$. Among the states $p^0, \ldots, p^{n\ell}$, at least $\ell + 1$ are the same one. Thus, the corresponding runs can be completed in accepting runs using the

same word (of length less than n). The delay between the values of two such runs (labelled by the same word) is necessarily different from 1: for all $j \neq j'$, $d(\alpha^j, \alpha^{j'}) \notin \Delta_W^{n+1}$ so by completing with a word of length less than n, the delay necessarily does not belong to Δ_W (either $1 \in \Delta_W$, either 1 does not belong to any Δ_W^m for all m) and so is different from 1. Finally, there are $\ell + 1$ different values for the same word, that contradicts the fact that W is ℓ-valued. ◀

▶ **Proposition 30.** *A weighted automaton satisfying TP_k for some natural k is finitely valued.*

Proof. Consider the accepting runs labelled by a given word w and their values $\alpha^0, \dots, \alpha^m$. By Lemma 13, there is a subset $P \subseteq \{0, \dots, m\}$ with k elements such that for all $j' \in \{0, \dots, m\}$, there is $j \in P$ such that $d(\alpha^j, \alpha^{j'}) \in \Gamma$. Since $\alpha^{j'} = \alpha^j d(\alpha^j, \alpha^{j'})$ then a value of an accepting run labelled by w is of the form $\alpha\beta$ where α can take k values and β belongs to Γ. So the automaton is $k|\Gamma|$-valued. ◀

F Proof of Section 4.1

▶ **Theorem 31.** *If $\mathbb{S}$ is an infinitary group, the two following assertions are equivalent:*

- *The automaton W does not satisfy TP_k.*
- *There is an integer $N > 0$ and a subset E of $\mathbb{S}$ satisfying (C_f) such that for all positive integers n, there are $w_0, \dots, w_k \in A^*$, $\alpha^0, \dots, \alpha^k \in \mathbb{S}$ such that:*
  - *for all $0 \leqslant i, j \leqslant k$, $dist(w_i, w_j) \leqslant N$,*
  - *for all $0 \leqslant i \leqslant k$, $\alpha^i \in f(w_i)$,*
  - *and for all $i \neq j$, $d(\alpha^i, \alpha^j) \notin E^{\leqslant n}$.*

Proof. First, suppose that W does not satisfy TP_k. Then, by applying Lemma 11, for all positive integers r, there is a word w, initial states $q^0, \dots, q^k$, co-accessible states $p^0, \dots, p^k$ and $k + 1$ runs:

$$q^j \xrightarrow{w|\beta^j} p^j \text{ for all } j \in \{0, \dots, k\},$$

such that for all $j \neq j'$, $d(\beta^j, \beta^{j'}) \notin \Delta_W^{\leqslant r}$. We can complete these $k + 1$ runs into accepting runs with words $u_0, \dots, u_k$ such that for all i, $|u_i| \leqslant |Q|$. Let us write $w_i = wu_i$ and denote by $\alpha^0, \dots, \alpha^k$ the weights of these runs. Since for all $j \neq j'$, $d(\beta^j, \beta^{j'}) \notin \Delta_W^{\leqslant r}$ and $|u_i| \leqslant |Q|$, then $d(\alpha^j, \alpha^{j'}) \notin \Delta_W^{\leqslant r-2|Q|}$. Take $r \geqslant n + 2|Q|$ to obtain the expected result with $E = \Delta_W$ and $N = 2|Q|$.

Conversely, suppose that there is an integer $N > 0$ and a set E satisfying (C_f) such that for all positive integers n, there are $w_0, \dots, w_k \in A^*$, $\alpha^0, \dots, \alpha^k \in \mathbb{S}$ such that:

1. for all $0 \leqslant i, j \leqslant k$, $dist(w_i, w_j) \leqslant N$,
2. for all $0 \leqslant i \leqslant k$, $\alpha_i \in f(w_i)$,
3. and for all $i \neq j$, $d(\alpha^i, \alpha^j) \notin E^{\leqslant n}$.

Since $\Delta_W^{\leqslant 2|Q|^{k+1}+2N}$ is finite then there is an integer n such that $(\Delta_W^{\leqslant 2|Q|^{k+1}+2N} \cap \bigcup_{n \geqslant 0} E^n) \subseteq E^{\leqslant n}$.

By item 2., there are $k + 1$ accepting runs in W labelled respectively by $w_0, \dots, w_k$ with values $\alpha^0, \dots, \alpha^k$. They induce $k+1$ runs labelled by w, where w denote the longest common prefix of the w_i, starting in an initial state and ending in a co-accessible state. Let us denote by $\beta^0, \dots \beta^k$ their values.

By contradiction, suppose now that there is $i \neq j$ such that $d(\beta^i, \beta^j) \in \Gamma$. Then, by item 1. there is $i \neq j$ such that $d(\alpha^i, \alpha^j) \in \Delta_W^{\leqslant 2|Q|^{k+1}+2N}$. But $d(\alpha^i, \alpha^j) \in \bigcup_{m \geqslant 0} E^m$. Thus, $d(\alpha^i, \alpha^j) \in E^{\leqslant n}$. This is a contradiction with item 3.

Thus, for all $i \neq j$, $d(\beta^i, \beta^j) \notin \Gamma$. By Lemma 10, this implies that W does not satisfy TP_k. ◀

G Proof of Section 4.2

G.1 $\mathcal{RA}_\ell(k) \subseteq \mathcal{WA}_\ell(k)$

We suppose that $\mathbb{S}$ is an infinitary group. We fix two positive integers k and ℓ. Given $R = (Q, q_{init}, R, \delta, \mu)$ a cost register automaton with k registers and output size ℓ, we want to construct an equivalent ℓ-valued weighted automaton $W = (Q', Q_{init}, Q_{final}, t, T)$ satisfying TP_k.

The idea is that the states of W will represent couples of a state and of a register of R. There will be a run labelled by w in W from an initial state ending in a state (p, X) if there is a run in R labelled by w ending in p. Moreover, the current value of the run in (p, X) will be the value stored in the register X after reading w in R.

More formally, we set:

- $Q' = (Q \times R) \cup (Q \times \{1, \dots, \ell\})$,
- $Q_{init} = \{q_{init}\} \times R$,
- $Q_{final} = Q \times \{1, \dots, \ell\}$,
- for all $q \in Q$, we arbitrarily choose an injective function $\tau_q : \mu(q) \to \{1, \dots, \ell\}$ and we set $t(q, \tau_q(Y, \beta)) = \beta$ if it is defined and 1 otherwise,
- The set of transitions is defined as the union of two sets $T = T_1 \cup T_2$ where:

$$T_1 = \{((p, X), a, \alpha, (q, Y)) \mid p, q \in Q,\ X, Y \in R,\ \delta(p, a) = (q, g) \text{ with } g(Y) = (X, \alpha)\}$$

and

$$T_2 = \{((p, X), a, \alpha, (q, f_q(Y, \beta))) \mid \delta(p, a) = (q, g),\ g(Y) = (X, \alpha),\ (Y, \beta) \in \mu(q)\}$$

The automata R and W compute the same function: Let w be a word. Let $\alpha \in \mathbb{S}$ an output associated to w by R. Let $(q_{init}, \nu_{init}), \dots, (q, \nu)$ the accepting run in R on w. Then, there is $Y \in R$ such that $\alpha = \nu(Y)\beta$ where $(Y, \beta) \in \mu(q)$. By definition, there is a register X such that the value of Y in q depends on the value of X in q_{init}. By construction, there is a run in W from (q_{init}, X) to $(q, \nu_q(Y, \beta))$ labelled by w. This run is accepting and with output $\nu(Y)\beta = \alpha$.

Conversely, let $\alpha \in \mathbb{S}$ an output associated to w by W. Then there is an accepting run from (q_{init}, X) to (q, i) labelled by w with output α. By construction, there is an accepting run going from (q_{init}, ν_{init}) to (q, ν) in R labelled by w, for some ν, and a register Y, such that $\alpha = \nu(Y)\beta$ where $(Y, \beta) \in \mu(q)$ and $\tau_q(Y, \beta) = i$. Thus, one of the output (the one associated to (Y, β)) is α.

The automaton W is ℓ-valued: It is a direct consequence of the fact that R and W compute the same function: R is of output size ℓ, thus every word has at most ℓ values and so W is ℓ-valued.

The automaton W satisfies TP_k: By contradiction, if the automaton does not satisfy TP_k, then by using Lemma 11, for all positive integers m, there is a word w, initial states $q^0, \dots, q^k$ and co-accessible states $p^0, \dots, p^k$ and $k+1$ runs:

$$q^j \xrightarrow{w|\alpha^j} p^j \text{ for all } j \in \{0, \dots, k\}$$

such that for all $j \neq j'$, $d(\alpha^j, \alpha^{j'}) \notin \Delta_W^{\leqslant m}$.

By construction, there is $p \in Q$, such that for all j, there is $X_j \in R$ or $i_j \in \{1, \dots, \ell\}$ such that $p^j = (p, X_j)$ or $p^j = (p, i_j)$. Since we are considering $k+1$ runs, there are two different runs such that the last transitions come from the same state (q, Y). Thus, there are $j \neq j'$ such that we are in one the two following cases:

- $p^j = (p, X_j)$, $p^{j'} = (p, i_{j'})$ and there is $\alpha \in \mathbb{S}$ such that $\tau_p(X_j, \alpha) = i_{j'}$.
- $p^j = (p, i_j)$, $p^{j'} = (p, i_{j'})$ and there are $X \in R$ and $\alpha, \beta \in \mathbb{S}$ such that $\tau_p(X, \alpha) = i_j$ and $\tau_p(X, \beta) = i_{j'}$.

Let n be an integer such that for all $q \in Q$, $X \in R$, $\alpha \in \mathbb{S}$ such that $(X, \alpha) \in \mu(q)$, $\alpha \in \Delta_W^{\leqslant n}$. By construction, the delay between α^j and $\alpha^{j'}$ belongs to Δ_W^{2n}. Thus by considering an m greater than $2n$, we have a contradiction.

G.2 $\mathcal{WA}_{\ell\text{-amb}}(k) \subseteq \mathcal{RA}_\ell(k)$

Consider a ℓ-ambiguous weighted automaton $W = (Q, Q_{init}, Q_{final}, t, T)$ satisfying TP_k. We will construct a cost register automaton $R = (Q', q_{init}, R, \delta, \mu)$ with k registers and output size ℓ computing the same function.

Recall that Θ denote the set of elements of $\mathbb{S}$ occurring on the transitions of W, $\Delta_W = \Theta \cup \Theta^{-1}$ and $\Gamma = \Delta_W^{2|Q|^{k+1}}$.

The idea behind the construction is to store in the states of R the delays between the values computed in the states of W reached by runs labelled by a given word. We will use the fact that there are at most k diverging behaviours (thanks to TP_k) to prove that we can store the delays up to a bound and use only k registers to capture the k diverging behaviours.

Construction of the set of states Q'.

Let Ω denote the set of functions from $(Q \times \{1, \dots, \ell\})^2$ to $\Gamma \cup \{\infty, \bot\}$ such that for all $x, y \in Q \times \{1, \dots, \ell\}$:

- $f(x, y) = f(y, x)^{-1}$ (with the convention $\infty^{-1} = \infty$ and $\bot^{-1} = \bot$),
- $f(x, x) \in \{1, \bot\}$,
- if $f(x, x) = \bot$, then $f(x, y) = f(y, x) = \bot$,
- if $f(x, x) = f(y, y) = 1$ then $f(x, y) \neq \bot$.

Given f and g in Ω, we say that f *is equivalent to* g (denoted by $f \equiv g$) if for all $q \in Q$, there is a permutation σ_q of $\{1, \dots, \ell\}$ such that for all $p, q \in Q$, $1 \leqslant i, j \leqslant \ell$,

$$g((q, i), (p, j)) = f((q, \sigma_q(i)), (p, \sigma_p(j)))$$

We denote by $\sigma_q^{(f,g)}$ this permutation if it exists. Remark that $\equiv$ is an equivalence relation. Let $\overline{f}$ denote the equivalence class of f.

The set of states Q' is defined as the set of the equivalence classes of $\equiv$.

Initial state and registers.

Let f_{init} be the function in Ω defined by $f(x, y) = \mathbf{1}$ if $x, y \in Q_{init} \times \{1\}$ and $f(x, y) = \bot$ otherwise. The initial state q_{init} is $\overline{f_{init}}$. The set of registers is $R = \{X_1, \dots, X_k\}$.

Transition function.

Let a be a letter. We will define the transitions labelled by a.

For all $p \in Q$, $f \in \Omega$, set:

$$E_p^{(f,a)} = \{(q,i) \mid \text{ there exist } (q,a,\alpha,p) \in T \text{ and } f((q,i),(q,i)) = 1\}$$

Remark that $|E_p^{(f,a)}| = |E_p^{(g,a)}|$ if $f \equiv g$. If $|E_p^{(f,a)}| > \ell$ then the transition from $\overline{f}$ labelled by a is undefined.

Otherwise, suppose that $|E_p^{(f,a)}| \leqslant \ell$, and consider a one-to-one function $\tau_p^{(f,a)} : E_p^{(f,a)} \to \{1,\dots,\ell\}$. If $g \equiv f$, then we can choose $\tau_p^{(g,a)}$ and $\tau_p^{(f,a)}$ such that $\tau_p^{(g,a)}(q,i) = \tau_p^{(f,a)}(q,\sigma_q^{(f,g)}(i))$. Fix now such functions $\tau_p^{(f,a)}$ compatible over the equivalence classes (when there are defined).

Let f_a be the function defined by:

$$f_a((p',i'),(q',j')) = \begin{cases} \alpha^{-1} f((p,i),(q,j))\beta & \text{if } \tau_{p'}^{(f,a)}(p,i) = i',\ \tau_{q'}^{(f,a)}(q,j) = j', \\ & \quad (p,a,\alpha,p') \in T,\ (q,a,\beta,q') \in T \\ & \quad \text{and } \alpha^{-1} f((p,i),(q,j))\beta \in \Gamma \\ \infty & \text{if } \tau_{p'}^{(f,a)}(p,i) = i',\ \tau_{q'}^{(f,a)}(q,j) = j', \\ & \quad (p,a,\alpha,p') \in T,\ (q,a,\beta,q') \in T \\ & \quad \text{and } \alpha^{-1} f((p,i),(q,j))\beta \notin \Gamma \cup \{\bot\} \\ \bot & \text{otherwise} \end{cases}$$

with the convention that for all $x \in \mathbb{S} \cup \{\infty\}$, $x\infty = \infty x = \infty$, and for all $x \in \mathbb{S} \cup \{\infty, \bot\}$, $x\bot = \bot x = \bot$.

Since $f \in \Omega$, f_a also belongs to Ω. Moreover, if $g \equiv f$ then $g_a \equiv f_a$ and more precisely, $g_a((p,i),(q,j)) = f_a((p,\sigma_p^{(f_a,g_a)}(i)),(q,\sigma_q^{(f_a,g_a)}(j))$.

Given $f \in \Omega$, we define Z_f the set of functions:

$$r_f : \{1,\dots,k\} \to Q \times \{1,\dots,\ell\}$$

such that:

- for all $\kappa \in \{1,\dots,k\}$, $f(r_f(\kappa), r_f(\kappa)) \neq \bot$,
- and for all $x \in Q \times \{1,\dots,\ell\}$ satisfying $f(x,x) \neq \bot$, there is $\kappa \in \{1,\dots,k\}$ such that $f(r_f(\kappa), x) \notin \{\infty, \bot\}$.

If $f \equiv g$ then $Z_f \neq \varnothing$ if and only if $Z_g \neq \varnothing$. Moreover, if $r_f \in Z_f$ then the function r_g defined by $r_g(\kappa) = (p, \sigma_p^{f,g}(i))$ if $r_f(\kappa) = (p,i)$ belongs to Z_g. For all $f \in \Omega$ such that $Z_f \neq \varnothing$, we fix now a function $r_f \in Z_f$ such that the choices are compatible over the equivalence classes.

If Z_f or Z_{f_a} are empty then we define (it will never be the case for accessible $\overline{f}$): $\delta(\overline{f}, a) = (\overline{f_a}, h)$ for an arbitrary chosen function h.

Suppose now that Z_f and Z_{f_a} are not empty. Given $\kappa' \in \{1,\dots,k\}$, let $(q',i') = r_{f_a}(\kappa')$ and q, i, α such that $\tau_{q'}^{(f,a)}(q,i) = i'$ and $(q,a,\alpha,q') \in T$. By property of r_f, there is $\kappa \in \{1,\dots,k\}$ such that $f(r_f(\kappa),(q,i)) \notin \{\infty,\bot\}$. We set $\beta_{\kappa'}^{(f,a)} = f(r_f(\kappa),(q,i))\alpha$. If $g \equiv f$ then $\beta_\kappa^{(g,a)} = \beta_\kappa^{(f,a)}$.

In this case, the transition is defined by:

$$\delta(\overline{f}, a) = (\overline{f_a}, h) \text{ with for all } \kappa' \in \{1,\dots,k\},\ h(X_{\kappa'}) = (X_\kappa, \beta_{\kappa'}^{(f,a)})$$

Output function.

For $x \in Q_{final} \times \{1, \dots, \ell\}$ such that $f(x,x) = 1$, let $\kappa_x^{(f)} \in \{1, \dots, k\}$ such that $f(r_f(\kappa_x^{(f)}), x) = \alpha_x \notin \{\infty, \bot\}$. Set also $\beta_x = t(q)$ where $x = (q,i)$. If $g \equiv f$ then we can make the choices before such that $\kappa_{(q,i)}^{(g)} = \kappa_{(q,\sigma_q^{(f,g)}(i))}^{(f)}$.

Let E_f be a maximal set of elements $x \in Q_{final} \times \{1, \dots, \ell\}$ such that $f(x,x) = 1$ and for all $x, y \in E_f$, $f(x,y) \neq 1$. The output function is defined by:

$$\mu(\overline{f}) = \{(X_{\kappa_x}, \alpha_x \beta_x) \mid x \in E_f\}$$

By construction, the automaton R uses k registers.

Automata W and R compute the same function. Let us note $Q = \{q_1, \dots, q_{|Q|}\}$. We suppose that all states are co-accessible.

Given a word w, denote by $\rho_i^1, \dots, \rho_i^{\ell_i}$ the runs labelled by w from an initial state to q_i. Remark that $\ell_i \leqslant \ell$ since W is ℓ-ambiguous. Let us note $\alpha_i^1, \dots, \alpha_i^{\ell_i}$ their respective weights.

▶ **Lemma 32.** *If there is i such that $\ell_i \neq 0$ then there is an accepting run in R labelled by w. Moreover, there is f such that this run ends in $\overline{f}$ such that:*

$$f((q_i,\kappa),(q_{i'},\kappa')) = \begin{cases} d(\alpha_i^\kappa, \alpha_{i'}^{\kappa'}) & \text{if } \kappa \leqslant \ell_i, \kappa' \leqslant \ell_{i'}, \rho_i^\kappa, \rho_{i'}^{\kappa'} \Gamma - \text{close} \\ \infty & \text{if } \kappa \leqslant \ell_i, \kappa' \leqslant \ell_{i'}, \rho_i^\kappa, \rho_{i'}^{\kappa'} \text{ not } \Gamma - \text{close} \\ \bot & \text{otherwise} \end{cases}$$

Proof. Suppose that there is $\ell_i \neq 0$. The proof is made by induction on the length of the word w. If $w = \varepsilon$ then there is an accepting run in R on w ending in $\overline{f_{init}}$ and f_{init} satisfies the condition.

Suppose now that $w = w'a$ for some letter a. If there is a run from an initial state labelled by w, there is also a run from an initial state labelled by w'. Let us denote by $\tau_i^1, \dots, \tau_i^{\ell_i'}$ the runs on w' ending in q_i and by $\beta_i^1, \dots, \beta_i^{\ell_i'}$ their respective weights. By induction hypothesis, there is an accepting run in R labelled by w' ending in $\overline{f}$ such that:

$$f((q_i,\kappa),(q_{i'},\kappa')) = \begin{cases} d(\beta_i^\kappa, \beta_{i'}^{\kappa'}) & \text{if } \kappa \leqslant \ell_i', \kappa' \leqslant \ell_{i'}', \tau_i^\kappa, \tau_{i'}^{\kappa'} \Gamma - \text{close} \\ \infty & \text{if } \kappa \leqslant \ell_i', \kappa' \leqslant \ell_{i'}', \tau_i^\kappa, \tau_{i'}^{\kappa'} \text{ not } \Gamma - \text{close} \\ \bot & \text{otherwise} \end{cases}$$

By ℓ-ambiguity, for all p, $|E_p^{(g,a)}| \leqslant \ell$. Thus, by construction, there is an accepting run labelled by w in R ending in $\overline{f_a}$. Moreover,

$$f_a((q_j,m),(q_{j'},m')) = \begin{cases} \alpha^{-1} f((q_i,\kappa),(q_{i'},\kappa'))\beta & \text{if } \tau_{q_j}^{(f,a)}(q_i,\kappa) = m,\ \tau_{q_{j'}}^{(f,a)}(q_{i'},\kappa') = m', \\ & (q_i,a,\alpha,q_j) \in T,\ (q_{i'},a,\beta,q_{j'}) \in T \\ & \text{and } \alpha^{-1} f((q_i,\kappa),(q_{i'},\kappa'))\beta \in \Gamma \\ \infty & \text{if } \tau_{q_j}^{(f,a)}(q_i,\kappa) = m,\ \tau_{q_{j'}}^{(f,a)}(q_{i'},\kappa') = m', \\ & (q_i,a,\alpha,q_j) \in T,\ (q_{i'},a,\beta,q_{j'}) \in T \\ & \text{and } \alpha^{-1} f((q_i,\kappa),(q_{i'},\kappa'))\beta \notin \Gamma \cup \{\bot\} \\ \bot & \text{otherwise} \end{cases}$$

If $\kappa \leqslant \ell_i$, $\kappa' \leqslant \ell_{i'}$, $\rho_i^\kappa, \rho_{i'}^{\kappa'}$ Γ-close, then we are in the first case and $f_a((q_j,m),(q_{j'},m')) = \alpha^{-1} f((q_i,\kappa),(q_{i'},\kappa'))\beta = \alpha^{-1} d(\beta_i^\kappa, \beta_{i'}^{\kappa'})\beta = d(\alpha_j^m, \alpha_{j'}^{m'})$. If $\kappa \leqslant \ell_i$, $\kappa' \leqslant \ell_{i'}$, $\rho_i^\kappa, \rho_{i'}^{\kappa'}$ not Γ-close then we are in the second case and $f_a((q_j,m),(q_{j'},m')) = \infty$. Otherwise, we are in the third case and $f_a((q_j,m),(q_{j'},m')) = \bot$. ◀

▶ **Lemma 33.** *If for all i, $\ell_i = 0$ then either there is no accepting run on w in R, or the unique accepting run ends in $\overline{f}$ where f maps all the pairs to $\bot$.*

Proof. The proof is made by induction. If $w = \varepsilon$, it means that there is no initial state in W, and thus the property is satisfied since f_{init} maps all the pairs to $\bot$. Suppose now that $w = w'a$ for some letter a.

The first possibility is that there is no run in W on w'. Thus by induction hypothesis, either there is no accepting run on w' in R, and then there is no accepting run on w in R, or the unique accepting run ends in $\overline{f}$ where f maps all the pairs to $\bot$, and thus if there is an accepting run in R on w, it also ends in $\overline{f}$ (since $f_a = f$ in this case).

The second possibility is that there are runs in W on w' ending in a set of states F, but no transition labelled by a starting in F. By using Lemma 32, there is a run on w' ending in $\overline{g}$ with g satisfying the conditions of Lemma 32. By construction of g_a, since there is no transition from F labelled by a, then g_a is the function that maps all the pairs to $\bot$. ◀

▶ **Lemma 34.** *If $\overline{f}$ is an accessible state in R then r_f is well-defined.*

Proof. It is a direct consequence of Lemma 32 and Lemma 13. ◀

▶ **Lemma 35.** *Suppose that there is an accepting run on w in R to $(\overline{f}, \nu)$, then for all $1 \leqslant \kappa \leqslant k$, $\nu(X_\kappa) = \alpha_i^j$ such that $r_f(\kappa) = (q_i, j)$.*

Proof. The proof is made by induction on the length of w. If $w = \varepsilon$ then for all $1 \leqslant \kappa \leqslant k$, $\nu(X_\kappa) = 1$ and the property is satisfied. Otherwise, suppose that $w = w'a$ for some letter a. Then there is an accepting run on w' in R to some $(\overline{g}, \nu)$ such that $g_a = f$ and $\nu(X_{\kappa'}) = \nu'(X_\kappa)\beta_{\kappa'}^{(g,a)} = \alpha_i^j$ by induction hypothesis, the construction of the transitions and Lemma 32. ◀

Let us finally prove that W and R compute the same function. Consider an accepting run on w in W of weight α ending in some final state q. Then by Lemma 32, there is an accepting run in R to some $(\overline{f}, \nu)$ such that there is $x = (q, i)$ for some i corresponding to this run. By definition of r_f, there is κ such that $f(r_f(\kappa), (q, i)) \in \Gamma$. Moreover, by Lemma 35, $\alpha t(q) = \nu(X_\kappa) f(r_f(\kappa), (q, i)) t(q)$. Finally, by the construction of the output function, $(X_\kappa, f(r_f(\kappa), (q, i))t(q))$ belongs to $\mu(\overline{f})$ and thus $\alpha t(q)$ is associated with w by R. The converse is similar, if R associates some value α with w, then W also associates α with w.

R is of output size ℓ. Consider an accessible state $\overline{f}$ of R. Pairs (q, i) such that $f((q, i), (q, i)) = 1$ corresponds to runs starting in an initial state and ending in q labelled by a same word. Moreover, thanks to Lemma 32, if $f((q, i), (q', i')) \neq 1$, then the two corresponding runs must be different. By ℓ-ambiguity of W, there is at most ℓ accepting runs labelled by a given word. Thus, $|E_f| \leqslant \ell$ and R is of output size ℓ.

H Hierarchy

▶ **Theorem 36.** *For all positive integers ℓ, k,*

- $\mathcal{WA}_\ell = \mathcal{RA}_\ell$
- $\mathcal{WA}(k) = \mathcal{RA}(k)$

Proof. *First item:*

- $\mathcal{WA}_\ell \subseteq \mathcal{RA}_\ell$: Consider a ℓ-valued weighted automaton with n states. By Proposition 29, the function computed belongs to $\mathcal{WA}_\ell(n\ell)$, that is equal to $\mathcal{RA}_\ell(n\ell)$ by Theorem 16. Thus, $\mathcal{WA}_\ell \subseteq \cup_{n \in \mathbb{N} - \{0\}} \mathcal{RA}_\ell(n\ell) \subseteq \mathcal{RA}_\ell$.

- $\mathcal{RA}_\ell \subseteq \mathcal{WA}_\ell$: Consider a function in $\mathcal{RA}_\ell$. It is computed by a register automaton of output size ℓ, with k registers for some k, thus belongs to $\mathcal{RA}_\ell(k)$. Finally, it belongs to $\mathcal{WA}_\ell(k)$ by Theorem 16 and thus to $\mathcal{WA}_\ell$.

Second item:

- $\mathcal{RA}(k) \subseteq \mathcal{WA}(k)$: Consider a function computed by a register automaton with k registers. Let ℓ denote its output size. Then the function belongs to $\mathcal{RA}_\ell(k) = \mathcal{WA}_\ell(k)$ by Theorem 16 and finally to $\mathcal{WA}(k)$.
- $\mathcal{WA}_k^B \subseteq \mathcal{RA}(k)$: Consider a weighted automaton satisfying TP_k, then by Proposition 30, it is finitely valued, and the function it computes belongs to $\mathcal{WA}_\ell(k)$ for some ℓ. Finally, $\mathcal{WA}_\ell(k) = \mathcal{RA}_\ell(k) \subseteq \mathcal{RA}(k)$ by Theorem 16.

◀

I Proofs of Section 5

▶ **Proposition 18**.

$$\mathcal{RA}_\ell(k) \cap \mathcal{F} \subseteq \mathcal{RA}_\ell^B(k)$$

Proof. Let $(Q, q_{init}, R, \delta, \mu)$ denote a register automaton over $\mathbb{S} = (B \cup B^{-1})^*$ computing a function $f \in \mathcal{F}$.

Set E the minimal subset of $Q \times R$ such that:

- $(q, X) \in E$ if there is $\alpha \in \mathbb{S}$ such that $(X, \alpha) \in \mu(q)$,
- $(q, X) \in E$ if there is $(p, Y) \in E$, $a \in A$ such that $\delta(q, a) = (p, h)$ for some h such that $h(Y) = (X, \alpha)$ for some $\alpha \in \mathbb{S}$.

We say that a register X is *alive* in q if $(q, X) \in E$.

▶ **Lemma 37.** *There is a non negative integer N such that for all configurations (q, ν), for all runs from (q_{init}, ν_{init}) to (q, ν), for all alive registers X in q, $\nu(X)$ belongs to $B^*(B \cup B^{-1})^{\leqslant N}$.*

Proof. Consider the set of outputs occurring on the updates of the registers:

$$\{\alpha \in \mathbb{S} \mid \delta(q, a) = (p, h), h(Y) = (X, \alpha), q, p \in Q, a \in A, X, Y \in R\}$$

This set is finite, thus there is an integer m such that it is included in $(B \cup B^{-1})^{\leqslant m}$.

Consider also the set of outputs occurring in the output function:

$$\{\alpha \in \mathbb{S} \mid (X, \alpha) \in \mu(q), X \in R, q \in Q\}$$

This set is finite, thus there is an integer s such that it is included in $(B \cup B^{-1})^{\leqslant s}$.

Set $N = |Q|m + s$. Consider a run from (q_{init}, ν_{init}) to (q, ν) and an alive register X in q such that $\nu(X) = c \in \mathbb{S}$. The run can be completed in a run that ends in some (q', ν') such that there are a register Y, $\alpha \in (B \cup B^{-1})^{\leqslant |Q|m}$, $\beta \in (B \cup B^{-1})^{\leqslant s}$ with $\nu'(Y) = c\alpha$, $(Y, \beta) \in \mu(q')$ and thus $c\alpha\beta \in B^*$. Finally, $c \in B^*(B \cup B^{-1})^{\leqslant N}$. ◀

We construct now an equivalent register automaton R' over B^*. Set $\mathcal{G}$ the set of functions $R \to (B \cup B^{-1})^{\leqslant N}$ where N is the integer given in Lemma 37.

The set of states of R' is $Q' = Q \times \mathcal{G}$. The initial state is (q_{init}, r_{init}) where r_{init} is the function that associates each register to $\mathbf{1}$ and the set of registers is R.

For $\alpha \in B^*(B \cup B^{-1})^{\leqslant N}$, let us denote by α_1 and α_2 the two elements such that $\alpha = \alpha_1\alpha_2$ and α_1 is the shortest element of B^* such that $\alpha_2 \in (B \cup B^{-1})^{\leqslant N}$. Remark that α_1 and α_2 always exist in this case.

The transition function δ' is defined in the following way: given $q \in Q$, $a \in A$, let $(p, g) = \delta(q, a)$. We set: $\delta'((q, r), a) = ((p, t), h)$ where $h(Y) = (X, (r(X)\alpha)_1)$, $t(Y) = (r(X)\alpha)_2$ for $X, Y \in R$ such that $g(Y) = (X, \alpha)$.

The output function μ' associates a state (q, r) to the set $\{(X, r(X)\alpha) \mid (X, \alpha) \in \mu(q)\}$.

The cost register automaton R' has, by construction, the same number of registers and the same output size as R. Let us prove now that they compute the same function.

▶ **Lemma 38.** *For all words w, there is a run in R on w from (q_{init}, ν_{init}) to some (q, ν) if and only if there is a run in R' on w from $((q_{init}, r_{init}), \nu_{init})$ to $((q, r), \sigma)$ such that for all registers X, $\nu(X) = \sigma(X)r(X)$.*

Proof. The proof is made by induction on the length of w. By construction the property holds for $w = \varepsilon$. Suppose now that $w = w'a$ for some $a \in A$.

Suppose that there is a run in R on w from (q_{init}, ν_{init}) to (q, ν). Set (q', ν') the configuration such that there is a run in R on w' from (q_{init}, ν_{init}) to (q', ν') and $\delta(q', a) = (q, g)$. By induction hypothesis, there is a run in R' on w' from $((q_{init}, r_{init}), \nu_{init})$ to $((q', r'), \sigma')$ such that for all registers X, $\nu'(X) = \sigma'(X)r'(X)$. Moreover, by construction, $\delta'((q', r'), a) = ((q, r), h)$ where $h(Y) = (X, (r'(X)\alpha)_1)$, $r(Y) = (r'(X)\alpha)_2$ for $X, Y \in R$ such that $g(Y) = (X, \alpha)$. The values $(r'(X)\alpha)_1$ and $(r'(X)\alpha)_2$ are well-defined thanks to Lemma 37. Thus there is a run in R on w from $((q_{init}, r_{init}), \nu_{init})$ to $((q, r), \sigma)$ with $\sigma(Y)r(Y) = \sigma'(X)(r'(X)\alpha)_1(r'(X)\alpha)_2$ for $X, Y \in R$ such that $g(Y) = (X, \alpha)$. Thus, $\sigma(Y)r(Y) = \nu'(X)\alpha = \nu(Y)$.

Conversely, suppose that there is a run in R' on w from $((q_{init}, r_{init}), \nu_{init})$ to $((q, r), \sigma)$. Set $((q', r'), \sigma')$ the configuration such that there is a run in R' on w' from $((q_{init}, r_{init}), \nu_{init})$ to $((q', r'), \sigma')$ and $\delta'((q', r'), a) = ((q, r), h)$. Then by induction hypothesis, there is a run in R on w' from (q_{init}, ν_{init}) to (q', ν') such that for all registers X, $\nu'(X) = \sigma'(X)r'(X)$. Moreover, by construction, $\delta(q, a) = (q, g)$ and if $g(Y) = (X, \alpha)$ then $h(Y) = (X, (r'(X)\alpha)_1)$, $r(Y) = (r'(X)\alpha)_2$ for $X, Y \in R$. The values $(r'(X)\alpha)_1$ and $(r'(X)\alpha)_2$ are well-defined thanks to Lemma 37. Thus, there is a run in R on w from (q_{init}, ν_{init}) to (q, ν) with $\nu(Y) = \nu'(X)\alpha = \sigma'(X)r'(X)\alpha = \sigma'(X)(r'(X)\alpha)_1(r'(X)\alpha)_2 = \sigma(Y)r(Y)$. ◀

Finally, let us prove that $[\![R]\!](w) = [\![R']\!](w)$. Consider a run in R on w from (q_{init}, ν_{init}) to (q, ν) and $(Y, \alpha) \in \mu(q)$. Then, by Lemma 38, there is a run in R' on w from $((q_{init}, r_{init}), \nu_{init})$ to $((q, r), \sigma)$ for some σ such that $\nu(Y) = \sigma(Y)r(Y)$. Thus, $\nu(Y)\alpha = \sigma(Y)r(Y)\alpha$ and by construction, $(Y, r(Y)\alpha) \in \mu'(q, r)$.

Conversely, suppose that there is a run in R' on w from $((q_{init}, r_{init}), \nu_{init})$ to $((q, r), \sigma)$ and a register Y such that $(Y, r(Y)\alpha) \in \mu'(q, r)$. Then by Lemma 38, there is a run in R on w from (q_{init}, ν_{init}) to (q, ν) such that $\nu(Y) = \sigma(Y)r(Y)$. Thus, $\sigma(Y)r(Y)\alpha = \nu(Y)\alpha$ and by construction, $(Y, \alpha) \in \mu(q)$. ◀