A Generalised Twinning Property for Minimisation of Cost Register Automata

Laure Daviaud
LIP, CNRS, École Normale Supérieure de Lyon, INRIA,
Université Claude-Bernard Lyon 1
laure.daviaud@ens-lyon.fr

Pierre-Alain Reynier Jean-Marc Talbot
Aix Marseille Université, CNRS, LIF UMR 7279
pierre-alain.reynier@lif.univ-mrs.fr
jean-marc.talbot@lif.univ-mrs.fr

Abstract

Weighted automata (WA) extend finite-state automata by associating with transitions weights from a semiring $\mathbb{S}$, defining functions from words to $\mathbb{S}$. Recently, cost register automata (CRA) have been introduced as an alternative model to describe any function realised by a WA by means of a deterministic machine. Unambiguous WA over a monoid $(M, \otimes)$ can equivalently be described by cost register automata whose registers take their values in M, and are updated by operations of the form $x := y \otimes c$, with $c \in M$. This class is denoted by $\mathsf{CRA}_{\otimes c}(M)$.

We introduce a twinning property and a bounded variation property parametrised by an integer k, such that the corresponding notions introduced originally by Choffrut for finite-state transducers are obtained for $k = 1$. Given an unambiguous weighted automaton W over an infinitary group $(G, \otimes)$ realizing some function f, we prove that the three following properties are equivalent: $i)$ W satisfies the twinning property of order k, $ii)$ f satisfies the k-bounded variation property, and $iii)$ f can be described by a $\mathsf{CRA}_{\otimes c}(G)$ with at most k registers.

In the spirit of tranducers, we actually prove this result in a more general setting by considering machines over the semiring of finite sets of elements from $(G, \otimes)$: the three properties are still equivalent for such finite-valued weighted automata, that is the ones associating with words subsets of G of cardinality at most ℓ, for some natural ℓ. Moreover, we show that if the operation $\otimes$ of G is commutative and 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.

Keywords weighted automata, cost register automata, minimisation, twinning property

1. Introduction

Weighted automata. 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 (Chatterjee et al. 2010). Weighted automata (Schützenberger 1961) (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.

For automata based models, a very important problem is to simplify the models. For instance, deterministic machines are 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. In general, not every WA can be transformed into an equivalent deterministic one. The determinisability problem then asks, given a WA on some semiring $(S, \oplus, \otimes)$, whether there exists an equivalent deterministic WA over $(S, \oplus, \otimes)$. This problem ranges from trivial to undecidable, depending on the considered semiring, see (Lombardy and Sakarovitch 2006) for a survey. Regarding size reduction, the problem is usually to minimize the number of states of deterministic instances. Algorithms that, given a weighted automaton, minimize the number of states of an equivalent deterministic one are given in (Mohri 2000) and (Maletti 2008).

Cost register automata. Recently, a new model of machine, named cost register automata (CRA), has been introduced in (Alur et al. 2013). 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 functions $x \mapsto x \otimes c$ for all c in S) compute the same functions (Alur et al. 2013). 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 operations, 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 (Alur and Raghothaman 2013).

CRA are deterministic by definition, and the main minimisation problem is captured by the notion of register complexity. It is defined for a function f as the minimal integer ℓ such that f can be defined by a CRA with ℓ registers. Computing the register complexity is a highly challenging problem that has been addressed in a recent paper for the particular case of ACRA (Alur and Raghothaman 2013), using ad-hoc techniques.

Transducers. Transducers 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 ℓ 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 (Choffrut 1977), 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 is important to notice that the bounded variation property is a machine independent characterisation. It has been first shown in (Weber and Klemm 1995) that the twinning property is decidable in PTIME. Recently, a so-called weak twinning property has been introduced in (Jecker and Filiot 2015). This property characterises finite state transducers recognising finite-valued relations that can be expressed as a finite union of deterministic transducers.

Finite-valued weighted automata. As several problems are undecidable for general weighted automata, it is important to identify large subclasses with decidability results. Finite-valued transducers enjoy good properties, such as effective transformation into finitely ambiguous transducers (see (Sakarovitch and de Souza 2010) for an elegant proof), and as a consequence a decidable equivalence problem. 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. ℓ-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 ℓ). It is finite valued if it is ℓ-valued for some ℓ. These definitions allow to transfer several positive results from transducers to weighted automata: unambiguous and functional WA are equivalent and the determinisability problem is decidable for functional weighted automata over infinitary groups (Filiot et al. 2015). These results rely on the twinning property originally introduced by Choffrut for transducers. Similarly, it has been proven in (Filiot et al. 2014) that finite-valued weighted automata over infinitary groups can effectively be decomposed as finite union of functional weighted automata, using a generalisation of the construction presented in (Sakarovitch and de Souza 2010) for transducers.

Contributions. Our objective is to identify the register complexity for cost register automata in the class $\mathsf{CRA}_{\otimes c}(S)$. More precisely, we study the register minimisation problem that aims, given a CRA and an integer k, at deciding whether there exists an equivalent CRA with only k registers.

We start with the framework of functional weighted automata and cost register automata with weights taken in some group $\mathbb{G}$. Given an natural number k, we introduce a twinning property of order k and a k-bounded variation property, such that the original notions of Choffrut are obtained for $k = 1$. We then prove the following: let W be a functional weighted automaton on an infinitary group realizing some partial function f from words to elements of $\mathbb{G}$, then the three following properties are equivalent: $i)$ W satisfies the twinning property of order k, $ii)$ f satisfies the k-bounded variation property, and $iii)$ f can be defined by a $\mathsf{CRA}_{\otimes c}(\mathbb{G})$ with k registers. This constitutes a strong equivalence between three properties with different roles: $i)$ is a property expressed by means of a pattern on a weighted automaton, that can be used to derive efficient decision procedures; $ii)$ is a machine independent characterisation; $iii)$ corresponds to the class that we want to characterize, and thus allows us to minimise the number of registers. We actually first prove the equivalence between $i)$ and $ii)$, thus showing that the twinning property is machine independent, and then use this result together with automata constructions to prove the equivalence between $i)$ and $iii)$. Note also that our constructions are effective: given a weighted automaton satisfying the twinning property of order k, we effectively build an equivalent cost register automaton with k registers, and conversely.

Our result holds actually in a more general setting, that of finite-valued weighted automata. To prove this, we consider a generalisation of cost register automata in which the final output function may produce a subset of the register's values, and denote by $\mathsf{CRA}^{+}_{\otimes c}(\mathbb{G})$ the resulting class. Using an adequate generalisation of the k-bounded variation to relations instead of functions, we prove that our equivalence still holds.

Beyond weighted automata over infinitary groups, we also prove that our results apply also to 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 (Alur and Cerný 2010), and that the class $\mathsf{CRA}_{.c}(B^*)$ 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 T realizing a relation R satisfies the twinning property of order k iff R satisfies the k-bounded variation property iff R can be expressed by a $\mathsf{CRA}^{+}_{.c}(B^*)$ with at most k registers.

Regarding decidability, we show that if the group $\mathbb{G}$ is commutative and has a computable internal operation, then the twinning property of order k is decidable. As a particular instance of our decision procedure, we obtain that this property can be decided in PSPACE for $\mathbb{G} = (\mathbb{Z}, +, 0)$. As a corollary, we obtain that the problem of register minimisation is PSPACE-complete for $\mathsf{CRA}^{+}_{+c}(\mathbb{Z})$ hence generalising the result of (Alur and Raghothaman 2013) for ACRA. We also prove that the twinning property of order k is decidable in PSPACE for finite-state transducers, and as a corollary the problem of register minimisation is PSPACE-complete for $\mathsf{CRA}^{+}_{.c}(B^*)$. This result entails that the register complexity of any finite-valued rational relations, given as a cost register automata with updates of the form $x := yu$ with x, y some registers and u a finite word on B, can be computed.

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 Section 4 and present its proof. In order to simplify the presentation, Sections 3 and 4 are presented in the setting of finitely generated groups, and we discuss in Section 5 how to lift our results to arbitrary groups. We turn to transducers in Section 6. Last we present our decidability results and their application to the register minimisation problem in Section 7. 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.

For a set S, we denote by $|S|$ the cardinality of 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 $\alpha \in M$, 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 to be *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 $(\alpha \oplus \beta) \otimes \gamma = (\alpha \otimes \gamma) \oplus (\beta \otimes \gamma)$).

Given a set S, the set of the finite subsets of S is denoted by $\mathcal{P}_{\mathsf{fin}}(S)$. For a monoid $\mathbb{M}$, the set $\mathcal{P}_{\mathsf{fin}}(M)$ equipped with the two operations $\cup$ (union of two sets) and the set extension of $\otimes$ is a semiring denoted $\mathbb{P}_{\mathsf{fin}}(\mathbb{M})$.

From now on, we may identify 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 $\mathit{dist}(u, v) = |u| + |v| - 2 * |\mathit{lcp}(u, v)|$ where $\mathit{lcp}(u, v)$ is the longest common prefix of u and v.

When considering groups, we use the following 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 $delay(\alpha, \beta)$.*

For a finitely generated group $\mathbb{G}$, with a fixed finite set of generators Γ, one can define a distance between two elements derived from the Cayley graph of $(\mathbb{G}, \Gamma)$. We consider here an undirected right Cayley graph : given $\alpha \in \mathbb{G}$, $\beta \in \Gamma$, there is a (non-oriented) edge between α and $\alpha\beta$.

Definition 2. *Let $\mathbb{G}$ be a finitely generated group and Γ be a finite set of generators. Given $\alpha,\ \beta \in \mathbb{G}$, the Cayley distance between α and β is the length of the shortest path linking α and β in the undirected right Cayley graph of $(\mathbb{G}, \Gamma)$. It is denoted by $d(\alpha, \beta)$.*

Lemma 1. *Given a group $\mathbb{G}$, for all $\alpha, \alpha', \beta, \beta', \gamma, \gamma' \in \mathbb{G}$,*

1. *$delay(\alpha, \beta) = \mathbf{1}$ if and only if $\alpha = \beta$,*
2. *if $delay(\alpha, \alpha') = delay(\beta, \beta')$ then $delay(\alpha\gamma, \alpha'\gamma') = delay(\beta\gamma, \beta'\gamma')$.*

Given a finitely generated group $\mathbb{G}$ and a finite set of generators Γ, for all $\alpha, \beta \in \mathbb{G}$, $d(\alpha, \beta) = d(\mathbf{1}, delay(\alpha, \beta))$.

Definition 3. *A group $\mathbb{G}$ is said to be infinitary if for all $\alpha, \beta, \gamma \in \mathbb{G}$ such that $\alpha\beta\gamma \neq \beta$, the set $\{\alpha^n\beta\gamma^n \mid n \in \mathbb{N}\}$ is infinite.*

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 (Filiot et al. 2015).

Given a finite alphabet B, we denote by $\mathcal{F}(B)$ the free group over B. It is well known that it is finitely generated (with B as finite set of generators) and that it is infinitary.

Weighted automata. Given a semiring $\mathbb{S}$, weighted automata (WA) are non-deterministic finite automata in which transitions have for weights elements of $\mathbb{S}$. WA 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 $\mathbb{P}_{\mathsf{fin}}(\mathbb{M})$, formally defined as follows:

Definition 4. *Let A be a finite alphabet, a weighted automaton W over some monoid $\mathbb{M}$ is a tuple $(Q, Q_{\mathit{init}}, Q_{\mathit{final}}, t, T)$ where Q is a finite set of states, $Q_{\mathit{init}} \subseteq Q$ is the set of initial states, $Q_{\mathit{final}} \subseteq Q$ is the set of final states, $t : Q_{\mathit{final}} \to \mathbb{M}$ 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 $\mathbb{M}$, $\alpha = \alpha_1\alpha_2 \cdots \alpha_k$. We depict this situation as $q_1 \xrightarrow{w|\alpha} q_{k+1}$. A state q is *accessible* (*resp. co-accessible*) if there exists such a run with $q_1 \in Q_{\mathit{init}}$ and $q = q_{k+1}$ (*resp.* $q_{k+1} \in Q_{\mathit{final}}$ and $q = q_1$). The run ρ is said to be accepting if $q_1 \in Q_{\mathit{init}}$ and $q_{k+1} \in Q_{\mathit{final}}$. Moreover we say that the value of such an accepting run is $\alpha t(q_{k+1})$. W.l.o.g., we assume that every state of W is accessible and co-accessible.

This automaton W computes the (total) function $[\![W]\!] : A^* \to \mathcal{P}_{\mathsf{fin}}(\mathbb{M})$ 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).

Given a weighted automaton $W = (Q, Q_{\mathit{init}}, Q_{\mathit{final}}, t, T)$ over some finitely generated group $\mathbb{G}$ with finite set of generators Γ, we define the constant M_W as follows:

$$M_W = \max\{d(\mathbf{1}, \alpha) \mid \exists (p, a, \alpha, q) \in T\}$$

Definition 5 (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 weighed automaton is unambiguous if it is 1*-ambiguous.*

Obviously, an ℓ-ambiguous WA is an ℓ-valued automaton. $\mathcal{WA}_\ell$ (resp. $\mathcal{WA}_{\ell\text{-amb}}$) denotes the set of functions $A^* \to \mathcal{P}_{\mathsf{fin}}(\mathbb{M})$ computed by ℓ-valued (resp. ℓ-ambiguous) weighted automata. A weighted automaton is said to be *finitely valued* (resp. *finitely ambiguous*) if it is ℓ-valued (resp. ℓ-ambiguous) for some natural ℓ.

Theorem 1 ((Filiot et al. 2014))**.** *Let W be an ℓ-valued weighted automaton over some infinitary group $\mathbb{G}$, then one can effectively build an equivalent ℓ-ambiguous weighted automaton over $\mathbb{G}$.*

Example 1. *Let us consider $A = \{a, b\}$ and $(\mathbb{M}, \otimes, \mathbf{1}) = (\mathbb{Z}, +, 0)$. The weighted automaton given in Figure 1 (above) 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. A cost register automaton (CRA) (Alur et al. 2013) is a deterministic automaton with registers containing values from a set S and that are updated through the transitions: for each register, its new value is computed from the old ones and from elements of S combined using some operations over S. 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 S.

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 updates are unary and are of the form $x := y \otimes c$, where $c \in \mathbb{M}$ and x, y are registers. When $\mathbb{M}$ is $(\mathbb{Z}, +, 0)$, this

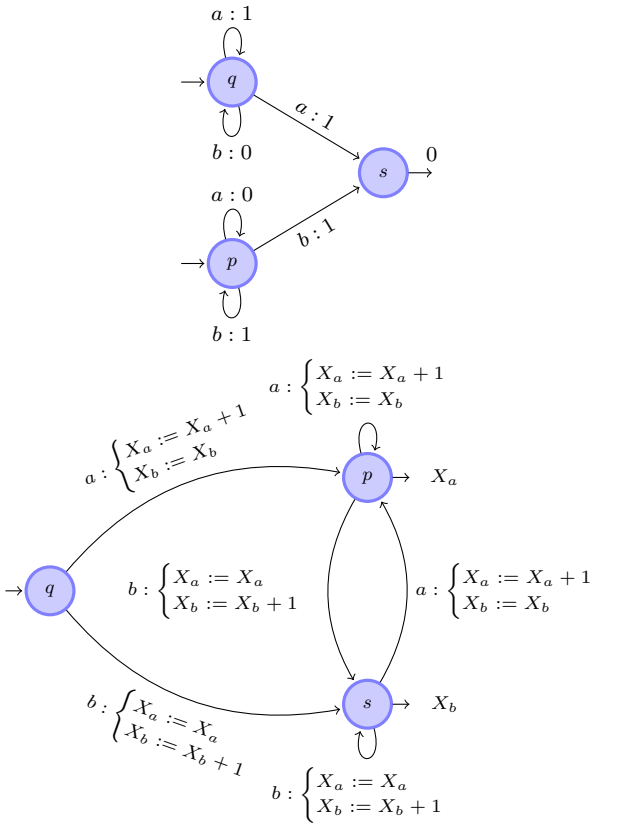

Figure 1. Examples of a weighted automaton (above) and of a cost register automaton (below).

class of automata is called additive cost register automata (Alur and Raghothaman 2013). When $\mathbb{M}$ is the free monoid $(A^*, ., \varepsilon)$, this class is a subclass of streaming string transducers (Alur and Cerný 2010) and turns out to be equivalent to the class of rational functions on words, *i.e.* those realized by finite-state transducers.

While cost register automata introduced in (Alur et al. 2013) define functions from A^* to $\mathbb{M}$, we are interested in defining finite-valued relations. To this aim, we slightly modify the definition of CRA, allowing to produce a set of values computed from register contents. 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 $(\mathbb{M}, \otimes, \mathbf{1})$ 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, $\mu : Q \to \mathcal{P}_{\mathsf{fin}}(\mathcal{X} \times \mathbb{M})$ is the output function.*

The semantics of such an automaton is as follows: if an update function f labels 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 $1 \leqslant i \leqslant k$, and all registers Y, if $\delta(q_i, w_i) = (q_{i+1}, g_i)$ with $g_i(Y) = (X, \alpha)$, 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 the set of $\nu_{k+1}(X)\alpha$ such that $(q_1, \nu_1)(q_2, \nu_2) \dots (q_{k+1}, \nu_{k+1})$ is an accepting run of R on w 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 $\mathsf{CRA}^{\ell}_{\otimes c}(\mathbb{M})$. Remark that $\mathsf{CRA}^{+}_{\otimes c}(\mathbb{M})$ is defined as the union of $\mathsf{CRA}^{\ell}_{\otimes c}(\mathbb{M})$ over $\ell \geqslant 1$.

Let $\mathcal{CRA}_\ell(k)$ (*resp.* $\mathcal{CRA}(k)$, $\mathcal{CRA}_\ell$) denote the set of functions $A^* \to \mathcal{P}_{\mathsf{fin}}(\mathbb{M})$ 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).

Example 2. *Consider $A = \{a, b\}$ and $(\mathbb{M}, \otimes, \mathbf{1}) = (\mathbb{Z}, +, 0)$. The cost register automaton given in Figure 1 (below) computes the same function as the one computed by the weighted automaton from Example 1. 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 (Choffrut 1977) are obtained for $k = 1$.

From now on, we consider a finitely generated infinitary group $\mathbb{G}$ and we fix a finite set of generators Γ.

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 Cayley distance between these values is bounded).

Definition 8. *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 $0 \leqslant j \leqslant k$, for all $1 \leqslant i \leqslant k$, $q_{i-1,j} \xrightarrow{u_i|\alpha_{i,j}} q_{i,j}$ 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 $i \in \{1, \dots, k\}$,

$$delay(\alpha_{1,j} \cdots \alpha_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}) = delay(\alpha_{1,j} \cdots \alpha_{i,j}\beta_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}\beta_{i,j'}).$$

Example 3. *The weighted automaton given in Figure 1 does not satisfy TP$_1$. Indeed, consider $q_{0,0} = q_{1,0} = q$, $q_{0,1} = q_{1,1} = p$, $u_1 = \varepsilon$ and $v_1 = a$. Then, $delay(0, 0) = 0 \neq delay(1, 0) = 1$. One can however prove that it satisfies TP$_2$.*

3.2 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.

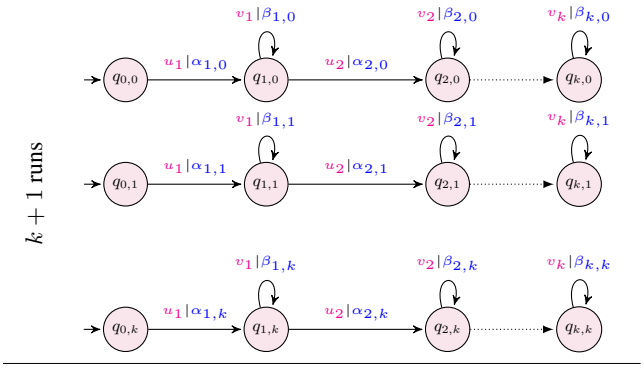

Figure 2. Twinning property of order k.

Given a partial mapping $f : A^* \rightharpoonup B^*$, the bounded variation property introduced by Choffrut in (Choffrut 1977) 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 $f : A^* \rightharpoonup \mathcal{P}_{\mathsf{fin}}(B^*)$ 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 finitely generated groups. Instead of $dist(,)$, we use the Cayley distance $d(,)$ to compare the images.

We present now our definition that generalises the original one of Choffrut (Choffrut 1977):

Definition 9. *Let $\mathbb{G}$ be a finitely generated group. A function $f : A^* \to \mathcal{P}_{\mathsf{fin}}(\mathbb{G})$ satisfies the k-bounded variation property if for all naturals $n > 0$, there is a natural N such that for all words $w_0, \dots, w_k \in A^*$ and all $\alpha_0 \in f(w_0), \dots, \alpha_k \in f(w_k)$, if for all $0 \leqslant i, j \leqslant k$, $dist(w_i, w_j) \leqslant n$ then there are $0 \leqslant i < j \leqslant k$ satisfying $d(\alpha_i, \alpha_j) \leqslant N$.*

Example 4. *We consider the weighted automaton depicted on Figure 1. This automaton is over the group $(\mathbb{Z}, +, 0)$ which is finitely generated with $\{1\}$ as a set of generators. The function f computed by the weighted automaton of Figure 1 does not satisfy the 1-bounded variation property. Indeed, set $n = 2$ and let N be a positive integer. Set $w_0 = a^{N+2}$ and $w_1 = a^{N+1}b$, then $dist(w_0, w_1) \leqslant 2$ but $d(f(w_0), f(w_1)) = d(N+2, 1) = N+1 > N$. One can however prove that the function f satisfies 2-bounded variation property.*

3.3 First properties on twinning and bounded variation properties

We can now state some properties on runs of weighted automata depending whether they satisfy TP_k or not.

In the following lemmas, $\mathbb{G}$ denotes some finitely generated group that is infinitary, and a finite set of generators Γ is fixed. Let W denote a weighted automaton on $\mathbb{G}$ and Q be its set of states.

Lemma 2. *The automaton W satisfies TP_k if and only if there is an integer N such that 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 $d(\alpha_j, \alpha_{j'}) \leqslant N$. In addition, if W satisfies TP_k then N can be chosen equal to $2M_W|Q|^{k+1}$.

We introduce a definition expressing the fact that two runs are *always* close (w.r.t. the Cayley distance):

Definition 10. *Let W be a weighted automaton on $\mathbb{G}$, $w = w_1 \dots w_r$ be a word and N be an integer. Let ρ_1, ρ_2 be two runs on w, with:*

$$\rho_j = q_{1,j} \xrightarrow{w_1|\alpha_{1,j}} q_{2,j} \cdots q_{r,j} \xrightarrow{w_r|\alpha_{r,j}} q_{r+1,j} \text{ for } j = 1, 2.$$

The runs ρ_1, ρ_2 are said to be N-close if for all $i \geq 1$,

$$d(\alpha_{1,1} \dots \alpha_{i,1}, \alpha_{1,2} \dots \alpha_{i,2}) \leqslant N.$$

Lemma 3. *Suppose that W satisfies 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 $2M_W|Q|^{k+1}$-close.*

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

Proposition 1. *A weighted automaton over an infinitary finitely generated group $\mathbb{G}$ satisfies TP_k for some natural k if and only if it is finitely valued.*

Sketch. First, if a weighted automaton W is ℓ-valued then it satisfies $TP_{n\ell}$ where n is its number of states. We proceed by contradiction. If this is not the case, then by Lemma 2, for all integer N, there are $n\ell + 1$ runs labelled by the same word with weights pairwise far, *i.e.* $d(\alpha, \beta) > N$ for every two such weights. Since n is the number of states of W, there exists a state q of W and $\ell + 1$ of these $n\ell + 1$ runs that end in state q. By completing these runs into accepting runs, and by taking N large enough, we get that these $\ell + 1$ accepting runs are labelled by the same word and have pairwise different values, which contradicts the ℓ-valuedness of W.

Conversely, consider a weighted automaton with n states that satisfies TP_k for some k. Consider the set of accepting runs labelled by some word w. By Lemma 3, one can extract a subset of k accepting runs such that any accepting run on w is $2M_W n^{k+1}$-close to one of these k runs. It implies that w can only have a finite number of values that depends on k, M_W and n. □

Equivalence between TP_k and the k-bounded variation property.

Proposition 2. *A weighted automaton W over an infinitary finitely generated group $\mathbb{G}$ satisfies TP_k if and only if $[\![W]\!]$ satisfies the k-bounded variation property.*

Sketch. First, if W does not satisfy TP_k then it does not satisfy the k-bounded variation property. Indeed, by applying Lemma 2, one can find $k+1$ runs labelled by the same word with arbitrarily large pairwise delays. By completing these runs into accepting runs, we obtain $k+1$ runs labelled by $k+1$ words that are pairwise close, but whose weights are pairwise arbitrarily far. This proves that $[\![W]\!]$ does not satisfy the k-bounded variation property.

Conversely, if the k-bounded variation property is not satisfied, then there exist $k+1$ words that are close and that label accepting runs whose values are pairwise arbitrarily far. By considering w the longest common prefix of these $k+1$ words, we get $k+1$ runs labelled by w that have weights pairwise far, yielding the result by Lemma 2. □

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

4.1 Main result

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 2. *Let W be an ℓ-valued weighted automaton over the semiring $\mathbb{P}_{\mathsf{fin}}(\mathbb{G})$ where $(\mathbb{G}, \otimes)$ is an infinitary finitely generated group, and k be a positive integer. 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, which can be effectively computed.*

We have already proved the equivalence between 1 and 2 of Theorem 2 in the previous section (Proposition 2). The equivalence between 1 and 3 is much more intricate. This equivalence follows from the following proposition, and uses the equivalence between 1 and 2.

Proposition 3. *Given an ℓ-valued weighted automaton W on some infinitary finitely generated group $\mathbb{G}$ satisfying the twinning property of order k, one can effectively build an equivalent $\mathsf{CRA}^{\ell}_{\otimes c}(\mathbb{G})$ with k registers, and conversely.*

Proof. (Sketch) Going from a cost register automaton to a weighted automaton is rather simple. The construction is similar to the one given in (Alur et al. 2013), and intuitively uses states of the form (p, X) where p (resp. X) denotes a state (resp. a register) of the cost register automaton. The resulting weighted automaton is trivially ℓ-valued, and one can easily verify that it satisfies the TP_k. Thanks to the equivalence of assertions 1 and 2 of Theorem 2 (the twinning property of order k is a machine independent property), we deduce that the weighted automaton W also satisfies the twinning property of order k.

We describe now the translation of a weighted automaton into a cost register automaton. We thus consider a weighted automaton W on some infinitary finitely generated group $\mathbb{G}$ satisfying the twinning property of order k. By Theorem 1, we can build an equivalent weighted automaton that is ℓ-ambiguous. Thanks to the equivalence of assertions 1 and 2 of Theorem 2 (the twinning property of order k is a machine independent property), we deduce that this weighted automaton satisfies the twinning property of order k.

Remind that Γ denotes a finite set of generators of $\mathbb{G}$ and that the Cayley distance is defined in $(\mathbb{G}, \Gamma)$. Consider an ℓ-ambiguous weighted automaton $W = (Q, Q_{\mathit{init}}, Q_{\mathit{final}}, t, T)$ satisfying TP_k, and let $N = 2M_W|Q|^{k+1}$ (recall that M_W is the maximum of the Cayley distances between $\mathbf{1}$ and the weights on the transitions of W). We build $R \in \mathsf{CRA}^{\ell}_{\otimes c}(\mathbb{G})$ with k registers computing the same function. We write $R = (Q', q_{\mathit{init}}, \mathcal{X}, \delta, \mu)$ with $\mathcal{X} = \{X_1, \ldots, X_k\}$.

The idea underlying the construction is to store in the states of R an abstraction of all the pairwise 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 (the bound N) and use only k registers to capture the k diverging behaviours.

Set of states of R. Consider a word w. For all co-accessible states $q \in Q$, by ℓ-ambiguity, there are at most ℓ runs from an initial state to q labelled by w. Let $\rho_{q,1}, \rho_{q,2}, \ldots, \rho_{q,\ell_q}$ denote these runs and $\alpha_{q,1}, \alpha_{q,2}, \ldots, \alpha_{q,\ell_q}$ denote their respective weights, with $\ell_q \leqslant \ell$. We construct R such that the unique run labelled by w in R ends in a state representing all the pairwise delays between $\alpha_{q,j}$ and $\alpha_{q',j'}$ up to the bound N.

In order to simplify the notations, we let $[\ell] = \{1, \ldots, \ell\}$. We use functions with domain $(Q \times [\ell])^2$ and range $(\Gamma \cup \Gamma^{-1})^{\leqslant N} \cup \{\infty, \bot\}$. Formally, the function representing the delays after reading word w is the following one: $f_w((q,j),(q',j')) =$

$$\begin{cases} \mathit{delay}(\alpha_{q,j}, \alpha_{q',j'}) & \text{if } j \leqslant \ell_q, j' \leqslant \ell_{q'}, (\rho_{q,j}, \rho_{q',j'}) \ N\text{-close} \\ \infty & \text{if } j \leqslant \ell_q, j' \leqslant \ell_{q'}, (\rho_{q,j}, \rho_{q',j'}) \text{ not } N\text{-close} \\ \bot & \text{otherwise (\textit{i.e.} one of the two runs is not defined)} \end{cases}$$

The functions f defined this way satisfy the following natural properties $(\star)$:

- $f(x,y) = f(y,x)^{-1}$ (with the convention $\infty^{-1} = \infty$ and $\bot^{-1} = \bot$), since $\mathit{delay}(\alpha,\beta) = \mathit{delay}(\beta,\alpha)^{-1}$,
- $f(x,x) \in \{\mathbf{1}, \bot\}$, since either x represents an existing run (and in this case we use the fact that $\mathit{delay}(\alpha,\alpha) = \mathbf{1}$), or not (and in this case the value $\bot$ is used).
- if $f(x,x) = \bot$, then $f(x,y) = f(y,x) = \bot$ (case where x does not represent an existing run)
- if $f(x,x) = f(y,y) = \mathbf{1}$ then $f(x,y) \neq \bot$ (case where both x and y represent existing runs).

Of course, the above representation depends on the numbering chosen for the runs of W. In order to avoid this, we introduce an equivalence relation $\equiv$ on such functions, based on permutations σ_q of $\{1, \ldots, \ell\}$ for each state q.

Let Ω denote the set of functions from $(Q \times [\ell])^2$ to $(\Gamma \cup \Gamma^{-1})^{\leqslant N} \cup \{\infty, \bot\}$ satisfying properties $(\star)$. The set of states of R is the set of $\equiv$-classes of Ω. Formally, given $f \in \Omega$, we denote by $[f]$ the equivalence class of f w.r.t. $\equiv$ and we let $Q' = \{[f] \mid f \in \Omega\}$. For every equivalence class C, we fix a representative function $g_C \in \Omega$.

More precisely, the cost register automaton R is constructed using a forward exploration starting from the initial state. The initial state is the class of the function $f_\varepsilon \in \Omega$ defined by $f_\varepsilon(x,y) = \mathbf{1}$ if $x, y \in Q_{\mathit{init}} \times \{1\}$ and $f_\varepsilon(x,y) = \bot$ otherwise. As a consequence, only reachable states are considered, *i.e.* states C such that there exists a word $w \in A^*$ satisfying $C = [f_w]$.

As W satisfies the twinning property of order k, Lemma 3 applies. This implies the following property: for every reachable state C of R, there exists a subset P_C of $Q \times [\ell]$ of size at most k such that for all $x \in Q \times [\ell]$, we have:

- either $g_C(x,x) = \bot$,
- or $g_C(x,x) = \mathbf{1}$ and there exists $y \in P_C$ such that $g_C(x,y) \in (\Gamma \cup \Gamma^{-1})^{\leqslant N}$.

For each reachable state C, we fix such a set P_C, a surjective mapping r_C from $\{1, \ldots, k\}$ to P_C, and a partial mapping $\chi_C : Q \times [\ell] \rightharpoonup P_C$ concretising the property stated in the second above item. Intuitively, if the function g_C is understood as a matrix, the set P_C corresponds to a subset of indices that allows to capture (at most) k runs with diverging behaviours. The mapping r_C simply gives a numbering of these indices and the mapping χ_C relates every other index to P_C. Let us point out that r_C is used to associate a register of $\mathcal{X}$ with each of these k runs, storing its computed value.

Transition function. Let C be a reachable state, and $a \in A$ be a letter, we define the state C' such that $\delta(C,a) = (C', h)$ with $h \in \mathcal{UP}(\mathcal{X})$. From the function g_C, one can easily define a function $f \in \Omega$ by extending the runs of W represented in g_C using the transitions of W on letter a. The state C' is then defined as $[f]$.

We now explain how the register update h is defined and illustrate it using the register X_1. Let $(q, j) = r_{C'}(1)$. By construction of $g_{C'}$, there exists a state p, and an index $i \in \{1, \dots, \ell\}$ such that the run represented by the pair (q, j) in $g_{C'}$ is of the following form:

$$Q_{init} \ni q_1 \xrightarrow{*} p \xrightarrow{a|\alpha} q$$

where the run $\rho : q_1 \xrightarrow{*} p$ is represented by the pair (p, i) in g_C.

We let $m \in \{1, \dots, k\}$ such that $r_C(m) = \chi_C(p, i)$, meaning that the weight of the run ρ can be recovered from the value of register X_m, and that register X_m corresponds in g_C to the run represented by the entry $r_C(m) \in Q \times [\ell]$. We finally define $h(X_1)$ as follows:

$$h(X_1) = (X_m, g_C(r_C(m), (p, i))\alpha)$$

Output function We describe how the output function μ is defined for a given reachable state C. Intuitively, we want to identify registers corresponding to a final state of W. However, it may be the case that two registers of $\mathcal{X}$ correspond to two accepting runs computing the same value. In order to respect the constraint on the output size of R, we let D as a maximal subset of $\{1, \dots, k\}$ such that:

- for all $m \in D$, we have $r_C(m) = (q, j) \in Q_{final} \times [\ell]$
- for all $m \neq m' \in D$, we have $g_C(r_C(m), r_C(m')) \neq \mathbf{1}$

The ℓ-ambiguity of W implies that the cardinality of D is at most ℓ. We have thus selected at most ℓ successful runs computing pairwise distinct values. We then define

$$\mu(C) = \{(X_m, \alpha) \mid m \in D, r_C(m) = (q, j), \text{ and } \alpha = t(q)\}.$$

This immediately implies that the output size of R is at most ℓ. □

4.2 Hierarchy

The previous 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.

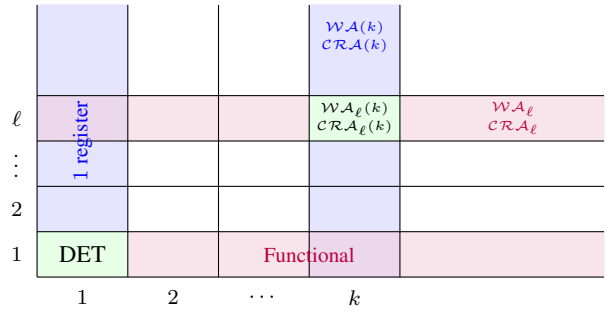

Figure 3. Hierarchy.

Theorem 3. *The following results hold:*

1. *For all naturals k, ℓ, $\mathcal{CRA}_\ell(k) = \mathcal{WA}_\ell(k)$.*
2. *For all naturals k, $\mathcal{CRA}(k) = \mathcal{WA}(k)$*
3. *For all naturals ℓ, $\mathcal{CRA}_\ell = \mathcal{WA}_\ell$*

Proof. The first item comes from Theorem 2.

As for the second item, by definition of a cost register automaton and first item, we have the following sequence of inclusions: $\mathcal{CRA}(k) = \cup_{\ell>0}\mathcal{CRA}_\ell(k) = \cup_{\ell>0}\mathcal{WA}_\ell(k) \subseteq \mathcal{WA}(k)$. It remains to prove that $\mathcal{WA}(k) \subseteq \cup_{\ell>0}\mathcal{WA}_\ell(k)$ to conclude. It comes from Proposition 1 since a weighted automaton satisfying TP_k for some k is finitely valued.

Finally, regarding the third item, still by definition of a cost register automaton and first item, we have the following sequence of inclusions: $\mathcal{CRA}_\ell = \cup_{k>0}\mathcal{CRA}_\ell(k) = \cup_{k>0}\mathcal{WA}_\ell(k) \subseteq \mathcal{WA}_\ell$. And $\mathcal{WA}_\ell \subseteq \cup_{k>0}\mathcal{WA}_\ell(k)$ by Proposition 1 since a weighted automaton that is finitely valued satisfies TP_k for some k. □

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

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

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 $\mathcal{CRA}_\ell(k)$, but not in the classes $\mathcal{CRA}_{\ell'}(k')$ for $k' < k$ or $\ell' < \ell$, showing that this hierarchy is strict.

5. The case of non finitely generated groups

In this section, we explain how to state our main result (Theorem 2) in the setting of non finitely generated groups. The twinning property only depends on the notion of delay, it is thus well-defined for arbitrary groups. However, we have to introduce a new notion of k-bounded variation property, as the one introduced in Definition 9 relies on the assumption that the group $\mathbb{G}$ is finitely generated.

Given a weighted automaton W on some group $\mathbb{G}$, the set of weights it may produce actually belongs to a finitely generated group, whose set of generators is the set of weights appearing on transitions of W. However, this observation is not sufficient, as the bounded variation property is intended to be machine-independent. In order to deal with this issue, we have introduced a notion of generator set for the computed function. More precisely, in the bounded variation property, in order to quantify the "distance" between the outputs, we introduce a notion of generator that depends on the function under study.

Given a mapping $f : A^* \to \mathcal{P}_{\text{fin}}(\mathbb{G})$ computed by a weighted automaton, we say that a subset Γ of $\mathbb{G}$ is an *f-generator* if $\{delay(\alpha, \alpha') \mid \alpha \in f(w), \alpha' \in f(w'), w, w' \in A^*\} \subseteq \bigcup_{n\geqslant 0} \Gamma^n$. Note that if $\mathbb{G}$ is finitely generated, then any finite set of generators of $\mathbb{G}$ is an f-generator.

This leads to a new definition of the bounded variation property that is still machine-independent and well defined for non finitely generated groups.

Definition 11. *A function $f : A^* \to \mathcal{P}_{\text{fin}}(\mathbb{G})$ satisfies the k-bounded variation property if for all naturals $n > 0$ and all f-generators Γ there is a natural N such that for all words $w_0, \dots, w_k \in A^*$ and all $\alpha_0 \in f(w_0), \dots, \alpha_k \in f(w_k)$, if for all $0 \leqslant i, j \leqslant k$, dist$(w_i, w_j) \leqslant n$ then there are $0 \leqslant i < j \leqslant k$ such that $delay(\alpha_i, \alpha_j) \in \Gamma^{\leqslant N}$.*

With this new definition, we can prove that Theorem 2 holds for all infinitary groups. Let us first comment the proof of the equivalence between statements 1 and 2 of Theorem 2. The direction from 2 to 1 holds true as the new definition of bounded variation implies the previous one, when considering the set of weights of W as a $[\![W]\!]$-generator. Conversely, one can prove a rephrasing of Lemma 2 that takes into account a $[\![W]\!]$-generator Γ, and replaces the constraint on the Cayley distance $d(\alpha_j, \alpha_{j'}) \leqslant N$ by the constraint $delay(\alpha_j, \alpha_{j'}) \in \Gamma^{\leqslant N}$. This new result can be used to adapt the proof of Proposition 2.

The equivalence between weighted automata satisfying TP_k and k-register automata follows the lines of the proof of Proposition 3 by considering the sub-group generated by the finite set of weights

occurring on the transitions of the automaton under consideration. This finite set plays the role of the finite set of generators and the Cayley distance is defined with respect to it.

6. The case of transducers

A transducer is defined as a weighted automaton with weights in the monoid B^*. It can thus be seen as a weighted automaton with weights in the free group $\mathcal{F}(B)$ (see Section 2 for a presentation of the free group). We say that a transducer T satisfies the twinning property of order k if, viewed as a weighted automaton over $\mathcal{F}(B)$, it satisfies TP_k. Similarly, a function $f : A^* \to \mathcal{P}_{\mathsf{fin}}(B^*)$ is said to satisfy the k-bounded variation property iff it is the case when viewing f as a mapping from A^* to $\mathcal{P}_{\mathsf{fin}}(\mathcal{F}(B))$.

Theorem 4. *Let T be an ℓ-valued transducer from A^* to B^*, and k be a positive integer. The following assertions are equivalent:*

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

The equivalence between 1 and 2 follows from Theorem 2. We now detail the proof of the equivalence between 1 and 3.

First, the implication $3 \Rightarrow 1$ is simple as the conversion from cost register automata to weighted automata preserves the weights used. We prove now the other direction.

Given a finite alphabet B, let $\mathcal{WA}^B_\ell(k)$ denote the set of functions computed by a ℓ-valued weighted automaton over the free group $\mathcal{F}(B)$ satisfying TP_k, but only using weights in B^*. Similarly, we denote by $\mathcal{CRA}^B_\ell(k)$ the set of functions computed by a cost register automaton over $\mathcal{F}(B)$ with k registers and output size ℓ, but only using updates involving elements of B^*. We thus have to prove the inclusion $\mathcal{WA}^B_\ell(k) \subseteq \mathcal{CRA}^B_\ell(k)$.

Let $\mathcal{G}_{A,B}$ denote the set of functions $A^* \to \mathcal{P}_{\mathsf{fin}}(B^*)$. By Theorem 2, we have the following sequence of inclusions:

$$\mathcal{WA}^B_\ell(k) \subseteq \mathcal{WA}_\ell(k) \cap \mathcal{G}_{A,B} \subseteq \mathcal{CRA}_\ell(k) \cap \mathcal{G}_{A,B}$$

Thus, by proving Proposition 4, we will obtain the expected result.

Proposition 4. $\mathcal{CRA}_\ell(k) \cap \mathcal{G}_{A,B} \subseteq \mathcal{CRA}^B_\ell(k)$

Sketch. Consider a cost register automaton R that computes a function in $\mathcal{G}_{A,B}$. On can prove that there is a bound N' such that along the runs of R, the values stored in the registers always belong to $B^*(B \cup B^{-1})^{\leqslant N'}$. This intuitively relies on the fact that for every run that can be completed into an accepting run, there exists a "short" completion, and this completion should lead to a weight in B^*. At anytime during a computation, the values stored in registers are thus of the form $\alpha_1\alpha_2$ with $\alpha_1 \in B^*$ and $\alpha_2 \in (B \cup B^{-1})^{\leqslant N'}$. For a given register X, the idea is then to associate with X the shortest α_1 satisfying these conditions, and to store the value α_2 in the states of the automaton. This ensures that every continuation of the computation will be compatible with the value α_1 already computed. We use here the fact that the weights are elements of the free group in order to prove the existence of a "shortest" α_1. This construction preserves parameters k and ℓ. □

7. Decidability of TP_k and application to register minimisation

In this section, we prove the decidability of the twinning property for some algebraic structures, and as a consequence, the decidability of the register minimisation problem on these structures. We consider the two following problems:

The TP_k Problem: given a weighted automaton W on some monoid $\mathbb{M}$ and a number k, does W satisfy the TP_k?

The Register Minimisation Problem: given $R \in \mathsf{CRA}^+_{\otimes c}(\mathbb{M})$ and a number k, does there exist $R' \in \mathsf{CRA}^+_{\otimes c}(\mathbb{M})$ 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 4. *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 5. *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 $delay(\alpha_i, \alpha_j) \neq \mathbf{1}$. 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 $delay(\alpha_i, \alpha_j)$.

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 6. *Over the group $(\mathbb{Z}, +)$, the TP_k problem is in* PSPACE *(k 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 (Alur and Raghothaman 2013)):

Corollary 1. *The register minimisation problem for $\mathsf{CRA}^+_{+c}(\mathbb{Z})$ is* PSPACE*-complete.*

This result slightly generalises that of (Alur and Raghothaman 2013), as we allow more general output functions. In addition, it follows from a general framework, and similar results for other infinitary groups (with computable operations) can be derived similarly.

Case of transducers. Let us first recall the procedure of (Weber and Klemm 1995) 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 v_1, v_2 on cycles 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 5. *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 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 guesses 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 7. *Over $(B^*, \cdot)$, the TP$_k$ problem is in* PSPACE *(k is given in unary).*

Corollary 2. *The register minimisation problem for $\mathsf{CRA}^+_c(B^*)$ is* PSPACE*-complete.*

8. 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 (Filiot et al. 2015)).

As a particular case, for the setting of additive cost regular functions, we obtain a generalization of the result of (Alur and Raghothaman 2013) on the minimisation of registers.

References

R. Alur and P. 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.

R. Alur and M. 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.

R. Alur, L. D'Antoni, J. V. Deshmukh, M. Raghothaman, and Y. 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.

K. Chatterjee, L. Doyen, and T. A. Henzinger. Quantitative languages. *ACM Trans. Comput. Log.*, 11(4), 2010. doi: 10.1145/1805950.1805953. URL `http://doi.acm.org/10.1145/1805950.1805953`.

C. Choffrut. Une caracterisation des fonctions sequentielles et des fonctions sous-sequentielles en tant que relations rationnelles. *Theor. Comput. Sci.*, 5(3):325–337, 1977. doi: 10.1016/0304-3975(77)90049-4. URL `http://dx.doi.org/10.1016/0304-3975(77)90049-4`.

E. Filiot, R. Gentilini, and J. 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.

E. Filiot, R. Gentilini, and J.-F. Raskin. Quantitative languages defined by functional automata. *Logical Methods in Computer Science*, 11(3:14): 1–32, 2015. URL `http://arxiv.org/abs/0902.3958`.

I. Jecker and E. Filiot. Multi-sequential word relations. In *Developments in Language Theory - 19th International Conference, DLT 2015, Liverpool, UK, July 27-30, 2015, Proceedings.*, volume 9168 of *Lecture Notes in Computer Science*, pages 288–299. Springer, 2015.

S. Lombardy and J. Sakarovitch. Sequential? *Theor. Comput. Sci.*, 356(1-2):224–244, 2006. doi: 10.1016/j.tcs.2006.01.028. URL `http://dx.doi.org/10.1016/j.tcs.2006.01.028`.

A. Maletti. Minimizing deterministic weighted tree automata. In C. Martín-Vide, F. Otto, and H. Fernau, editors, *Language and Automata Theory and Applications, Second International Conference, LATA 2008, Tarragona, Spain, March 13-19, 2008. Revised Papers*, volume 5196 of *Lecture Notes in Computer Science*, pages 357–372. Springer, 2008. ISBN 978-3-540-88281-7.

M. Mohri. Minimization algorithms for sequential transducers. *Theor. Comput. Sci.*, 234(1-2):177–201, 2000.

J. Sakarovitch and R. de Souza. Lexicographic decomposition of k -valued transducers. *Theory of Computing Systems*, 47(3):758–785, 2010.

M.-P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4, 1961.

A. Weber and R. Klemm. Economy of description for single-valued transducers. *Inf. Comput.*, 118(2): 327–340, 1995. doi: 10.1006/inco.1995.1071. URL `http://dx.doi.org/10.1006/inco.1995.1071`.

A. Appendix

In all the results and proofs in the Appendix, unless it is explicitly mentioned, $\mathbb{G}$ denotes an infinitary finitely generated group and Γ a fixed finite set of generators of $\mathbb{G}$.

Let $W = (Q, Q_{init}, Q_{final}, t, T)$ denote a weighted automaton over $\mathbb{G}$ computing a function f and M_W denote the maximum of the Cayley distances in $(\mathbb{G}, \Gamma)$ between $\mathbf{1}$ and any element occurring on the transitions of W.

Finally, let k denote a positive integer.

A.1 Delays

Lemma 1. *Given a group $\mathbb{G}$, for all $\alpha, \alpha', \beta, \beta', \gamma, \gamma' \in \mathbb{G}$,*

1. *$delay(\alpha, \beta) = \mathbf{1}$ if and only if $\alpha = \beta$,*
2. *if $delay(\alpha, \alpha') = delay(\beta, \beta')$ then $delay(\alpha\gamma, \alpha'\gamma') = delay(\beta\gamma, \beta'\gamma')$.*

Given a finitely generated group $\mathbb{G}$ and a finite set of generators Γ, for all $\alpha, \beta \in \mathbb{G}$, $d(\alpha, \beta) = d(\mathbf{1}, delay(\alpha, \beta))$.

Proof. 1. $delay(\alpha, \beta) = \alpha^{-1}\beta = 1$ if and only if $\alpha = \beta$.

2.

$$\begin{aligned} delay(\alpha\gamma, \alpha'\gamma') &= \gamma^{-1}\alpha^{-1}\alpha'\gamma' = \gamma^{-1}delay(\alpha, \alpha')\gamma' \\ &= \gamma^{-1}delay(\beta, \beta')\gamma' = delay(\beta\gamma, \beta'\gamma') \end{aligned}$$

Consider now the property stated for finitely generated groups. The integer $d(\mathbf{1}, delay(\alpha, \beta))$ is the smallest integer n such that $delay(\alpha, \beta) \in (\Gamma \cup \Gamma^{-1})^n$, or equivalently $\alpha^{-1}\beta \in (\Gamma \cup \Gamma^{-1})^n$. Thus, $d(\alpha, \beta) = d(\mathbf{1}, delay(\alpha, \beta))$. □

A.2 Infinitary group

The following Lemma relies on the fact that $\mathbb{G}$ is an infinitary group.

Lemma 6. *For all $\alpha, \alpha', \beta, \beta' \in \mathbb{G}$ such that $delay(\alpha, \alpha') \neq delay(\alpha\beta, \alpha'\beta')$, for all non negative integers m, there is N such that for all $n \geqslant N$, $d(\alpha\beta^n, \alpha'\beta'^n) > m$.*

Proof. Since $delay(\alpha, \alpha') \neq delay(\alpha\beta, \alpha'\beta')$, or equivalently $\alpha^{-1}\alpha' \neq \beta^{-1}\alpha^{-1}\alpha'\beta'$ then by infinitary hypothesis the set $\{delay(\alpha\beta^n, \alpha'\beta'^n) \mid n \in \mathbb{N}\}$ is infinite. Moreover, for all non negative integers m, $(\Gamma \cup \Gamma^{-1})^m$ is finite, thus there is N such that for all $n \geqslant N$,

$$delay(\alpha\beta^n, \alpha'\beta'^n) \notin (\Gamma \cup \Gamma^{-1})^{\leqslant m}$$

or equivalently, $d(\alpha\beta^n, \alpha'\beta'^n) > m$. □

A.3 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 $i \in \{1, \dots, m\}$:

$$\begin{aligned} &delay(\alpha_{1,j} \cdots \alpha_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}) \\ =&delay(\alpha_{1,j} \cdots \alpha_{i,j}\beta_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}\beta_{i,j'}) \end{aligned}$$

Lemma 4. *For all positive integer k, a weighted automaton satisfies TP_k if and only if it satisfies TP'_k.*

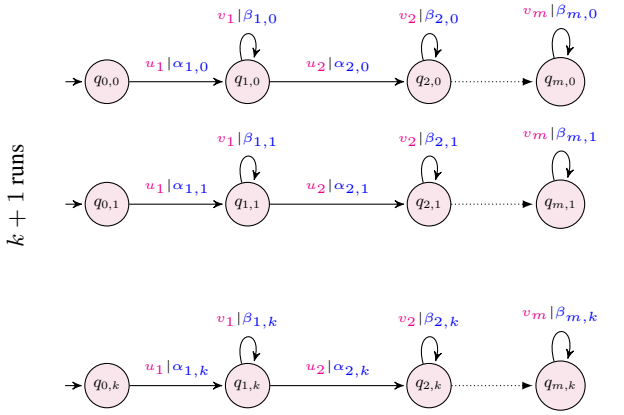

Figure 4. Equivalent definition of the twinning property of order k

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

- 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:

$$\begin{aligned} &delay(\alpha_{1,j} \cdots \alpha_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}) \\ =&delay(\alpha_{1,j} \cdots \alpha_{i,j}\beta_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}\beta_{i,j'}) \end{aligned}$$

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 $delay(\alpha_{1,j} \cdots \alpha_{i+1,j}, \alpha_{1,j'} \cdots \alpha_{i+1,j'}) = delay(\alpha_{1,j} \cdots \alpha_{i+1,j}\beta_{i+1,j}, \alpha_{1,j'} \cdots \alpha_{i+1,j'}\beta_{i+1,j'})$.

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 $i_1 < \dots < i_k$. Then for all $j \neq j'$, consider i_s the smallest index such that j and j' are not is the same set in P_{i_s}. Then:

$$\begin{aligned} &delay(\alpha_{1,j} \cdots \alpha_{i_s,j}, \alpha_{1,j'} \cdots \alpha_{i_s,j'}) \\ \neq&delay(\alpha_{1,j} \cdots \alpha_{i_s,j}\beta_{i_s,j}, \alpha_{1,j'} \cdots \alpha_{i_s,j'}\beta_{i_s,j'}) \end{aligned}$$

that proves that TP_k is not satisfied and concludes the proof. □

A.4 Proofs of Section 3.3

Proof of Lemma 2. The two following lemmas give the two reverse implications of Lemma 2.

Lemma 7. *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:*

$$q_j \xrightarrow{w|\alpha_j} p_j \quad \textit{for all } j \in \{0, \dots, k\},$$

there are $j \neq j'$ such that $d(\alpha_j, \alpha_{j'}) \leqslant 2M_W|Q|^{k+1}$.

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 $j \neq j'$, $d(\alpha_j, \alpha_{j'}) \leqslant 2M_W|Q|^{k+1}$, 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 $\mathbb{G}$, $\alpha_{i,j}$, $\beta_{i,j}$ 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 $i \in \{1, \dots, m\}$,

$$\begin{aligned} &delay(\alpha_{1,j} \cdots \alpha_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}) \\ &= delay(\alpha_{1,j} \cdots \alpha_{i,j}\beta_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}\beta_{i,j'}) \end{aligned}$$

Thus, by using at each step the item 2 of Lemma 1,

$$\begin{aligned} &delay(\alpha_j, \alpha_{j'}) \\ &= delay(\alpha_{1,j}\beta_{1,j}\alpha_{2,j} \cdots \alpha_{m,j}\beta_{m,j}, \\ &\qquad \alpha_{1,j'}\beta_{1,j'}\alpha_{2,j'} \cdots \alpha_{m,j'}\beta_{m,j'}) \\ &= delay(\alpha_{1,j}\alpha_{2,j}\beta_{2,j} \cdots \alpha_{m,j}\beta_{m,j}, \\ &\qquad \alpha_{1,j'}\alpha_{2,j'}\beta_{2,j'} \cdots \alpha_{m,j'}\beta_{m,j'}) \\ &\quad (\text{since } delay(\alpha_{1,j}, \alpha_{1,j'}) = delay(\alpha_{1,j}\beta_{1,j}, \alpha_{1,j'}\beta_{1,j'})) \\ &\qquad \vdots \\ &= delay(\alpha_{1,j}\alpha_{2,j} \cdots \alpha_{m,j}, \alpha_{1,j'}\alpha_{2,j'} \cdots \alpha_{m,j'}) \end{aligned}$$

Thus,

$$d(\alpha_j, \alpha_{j'}) = d(\alpha_{1,j} \cdots \alpha_{m,j}, \alpha_{1,j'} \cdots \alpha_{m,j'}) \leqslant 2M_W|Q|^{k+1}$$

since $|u_1 \cdots u_m| \leqslant |Q|^{k+1}$. □

Lemma 8. *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 $j \neq j'$, $d(\alpha_j, \alpha_{j'}) \geqslant m$.

Proof. Let m be a positive integer. The idea is to consider a witness as described in the Figure 2. If TP_k is not satisfied, then there are two runs such that by repeating the loops, the two runs can be as far as possible (with respect to the Cayley distance).

More precisely, 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:

$$\begin{aligned} &delay(\alpha_{1,j} \cdots \alpha_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}) \\ &\neq delay(\alpha_{1,j} \cdots \alpha_{i,j}\beta_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}\beta_{i,j'}) \end{aligned}$$

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:

$$\begin{aligned} &delay(\alpha_{1,j} \cdots \alpha_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}) \\ &\neq delay(\alpha_{1,j} \cdots \alpha_{i,j}\beta_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}\beta_{i,j'}) \end{aligned}$$

Thanks to Lemma 6, we can choose an integer N (the same for all such pairs (j, j')) such that

$$d(\alpha_{1,j} \cdots \alpha_{i,j}(\beta_{i,j})^N, \alpha_{1,j'} \cdots \alpha_{i,j'}(\beta_{i,j'})^N) > 2M_W L + m$$

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:

$$\begin{aligned} &delay(\alpha_{1,j} \cdots \alpha_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}) \\ &\neq delay(\alpha_{1,j} \cdots \alpha_{i,j}\beta_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}\beta_{i,j'}) \end{aligned}$$

Let:

$$\overline{\alpha}_j = \alpha_{1,j}\alpha_{2,j} \cdots \alpha_{i-1,j}\alpha_{i,j}(\beta_{i,j})^{t_i}$$

and:

$$\overline{\alpha'}_j = \alpha_{1,j'}\alpha_{2,j'} \cdots \alpha_{i-1,j'}\alpha_{i,j'}(\beta_{i,j'})^{t_i}$$

By using the second item of Lemma 1 at each steps before index i, one gets:

$$\begin{aligned} &delay(\alpha_{1,j}(\beta_{1,j})^{t_1}\alpha_{2,j} \cdots \alpha_{k,j}(\beta_{k,j})^{t_k}, \\ &\qquad \alpha_{1,j'}(\beta_{1,j'})^{t_1}\alpha_{2,j'} \cdots \alpha_{k,j'}(\beta_{k,j'})^{t_k}) \\ &= delay(\overline{\alpha}_j\alpha_{i+1,j} \cdots \alpha_{k,j}(\beta_{k,j})^{t_k}, \\ &\qquad \overline{\alpha'}_j\alpha_{i+1,j'} \cdots \alpha_{k,j'}(\beta_{k,j'})^{t_k}) \\ &= (\alpha_{i+1,j}(\beta_{i+1,j})^{t_{i+1}} \cdots \alpha_{k,j}(\beta_{k,j})^{t_k})^{-1} \\ &\qquad delay(\overline{\alpha}_j, \overline{\alpha'}_j)\alpha_{i+1,j'}(\beta_{i+1,j'})^{t_{i+1}} \cdots \alpha_{k,j'}(\beta_{k,j'})^{t_k} \\ &\notin (\Gamma \cup \Gamma^{-1})^{\leqslant m} \end{aligned}$$

□

Proof of Lemma 3. Lemma 3 is a direct consequence of the following lemma.

Lemma 9. *If W satisfies TP_k then for all words w, for all runs $\rho_1, \dots, \rho_{k+1}$ from an initial state to a co-accessible state, labelled by w, there are $j \neq j'$ such that $\rho_j, \rho_{j'}$ are $2M_W|Q|^{k+1}$-close.*

Proof. The ideas are similar to the ones given in Lemma 2. Consider $k+1$ runs $\rho_1, \dots, \rho_{k+1}$ 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 $2M_W|Q|^{k+1}$-close.

Then, for $1 \leqslant j \leqslant k+1$, 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 (starts and ends 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

$$\begin{aligned} &delay(\alpha_{1,j} \cdots \alpha_{i,j}, \alpha_{1,j'} \cdots \alpha_{i,j'}) \\ &\neq delay(\alpha_{1,j} \cdots \alpha_{i,j}\overline{\alpha}_{i,j}, \alpha_{1,j'} \cdots \overline{\alpha}_{i,j'}) \end{aligned}$$

where $\alpha_{i,j}$ (resp. $\overline{\alpha}_{i,j}$) is the weight of the run $\rho_{i,j}$ (resp. $\overline{\rho}_{i,j}$).

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

Proof of Proposition 1. The two following lemmas correspond to the two converse implications of Proposition 1.

Lemma 10. *A ℓ-valued weighted automaton with n states satisfies $TP_{n\ell}$.*

Proof. Suppose that a weighted automaton W does not satisfy $TP_{n\ell}$, and set $N = 2M_W n+2r+1$ where r is the maximum of the distances between $\mathbf{1}$ and the $t(q)$'s for all $q \in Q$. From Lemma 2, 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 $j \neq j'$, $d(\alpha_j, \alpha_{j'}) > N$. 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 accepting runs (labelled by the same word) is necessarily different from $\mathbf{1}$: for all $j \neq j'$, $d(\alpha_j, \alpha_{j'}) > N$ so by completing with a word of length less than n in an accepting run, the Cayley distance is necessarily at least 1 and so the delay is different from $\mathbf{1}$. Finally, there are $\ell + 1$ different values for the same word, that contradicts the fact that W is ℓ-valued. □

Lemma 11. *A weighted automaton satisfying TP_k for some natural k is finitely valued.*

Proof. Suppose that W satisfies TP_k. Consider the accepting runs labelled by a given word w. These runs have weights $\alpha_0, \ldots, \alpha_m$. By Lemma 3, there exists a subset $P \subseteq \{0, \ldots, m\}$ with k elements such that for all $j' \in \{0, \ldots, m\}$, there is $j \in P$ such that $d(\alpha_j, \alpha_{j'}) \leqslant 2M_W|Q|^{k+1}$. Since $\alpha_{j'} = \alpha_j delay(\alpha_j, \alpha_{j'})$ then a value of an accepting run labelled by w is of the form $\alpha\beta\gamma$ where α can take k values, β belongs to $(\Gamma \cup \Gamma^{-1})^{\leqslant 2M_W|Q|^{k+1}}$ and $\gamma = t(q)$ for some state q. So the automaton is $k|(\Gamma \cup \Gamma^{-1})^{\leqslant M_W}||Q|$-valued. □

Proof of Proposition 2.

Proof. Let W be a weighted automaton over an infinitary finitely generated group $\mathbb{G}$. Let Q denote its set of states, M_W the maximum of the distances between $\mathbf{1}$ and the weights of the transitions of W and r the maximum of the distances between $\mathbf{1}$ and the $t(q)$'s for all $q \in Q$. First, suppose that W does not satisfy TP_k. Then, by applying Lemma 2, for all positive integers m, there is a word w, initial states $q_0, \ldots, q_k$, co-accessible states $p_0, \ldots, p_k$ and $k + 1$ runs:

$$q_j \xrightarrow{w|\beta_j} p_j \text{ for all } j \in \{0, \ldots, k\},$$

such that for all $j \neq j'$, $d(\beta_j, \beta_{j'}) \geqslant m$. We can complete these $k + 1$ runs into accepting runs with words $u_0, \ldots, u_k$ such that for all i, $|u_i| \leqslant |Q|$. Let us write $w_i = wu_i$. The word w_i has a value α_i such that $d(\alpha_i, \beta_i) \leqslant |Q|M_W + r$. Since for all $j \neq j'$, $d(\beta_j, \beta_{j'}) \geqslant m$, then $d(\alpha_j, \alpha_{j'}) \geqslant m - 2M_W|Q| - 2r$. Set $n = 2|Q|$. Let N be a natural. Choose $m > N + 2M_W|Q| + 2r$. Then we obtain that $[\![W]\!]$ does not satisfy the k-bounded variation property.

Conversely, suppose that there is an integer $n > 0$ such that for all positive integers N, there are $w_0, \ldots, w_k \in A^*$, $\alpha_0, \ldots, \alpha_k \in \mathbb{G}$ 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 $i \neq j$, $d(\alpha_i, \alpha_j) > N$.

By item 2., there are $k + 1$ accepting runs in W labelled respectively by $w_0, \ldots, w_k$ with values $\alpha_0, \ldots, \alpha_k$. They induce $k + 1$ runs labelled by w, where w denotes the longest common prefix of the w_i's, starting in an initial state and ending in a co-accessible state. Let $\beta_0, \ldots \beta_k$ denote their values.

By contradiction, suppose now that there is $i \neq j$ such that $d(\beta_i, \beta_j) \leqslant 2M_W|Q|^{k+1}$. Then, by item 1. there is $i \neq j$ such that $d(\alpha_i, \alpha_j) \leqslant 2M_W|Q|^{k+1} + 2M_W n + 2r$. This is a contradiction with item 3 for $N = 2M_W|Q|^{k+1} + 2M_W n + 2r$.

Thus, for all $i \neq j$, $d(\beta_i, \beta_j) > 2M_W|Q|^{k+1}$. By Lemma 2, this implies that W does not satisfy TP_k. □

A.5 Proof of Section 4

Proof of Proposition 3. Remind that $\mathbb{G}$ is an infinitary finitely generated group and Γ a finite set of generators. Let k and ℓ be two positive integers.

A.5.1 From cost register automata to weighted automata

Given $R = (Q, q_{init}, \mathcal{X}, \delta, \mu)$ 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 is 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:

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

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

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 $\alpha \in \mathbb{G}$ an output associated to w by R. Let $(q_{init}, \nu_{init}), \ldots, (q, \nu)$ 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 $\alpha \in \mathbb{G}$ 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 2, 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 $j \neq j'$, $d(\alpha_j, \alpha_{j'}) \geqslant m$.

By construction, there is $p \in Q$, such that for all j, there is $X_j \in \mathcal{X}$ 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 $\alpha \in \mathbb{G}$ such that $\tau_p(X_j, \alpha) = i_{j'}$.
- $p_j = (p, i_j)$, $p_{j'} = (p, i_{j'})$ and there are $X \in \mathcal{X}$ and $\alpha, \beta \in \mathbb{G}$ such that $\tau_p(X, \alpha) = i_j$ and $\tau_p(X, \beta) = i_{j'}$.

Let us denote M_W (*resp.* M'_W) the maximum of the Cayley distances between $\mathbf{1}$ and the weights on the transitions of W (*resp.* the elements α of $\mathbb{G}$ such that there are $q \in Q$ and $X \in \mathcal{X}$ with $(X, \alpha) \in \mu(q)$).

By construction, $d(\alpha_j, \alpha_{j'}) \leqslant 2\max(M_W, M'_W)$. Thus by considering an m greater than $2\max(M_W, M'_W)$, we have a contradiction.

A.5.2 From weighted automata to cost register automata

Consider an ℓ-ambiguous weighted automaton

$$W = (Q, Q_{init}, Q_{final}, t, T)$$

satisfying TP$_k$. We will construct a cost register automaton $R = (Q', q_{init}, \mathcal{X}, \delta, \mu)$ with k registers and output size ℓ computing the same function.

Let M_W be the maximum of the Cayley distances between $\mathbf{1}$ and the weights on the transitions of W and let $N = 2M_W|Q|^{k+1}$.

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 $(\Gamma \cup \Gamma^{-1})^{\leqslant N} \cup \{\infty, \bot\}$ 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 $\bot^{-1} = \bot$),
- $f(x, x) \in \{\mathbf{1}, \bot\}$,
- if $f(x, x) = \bot$, then $f(x, y) = f(y, x) = \bot$,
- if $f(x, x) = f(y, y) = \mathbf{1}$ 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 $[f]$ 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 $[f_{init}]$. The set of registers is $\mathcal{X} = \{X_1, \dots, X_k\}$.

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

For all $p \in Q$, $f \in \Omega$, set: $E_p^{(f,a)} =$

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

Remark that $|E_p^{(f,a)}| = |E_p^{(g,a)}|$ if $f \equiv g$. If $|E_p^{(f,a)}| > \ell$ then the transition from $[f]$ 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 $\tau_p^{(g,a)}(q, i) = \tau_p^{(f,a)}(q, \sigma_q^{(f,g)}(i))$. We 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:

$$f_a((p', i'), (q', j')) =$$

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

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

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 $[f]$): $\delta([f], a) = ([f_a], h)$ 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:

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

Output function. For $x \in Q_{final} \times \{1, \dots, \ell\}$ such that $f(x, x) = \mathbf{1}$, 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 choose such that $\kappa_{(q,i)}^{(g)} = \kappa_{(q, \sigma_q^{(f,g)}(i))}^{(f)}$.

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

$$\mu([f]) = \{(X_{\kappa_x}, \alpha_x \beta_x) \mid x \in D_f\}$$

By construction, the automaton R uses k registers.

The correction of this construction can be deduced from Lemmas 12, 13, 14 and 15 that follow.

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 12. *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 $[f]$ such that: $f((q_i,\kappa),(q_{i'},\kappa')) =$*

$$\begin{cases} delay(\alpha_{i,j}, \alpha_{i',j'}) & \text{if } j \leqslant \ell_i, j' \leqslant \ell_{i'}, \rho_{i,j}, \rho_{i',j'} N\text{-close} \\ \infty & \text{if } j \leqslant \ell_i, j' \leqslant \ell_{i'}, \rho_{i,j}, \rho_{i',j'} \text{ not } N\text{-close} \\ \bot & \text{otherwise} \end{cases}$$

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 $[f_{init}]$ 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 $[f]$ such that: $f((q_i,j),(q_{i'},j')) =$

$$\begin{cases} delay(\beta_{i,j}, \beta_{i',j'}) & \text{if } j \leqslant \ell'_i, j' \leqslant \ell'_{i'}, \tau_{i,j}, \tau_{i',j'} N\text{-close} \\ \infty & \text{if } j \leqslant \ell'_i, j' \leqslant \ell'_{i'}, \tau_{i,j}, \tau_{i',j'} \text{ not } N\text{-close} \\ \bot & \text{otherwise} \end{cases}$$

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 $[f_a]$. Moreover, $f_a((q_s,m),(q_{s'},m')) =$

$$\begin{cases} \alpha^{-1} f((q_i,j),(q_{i'},j'))\beta \\ \qquad \text{if } \tau_{q_s}^{(f,a)}(q_i,j) = m, \tau_{q_{s'}}^{(f,a)}(q_{i'},j') = m', \\ \qquad (q_i,a,\alpha,q_s) \in T, (q_{i'},a,\beta,q_{s'}) \in T, \\ \qquad \alpha^{-1} f((q_i,j),(q_{i'},j'))\beta \in (\Gamma \cup \Gamma^{-1})^{\leqslant N} \\ \infty \\ \qquad \text{if } \tau_{q_s}^{(f,a)}(q_i,j) = m, \tau_{q_{s'}}^{(f,a)}(q_{i'},j') = m', \\ \qquad (q_i,a,\alpha,q_s) \in T, (q_{i'},a,\beta,q_{s'}) \in T \\ \qquad \alpha^{-1} f((q_i,s),(q_{i'},s'))\beta \notin (\Gamma \cup \Gamma^{-1})^{\leqslant N} \cup \{\bot\} \\ \bot \\ \qquad \text{otherwise} \end{cases}$$

If $j \leqslant \ell_i$, $j' \leqslant \ell_{i'}$, $\rho_{i,j}, \rho_{i',j'}$ N-close, then we are in the first case and $f_a((q_s,m),(q_{s'},m')) = \alpha^{-1} f((q_i,j),(q_{i'},j'))\beta = \alpha^{-1} delay(\beta_{i,\kappa}, \beta_{i',\kappa'})\beta = delay(\alpha_{s,m}, \alpha_{s',m'})$. If $j \leqslant \ell_i$, $j' \leqslant \ell_{i'}$, $\rho_{i,j}, \rho_{i',j'}$ not N-close then we are in the second case and $f_a((q_s,m),(q_{s'},m')) = \infty$. Otherwise, we are in the third case and $f_a((q_s,m),(q_{s'},m')) = \bot$. □

Lemma 13. *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 $[f]$ 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 $[f]$ where f maps all the pairs to $\bot$, and thus if there is an accepting run in R on w, it also ends in $[f]$ (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 12, there is a run on w' ending in $[g]$ with g satisfying the conditions of Lemma 12. 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 14. *If $[f]$ is an accessible state in R then r_f is well-defined.*

Proof. It is a direct consequence of Lemma 12 and Lemma 3. □

Lemma 15. *Suppose that there is an accepting run on w in R to $([f],\nu)$, 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 $1 \leqslant \kappa \leqslant k$, $\nu(X_\kappa) = \mathbf{1}$ 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 $([g],\nu)$ 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 12. □

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 12, there is an accepting run in R to some $([f],\nu)$ such that there is $x = (q,i)$ for some i corresponding to this run. By definition of r_f, there is κ such that $f(r_f(\kappa),(q,i)) \in (\Gamma \cup \Gamma^{-1})^{\leqslant N}$. Moreover, by Lemma 15, $\alpha t(q) = \nu(X_\kappa) f(r_f(\kappa),(q,i)) t(q)$. Finally, by the construction of the output function, $(X_\kappa, f(r_f(\kappa),(q,i))t(q))$ belongs to $\mu([f])$ 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 $[f]$ of R. Pairs (q,i) such that $f((q,i),(q,i)) = \mathbf{1}$ correspond to runs starting in an initial state and ending in q labelled by a same word. Moreover, thanks to Lemma 12, if $f((q,i),(q',i')) \neq \mathbf{1}$, 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, $|D_f| \leqslant \ell$ and R is of output size ℓ.

A.6 Proof of Section 5

In this section, $\mathbb{G}$ is not necessarily finitely generated. The definition of f-generator is given in Section 5.

Proposition 5. *Let W be an ℓ-valued weighted automaton over the semiring $\mathbb{P}_{\text{fin}}(\mathbb{G})$ where $(\mathbb{G},\otimes)$ is a infinitary, and k be a positive integer. The two following assertions are equivalent:*

- *The automaton W does not satisfy TP$_k$.*
- *There is an integer $N > 0$ and a f-generator Γ such that for all positive integers n, there are $w_0, \dots, w_k \in A^*$, $\alpha_0, \dots, \alpha_k \in \mathbb{G}$ 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 $i \neq j$, $delay(\alpha_i, \alpha_j) \notin \Gamma^{\leqslant n}$.*

Proof. Let Θ denote the set of elements of $\mathbb{G}$ occurring on the transitions of W and on the $t(q)$'s for all $q \in Q$, $\Delta = \Theta \cup \Theta^{-1}$. Let r be an integer such that $\{t(q) \mid q \in Q\} \in \Delta^{\leqslant r}$. First, suppose that W does not satisfy TP_k. Then, by applying Lemma 2, for all positive integers m, there is a word w, initial states $q_0, \ldots, q_k$, co-accessible states $p_0, \ldots, p_k$ and $k+1$ runs:

$$q_j \xrightarrow{w|\beta_j} p_j \text{ for all } j \in \{0, \ldots, k\},$$

such that for all $j \neq j'$, $delay(\beta_j, \beta_{j'}) \notin \Delta^{\leqslant m}$. We can complete these $k+1$ runs into accepting runs with words $u_0, \ldots, u_k$ such that for all i, $|u_i| \leqslant |Q|$. Let us write $w_i = wu_i$ and denote by $\alpha_0, \ldots, \alpha_k$ the weights of these runs. Since for all $j \neq j'$, $delay(\beta_j, \beta_{j'}) \notin \Delta^{\leqslant m}$ and $|u_i| \leqslant |Q|$, then $delay(\alpha_j, \alpha_{j'}) \notin \Delta^{\leqslant m-2|Q|-2r}$. Take $m \geqslant n + 2|Q| + 2r$ to obtain the expected result with $\Gamma = \Delta$ and $N = 2|Q|$.

Conversely, suppose that there is an integer $N > 0$ and a f-generator Γ such that for all positive integers n, there are $w_0, \ldots, w_k \in A^*$, $\alpha_0, \ldots, \alpha_k \in \mathbb{S}$ 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 $i \neq j$, $delay(\alpha_i, \alpha_j) \notin \Gamma^{\leqslant n}$.

Since $\Delta^{\leqslant 2|Q|^{k+1}+2N|Q|+2r}$ is finite then there is an integer n such that $(\Delta^{\leqslant 2|Q|^{k+1}+2N|Q|+2r} \cap \bigcup_{n \geqslant 0} \Gamma^n) \subseteq \Gamma^{\leqslant n}$. By item 2., there are $k+1$ accepting runs in W labelled respectively by $w_0, \ldots, w_k$ with values $\alpha_0, \ldots, \alpha_k$. 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 $\beta_0, \ldots \beta_k$ their values. By contradiction, suppose now that there is $i \neq j$ such that $delay(\beta_i, \beta_j) \in \Delta^{\leqslant 2|Q|^{k+1}}$. Then, by item 1. there is $i \neq j$ such that $delay(\alpha_i, \alpha_j) \in \Delta^{\leqslant 2|Q|^{k+1}+2|Q|N+2r}$. But $delay(\alpha_i, \alpha_j) \in \bigcup_{m \geqslant 0} \Gamma^m$. Thus, $delay(\alpha_i, \alpha_j) \in \Gamma^{\leqslant n}$. This is a contradiction with item 3. Thus, for all $i \neq j$, $delay(\beta_i, \beta_j) \notin \Delta^{\leqslant 2|Q|^{k+1}}$. By Lemma 2, this implies that W does not satisfy TP_k. □

A.7 Proofs of Section 6

Proposition 4.

$$\mathcal{CRA}_\ell(k) \cap \mathcal{G}_{A,B} \subseteq \mathcal{CRA}^B_\ell(k)$$

Proof. Let $(Q, q_{init}, \mathcal{X}, \delta, \mu)$ denote a register automaton over $\mathbb{G} = (B \cup B^{-1})^*$ computing a function $f \in \mathcal{G}_{A,B}$. The Cayley distance is defined in $(\mathbb{G}, B)$.

Let $N = |Q|m + s$ where:

- m is the maximum of the set:
$$\{d(\mathbf{1}, \alpha) \mid \delta(q, a) = (p, h), \\ h(Y) = (X, \alpha), q, p \in Q, a \in A, X, Y \in \mathcal{X}\}$$
- s is the maximum of the set:
$$\{d(\mathbf{1}, \alpha) \mid (X, \alpha) \in \mu(q), X \in \mathcal{X}, q \in Q\}$$

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

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 $\mathcal{X}$.

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 word in 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 $X, Y \in \mathcal{X}$ such that $g(Y) = (X, \alpha)$, if $r(X)\alpha \in B^*(B \cup B^{-1})^{\leqslant N}$. If $r(X)\alpha \in B^*(B \cup B^{-1})^{\leqslant N}$ (we will see that it is never the case for accessible, co-accessible states), then we set: $\delta'((q, r), a) = ((p, t), h)$ where $h(Y) = (X, r(X)\alpha)$, $t(Y) = \mathbf{1}$ for $X, Y \in \mathcal{X}$ such that $g(Y) = (X, \alpha)$.

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

First, it is easy to check that R and R' compute the same function. It relies on the following lemma:

Lemma 16. *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, we are in one of the following case:

- $\delta'((q', r'), a) = ((q, r), h)$ where $h(Y) = (X, (r'(X)\alpha)_1)$, $r(Y) = (r'(X)\alpha)_2$ for $X, Y \in R$ such that $g(Y) = (X, \alpha)$ if $r(X)\alpha \in B^*(B \cup B^{-1})^{\leqslant N}$. 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 $X, Y \in \mathcal{X}$ such that $g(Y) = (X, \alpha)$. Thus, $\sigma(Y)r(Y) = \nu'(X)\alpha = \nu(Y)$.
- $\delta'((q', r'), a) = ((p, t), h)$ where $h(Y) = (X, r'(X)\alpha)$, $r(Y) = \mathbf{1}$ if $r(X)\alpha \notin B^*(B \cup B^{-1})^{\leqslant N}$. Thus there is a run in R on w from $((q_{init}, r_{init}), \nu_{init})$ to $((q, r), \sigma)$ with $\sigma(Y)r(Y) = \sigma'(X)(r'(X)\alpha)$ for $X, Y \in \mathcal{X}$ 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, suppose that we are in the first case, $h(Y) = (X, (r'(X)\alpha)_1)$, $r(Y) = (r'(X)\alpha)_2$ for $X, Y \in \mathcal{X}$. Thus, there is a run in R on w from (q_{init}, ν_{init}) to (q, ν) with $\nu(Y) = \nu'(X)\alpha = \sigma'(X)r'(X)\alpha = \sigma'(X)(r'(X)\alpha)_1(r'(X)\alpha)_2 = \sigma(Y)r(Y)$. The second case is similar. □

Thank to the previous lemma, we can now 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 16, 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 16, 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, $(Y, \alpha) \in \mu(q)$. □

Thus, the cost register automaton R' has, by construction, the same number of registers, the same output size as R and computes the same function. What is left is to prove that it only uses elements in B^*.

Set E the minimal subset of $Q \times \mathcal{X}$ such that:

- $(q, X) \in E$ if there is $\alpha \in \mathbb{G}$ 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 $\alpha \in \mathbb{G}$.

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

Lemma 17. *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 a run from (q_{init}, ν_{init}) to (q, ν) and an alive register X in q such that $\nu(X) = c \in \mathbb{G}$. 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, $c \in B^*(B \cup B^{-1})^{\leqslant N}$. □

To prove that the updates of R' only use elements in B^*, we need to prove that for all accessible (q, r), for all $X, Y \in \mathcal{X}$, X alive in q, $\alpha \in \mathbb{G}$ such that $g(Y) = (X, \alpha)$, we have $r(X)\alpha$ belongs to $B^*(B \cup B^{-1})^{\leqslant N}$. We prove it by induction on the length of the shortest run ending in (q, r). It is true for (q_{init}, r_{init}) thanks to the definition of N.

By contradiction, if it is not true for (q, r), then $r(X)\alpha = ua^{-1}v$ with $u \in B^*$, $a \in B$, $v \neq av'$ for all $v' \in \mathbb{G}$, and $v \notin (B \cup B^{-1})^{<N}$. By induction hypothesis and completing the run, one of the output of the function is of the form $u'ua^{-1}vv'$ with $u' \in B^*$ and $v' \in (B \cup B^{-1})^{\leqslant N}$. This output is supposed to be in B^*. We are in one of the two following cases:

- The word u ends with a letter of B different from a. In this case, $u'ua^{-1}vv'$ cannot be in B^* since $v \neq av'$ for all $v' \in \mathbb{G}$, $v \notin (B \cup B^{-1})^{<N}$ and $v' \in (B \cup B^{-1})^{\leqslant N}$. That is a contradiction.
- The word u is empty. Since $v \neq av'$ for all $v' \in \mathbb{G}$, $v \notin (B \cup B^{-1})^{<N}$ and $v' \in (B \cup B^{-1})^{\leqslant N}$, then the last letter of u' must be an a. By definition of δ', if this a has been used to update the registers, it means that in R, the run corresponding to this output has a sequence of weights: $a, \alpha_1, \ldots, \alpha_s, a^{-1}$ such that $\alpha_1 \cdots \alpha_s = \mathbf{1}$, $\alpha_1 \cdots \alpha_s \neq a^{-1}\beta$ for all $\beta \in \mathbb{G}$ and by definition of δ', $\alpha_1 \cdots \alpha_s = \beta\gamma$ with $\beta \notin B^*(B \cup B^{-1})^{\leqslant N}$. Under these conditions, $u'\beta \notin B^*(B \cup B^{-1})^{\leqslant N}$, that is in contradiction with Lemma 17. $(\star)$

Finally, we need to prove that the output function of R' also uses only elements in B^*. Thus, let us prove that for all accessible (q, r) and $(X, \alpha) \in \mu(q)$, $r(X)\alpha$ belongs to B^*. By contradiction, if not, then $r(X)\alpha = ua^{-1}v$ with $a \in B$, $u \in B^*$ that does not end with a, $v \in \mathbb{G}, v \neq av'$ for some $v' \in \mathbb{G}$. Let us treat the two following cases:

- The word u ends with a letter $b \neq a$. In this case, by completing the run, there is an accepting run having an output $\beta r(X)\alpha$ with $\beta \in B^*$. But in this case, $\beta r(X)\alpha$ would not belong to B^*. Thus, R would not compute a function in Ω, that is a contradiction.
- The word u is empty. By completing the run into an accepting run in R', we get a sequence of weights of transitions $\alpha_1, \ldots, \alpha_s \in B^*$ such that one of the output is $\alpha_1 \cdots \alpha_s a^{-1}v$. Since the output is supposed to be in B^*, it means that the word $\alpha_1 \cdots \alpha_s$ ends with an a. And we can use a similar argument as $(\star)$.