Degree of sequentiality of weighted automata

Laure Daviaud[1], Ismael Jecker[2], Pierre-Alain Reynier[3], and Didier Villevalois[3]

[1] Warsaw University, Poland
[2] Université Libre de Bruxelles, Belgium
[3] Aix-Marseille Université, CNRS, LIF UMR 7279, France

Abstract. Weighted automata (WA) are an important formalism to describe quantitative properties. Obtaining equivalent deterministic machines is a longstanding research problem. In this paper we consider WA with a set semantics, meaning that the semantics is given by the set of weights of accepting runs. We focus on multi-sequential WA that are defined as finite unions of sequential WA. The problem we address is to minimize the size of this union. We call this minimum the degree of sequentiality of (the relation realized by) the WA.

For a given positive integer k, we provide multiple characterizations of relations realized by a union of k sequential WA over an infinitary finitely generated group: a Lipschitz-like machine independent property, a pattern on the automaton (a new twinning property) and a subclass of cost register automata. When possible, we effectively translate a WA into an equivalent union of k sequential WA. We also provide a decision procedure for our twinning property for commutative computable groups thus allowing to compute the degree of sequentiality. Last, we show that these results also hold for word transducers and that the associated decision problem is PSPACE-complete.

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 [6]. Weighted automata [15] (WA) is the oldest of such formalisms. They are defined over 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.

The decidability status of natural decision problems such as universality and equivalence highly depends on the considered semiring [14]. The first law of the semiring, used to aggregate the values computed by the different runs, plays an important role in the (un)decidability results. Inspired by the setting of word transducers, recent works have considered a set semantics that consists in keeping all these values as a set, instead of aggregating them [10], and proved several decidability results for the resulting class of finite-valued weighted automata [11].

For automata based models, a very important problem is to simplify the models. For instance, deterministic (a.k.a. sequential) machines are essential in order to derive efficient evaluation algorithms. In general, not every WA can be transformed into an equivalent sequential one. The sequentiality problem then asks, given a WA on some semiring $(S, \oplus, \otimes)$, whether there exists an equivalent sequential WA over $(S, \oplus, \otimes)$. This problem ranges from trivial to undecidable, depending on the considered semiring, see [13] for a survey.

Sequential transducers. Transducers define rational relations over words. They can be viewed as weighted automata over 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. When the underlying automaton is deterministic, then the transducer is said to be *sequential*. The class of sequential functions, *i.e.* those realized by sequential transducers, has been characterized among the class of rational functions by Choffrut, see for instance [4] for a presentation:

Theorem 1 ([7]). *Let T be a functional finite state transducer and $[\![T]\!]$ be the function realized by T. The following assertions are equivalent:*

i) $[\![T]\!]$ *satisfies the bounded variation property*
ii) T *satisfies the twinning property*
iii) $[\![T]\!]$ *is computed by a sequential transducer*

In this result, two key tools are introduced: a property of the function, known as the *bounded variation property*, and a pattern property of the transducer, known as the *twinning property*.

Multi-sequential weighted automata. Multi-sequential functions of finite words have been introduced in [8] as those functions that can be realized by a finite union of sequential transducers. A characterization of these functions among the class of rational functions is given in [8]. Recently, this definition has been lifted to relations in [12] where it is proved that the class of so-called multi-sequential relations can be decided in PTIME among the class of rational relations.

We consider in this paper *multi-sequential weighted automata*, defined as finite unions of sequential WA. As described above, and following [10], we consider weighted automata with a set semantics. We argue that multi-sequential WA are an interesting compromise between sequential and non-deterministic ones. Indeed, sequential WA have a very low expressiveness, while it is in general difficult to have efficient evaluation procedures for non-deterministic WA. Multi-sequential WA allow to encode standard examples requiring non-determinism (any regular look-ahead on the input word for instance), yet provide a natural evaluation procedure by evaluating simultaneously the different sequential WA of the union.

A natural problem then consists in minimizing the size of the union of multi-sequential WA that is, given a WA and a natural number k, decide whether it can be realized as a union of k sequential WA. More precisely, we are interested in identifying the minimal such k, that we call *degree of sequentiality* of the weighted automaton.

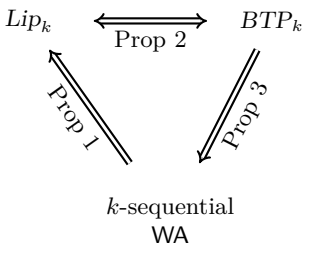
Contributions. In this paper, we propose a solution to the problem of the computation of the degree of sequentiality of WA. Following previous works [10, 9], we consider WA over infinitary finitely generated groups. We introduce new generalizations of the tools of Choffrut that allow us to characterize the relations that can be defined as unions of k sequential WA: first, a property of relations that extends a Lipschitz property for transducers, and is called *Lipschitz property of order k* (Lip_k for short); second, a pattern property of transducers, called *branching twinning property of order k* (BTP_k for short). We prove:

Theorem 2. *Let W be a weighted automaton over an infinitary finitely generated group and k be a positive integer. The following assertions are equivalent:*

i) $[\![W]\!]$ *satisfies the Lipschitz property of order k,*
ii) W *satisfies the branching twinning property of order k,*
iii) $[\![W]\!]$ *is computed by a k-sequential weighted automaton,*

In addition, the equivalent model of property iii) can be effectively computed.

As demonstrated by this result, a first important contribution of our work is thus to identify the correct adaptation of the properties of Choffrut suitable to characterize k-sequential relations. Sequential functions are characterized by both a bounded variation and a Lipschitz property [5]. In [9], we introduced a generalization of the bounded variation property to characterize relations that can be expressed using a particular class of cost register automata with exactly k registers, that encompasses the class of k-sequential relations. Though, to characterize k-sequential relations, we here introduce a generalization of the Lipschitz property. We actually believe that this class cannot be characterized by means of a generalization of the bounded variation property. Similarly, the difference between the twinning property of order k introduced in [9] and the branching twinning property of order k introduced in this paper is subtle: we allow here to consider runs on different input words, and the property requires the existence of two runs whose outputs are close on their common input words.



We now discuss the proof of Theorem 2 whose structure is depicted in the picture on the right. In [7], as well as in [9], the difficult part is the construction, given a machine satisfying the pattern property, of an equivalent deterministic machine. Here again, the most intricate proof of our work is that of Proposition 3: the construction, given a WA satisfying the BTP_k, of an equivalent k-sequential weighted automaton. It is worth noting that it is not a simple extension of [7] and [9]. Our proof proceeds by induction on k, and the result of [7] constitutes the base case while the tricky part resides in the induction step. Compared with [9], the construction of [9] stores pairwise delays between runs, and picks a minimal subset of "witness" runs that allows to express every other run. In [9], the choice of these witnesses may evolve along

an execution while in order to define a k-sequential WA, the way we choose the representative runs should be consistent during the execution. The technical part of our construction is thus the identification of a partition of size at most k of the different runs of the non-deterministic WA such that each element of this partition defines a sequential function. This relies on the branching structure of the twinning property we introduce in this paper.

Our result can also be rephrased in terms of cost register automata [2]. These are deterministic automata equipped with registers that aim to store along the run values from a given semiring S. The restriction of this model to updates of the form $X := X\alpha$ (we say that registers are *independent*) exactly coincides (if we allow k registers) with the class of k-sequential relations. Hence, our result also allows to solve the register minimization problem for this class of CRA.

Beyond weighted automata over infinitary groups, we also prove that our results apply to transducers from A^* to B^*.

Regarding decidability, we show that if the group $\mathbb{G}$ is commutative and has a computable internal operation, then checking whether the BTP_k is satisfied is decidable. As a particular instance of our decision procedure, we obtain that this can be decided in PSPACE for $\mathbb{G} = (\mathbb{Z}, +, 0)$, and show that the problem is PSPACE-hard. Last, we prove that checking the BTP_k for finite-state transducers is also PSPACE-complete.

Organization of the paper. We start with definitions in Section 2. In Section 3, we introduce our original Lipschitz and branching twinning properties. We present our main construction in Section 4. Section 5 is devoted to the presentation of our results about cost register automata, while transducers are dealt with in Section 6. Last we present our decidability results and their application to the computation of the degree of sequentiality in Section 7. Omitted proofs can be found in the Appendix.

2 Definitions and examples

Prerequisites and notation. 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 $\alpha\beta$. 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. The monoid (*resp.* group) is said to be *commutative* when $\oplus$ is commutative. Given a finite alphabet B, we denote by $\mathcal{F}(B)$ the free group generated by B.

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 with neutral element $\mathbf{0}$, $(S, \otimes, \mathbf{1})$ is a monoid with 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 $dist(u, v) = |u| + |v| - 2 * |lcp(u, v)|$ where $lcp(u, v)$ is the longest common prefix of u and v.

When considering a group $\mathbb{G}$ and $\alpha,\ \beta \in \mathbb{G}$, we define the *delay* between α and β as $\alpha^{-1}\beta$, denoted by $delay(\alpha, \beta)$.

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 $delay(\alpha\gamma, \alpha'\gamma') = delay(\beta\gamma, \beta'\gamma')$.*

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 $\alpha\beta$. 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 $d(\alpha, \beta)$.

For any $\alpha \in \mathbb{G}$, we define the *size* of α (with respect to the set of generators Γ) as the natural number $d(\mathbf{1}, \alpha)$. It is denoted by $|\alpha|$. Note that for a word u, considered as an element of $\mathcal{F}(A)$, the size of u is exactly the length of u (that is why we use the same notation).

Lemma 2. *Given a finitely generated group $\mathbb{G}$ and a finite set of generators Γ, for all $\alpha, \beta \in \mathbb{G}$, $d(\alpha, \beta) = |delay(\alpha, \beta)|$.*

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. See [10] for other examples.

Weighted automata. Given a semiring $\mathbb{S}$, weighted automata (WA) are non-deterministic finite automata in which transitions have for weights elements of $\mathbb{S}$. Weighted automata 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 labeled by w.

We will consider, for some monoid $\mathbb{M}$, weighted automata over the semiring $\mathbb{P}_{\mathsf{fin}}(\mathbb{M})$. In our settings, instead of considering the semantics of these automata in terms of functions from A^* to $\mathbb{P}_{\mathsf{fin}}(\mathbb{M})$, we will consider it in terms of relations over A^* and $\mathbb{M}$. More precisely, a weighted automaton (with initial and final relations), is formally defined as follows:

Definition 1. *Let A be a finite alphabet, a weighted automaton W over some monoid $\mathbb{M}$ is a tuple $(Q, t_{init}, t_{final}, T)$ where Q is a finite set of states, $t_{init} \subseteq Q \times \mathbb{M}$ (resp. $t_{final} \subseteq Q \times \mathbb{M}$) is the finite initial (resp. final) relation, $T \subseteq Q \times A \times \mathbb{M} \times Q$ is the finite set of transitions.*

A state q is said to be *initial* (*resp. final*) if there is $\alpha \in \mathbb{M}$ such that $(q, \alpha) \in t_{init}$ (*resp.* $(q, \alpha) \in t_{final}$), depicted as $\xrightarrow{\alpha} q$ (resp. $q \xrightarrow{\alpha}$). A run ρ from a state q_1 to a state q_k 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* of such a run is the element of $\mathbb{M}$, $\alpha = \alpha_1\alpha_2 \cdots \alpha_k$. We depict this situation as $q_1 \xrightarrow{w|\alpha} q_{k+1}$. The run ρ is said to be *accepting* if q_1 is initial and q_{k+1} final. This automaton W computes a relation $[\![W]\!] \subseteq A^* \times \mathbb{M}$ defined by the set of pairs $(w, \alpha\beta\gamma)$ such that there are $p, q \in Q$ with $\xrightarrow{\alpha} p \xrightarrow{w|\beta} q \xrightarrow{\gamma}$.

An automaton is *trimmed* if all its states appear in an accepting run. W.l.o.g., we assume that the automata we consider are trimmed.

Given a weighted automaton $W = (Q, t_{init}, t_{final}, T)$ over some finitely generated group $\mathbb{G}$ with finite set of generators Γ, we define the constant M_W with respect to Γ as $M_W = \max\{|\alpha| \mid (p, a, \alpha, q) \in T \text{ or } (q, \alpha) \in t_{init} \cup t_{final}\}$.

For any positive integer ℓ, a relation $R \subseteq X \times Y$ is said to be *ℓ-valued* if, for all $x \in X$, the set $\{y \mid (x, y) \in R\}$ contains at most ℓ elements. It is said to be *finitely valued* if it is ℓ-valued for some ℓ. A weighted automaton W is said to be *ℓ-valued* (resp. *finite-valued*) if it computes a ℓ-valued (resp. finite-valued) relation.

The union of two weighted automata $W_i = (Q_i, t^i_{init}, t^i_{final}, T_i)$, for $i \in \{1, 2\}$, with disjoint states $Q_1 \cap Q_2 = \varnothing$ is the automaton $W_1 \cup W_2 = (Q_1 \cup Q_2, t^1_{init} \cup t^2_{init}, t^1_{final} \cup t^2_{final}, T_1 \cup T_2)$. States can always be renamed to ensure disjointness. It is trivial to verify that $[\![W_1 \cup W_2]\!] = [\![W_1]\!] \cup [\![W_2]\!]$. This operation can be generalized to the union of k weighted automata.

Definition 2. *A weighted automaton $(Q, t_{init}, t_{final}, T)$ over $\mathbb{M}$ is said to be sequential if $|t_{init}| = 1$ and if for all $p \in Q$, $a \in A$ there is at most one transition in T of the form (p, a, α, q). It is said to be k-sequential if it is a union of k sequential automata. It is said to be multi-sequential if it is k-sequential for some k. A relation is said to be k-sequential (resp. multi-sequential) if it can be computed by a k-sequential (resp. multi-sequential) automaton. The* degree of sequentiality *of the relation is the minimal k such that it is k-sequential.*

Observe that unlike the standard definition of sequential weighted automata over $\mathbb{M}$ (see for instance [10]), we may associate finite sets of weights to final states, and not only singletons. This has an impact on the sequentiality degree, but we could handle this standard setting as well. Our definition allows us to simplify the presentation of our results.

Example 1. Let us consider $A = \{a, b\}$ and $(\mathbb{M}, \otimes, \mathbf{1}) = (\mathbb{Z}, +, 0)$. The weighted automaton W_0 given in Figure 1 computes the function f_{last} that associates with a word wa (resp. wb) its number of occurrences of the letter a (resp. b),

and associates 0 with the empty word. It is easy to verify that the degree of sequentiality of f_{last} is 2. It is also standard that the function f^*_{last} mapping the word $u_1\#\dots\#u_n$ (for any n) to $f_{last}(u_1)+\dots+f_{last}(u_n)$ is not multi-sequential (see for instance [12]).

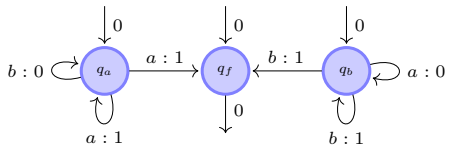


Fig. 1. Examples of a weighted automaton W_0 computing the function f_{last}.

3 Lipschitz and branching twinning properties

Sequential transducers have been characterized in [7] by Choffrut by means of a so-called bounded-variation property and a twinning property. The bounded-variation property is actually equivalent to a Lipschitz-like property (see for instance [5]). We provide adaptations of the Lipschitz and twinning properties so as to characterize k-sequential WA.

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

3.1 Lipschitz property of order k

Given a partial mapping $f : A^* \rightharpoonup B^*$, the Lipschitz property states that there exists $L \in \mathbb{N}$ such that for all $w, w' \in A^*$ such that $f(w), f(w')$ are defined, we have $dist(f(w), f(w')) \leqslant L dist(w, w')$ (see [5]). Intuitively, this property states that, for two words, their images by f differ proportionally to those words. This corresponds to the intuition that the function can be expressed by means of a sequential automaton.

When lifting this property to functions that can be expressed using a k-sequential automaton, we consider $k+1$ input words and require that two of those must have proportionally close images by f. The extension to relations $R \subseteq A^* \times B^*$ requires that for all $k+1$ pairs chosen in R, two of those have their range components proportionally close to their domain components. In addition, for relations, an input word may have more than one output word, we thus need to add a constant 1 in the right-hand side. Finally, our framework is that of infinitary finitely generated groups. Instead of $dist(,)$, we use the Cayley distance $d(,)$ to compare elements in the range of the relation.

Definition 3. *A relation $R \subseteq A^* \times \mathbb{G}$ satisfies the Lipschitz property of order k if there is a natural L such that for all pairs $(w_0, \alpha_0), \dots, (w_k, \alpha_k) \in R$, there are $0 \leqslant i < j \leqslant k$ satisfying $d(\alpha_i, \alpha_j) \leqslant L(dist(w_i, w_j) + 1)$.*

Example 2. The group $(\mathbb{Z}, +, 0)$ is finitely generated with $\{1\}$ as a set of generators. The function f_{last} does not satisfy the Lipschitz property of order 1 (take $w_1 = a^N a$ and $w_2 = a^N b$), but it satisfies the Lipschitz property of order 2.

Using the pigeon hole principle, it is easy to prove the implication from *iii*) to *i*) of Theorem 2:

Proposition 1. *A k-sequential relation satisfies the Lipschitz property of order k.*

3.2 Branching twinning property of order k

The idea behind the branching twinning property of order k is to consider $k+1$ runs labeled by arbitrary words with k cycles. If the branching twinning property is satisfied then there are two runs among these $k+1$ such that the values remain close (i.e. the Cayley distance between these values is bounded) along the prefix part of these two runs that read the same input. Note that the branching twinning property of order k is stronger than the twinning property of order k (TP_k for short) previously defined in [9]. Indeed, it requires the same conclusion as the TP_k but only for runs labeled by the same input words. In particular, satisfaction of the branching twinning property of order k implies that of the TP_k of [9].

Unlike the TP_k, the branching twinning property considers runs on different input words. This property is thus named after the intuition that the $k+1$ runs can be organized in a tree structure where the prefixes of any two runs are on the same branch up to the point where those two runs do not read the same input anymore.

Definition 4. *A weighted automaton over $\mathbb{G}$ satisfies the branching twinning property of order k (denoted by BTP_k) if: (see Figure 2)*

- *for all states $\{q_{i,j} \mid i, j \in \{0, \ldots, k\}\}$ with $q_{0,j}$ initial for all j,*
- *for all γ_j such that $(q_{0,j}, \gamma_j) \in t_{init}$ with $j \in \{0, \ldots, k\}$,*
- *for all words $u_{i,j}$ and $v_{i,j}$ with $1 \leqslant i \leqslant k$ and $0 \leqslant j \leqslant k$ such that there are $k+1$ runs satisfying for all $0 \leqslant j \leqslant k$, for all $1 \leqslant i \leqslant k$, $q_{i-1,j} \xrightarrow{u_{i,j}|\alpha_{i,j}} q_{i,j}$ and $q_{i,j} \xrightarrow{v_{i,j}|\beta_{i,j}} q_{i,j}$,*

there are $j \neq j'$ such that for all $i \in \{1, \ldots, k\}$, if for every $1 \leqslant i' \leqslant i$, we have $u_{i',j} = u_{i',j'}$ and $v_{i',j} = v_{i',j'}$, then we have

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

Example 3. The weighted automaton W_0 does not satisfy the BTP_1 (considering loops around q_a and q_b). One can prove however that it satisfies the BTP_2.

Let us denote by W_1 the weighted automaton obtained by concatenating W_0 with itself, with a fresh # separator letter. W_1 realizes the function f^2_{last} defined as $f^2_{last}(u\#v) = f_{last}(u) + f_{last}(v)$. We can see that the minimal k such that W_1 satisfies the BTP_k is $k = 4$. As we will see, this is the sequentiality degree of f^2_{last}.

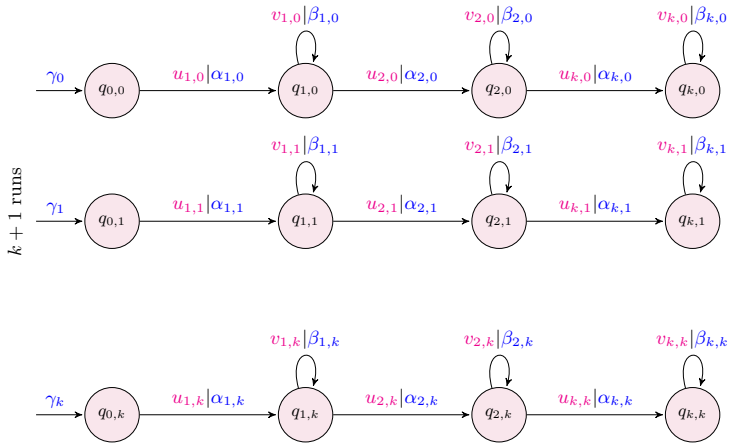


Fig. 2. Branching twinning property of order k

3.3 Equivalence of Lipschitz and branching twinning properties

We can prove that a weighted automaton satisfies the BTP_k if and only if its semantics satisfies the Lipschitz property of order k. This implies that the branching twinning property of order k is a machine independent property, *i.e.* given two WA W_1, W_2 such that $[\![W_1]\!] = [\![W_2]\!]$, W_1 satisfies the BTP_k iff W_2 satisfies the BTP_k.

Proposition 2. *A weighted automaton W over an infinitary finitely generated group $\mathbb{G}$ satisfies BTP_k if and only if $[\![W]\!]$ satisfies the Lipschitz property of order k.*

Proof (Sketch). Let us sketch the proof of the Proposition. First, suppose that W does not satisfy the BTP_k. Then consider a witness of this non satisfaction. Fix an integer L. By pumping the loops in this witness (enough time and going backward), one can construct $k+1$ words that remain pairwise sufficiently close while their outputs are pairwise at least at distance L. This leads to prove that $[\![W]\!]$ does not satisfy the Lipschitz property of order k.

Conversely, consider that the BTP_k is satisfied. For all $k+1$ pairs of words and weights in $[\![W]\!]$, we have $k+1$ corresponding runs in W labeled by those words. By exhibiting cycles on these runs, we can get an instance of BTP_k as in Figure 2 such that the non-cycling part is bounded (in length). By BTP_k, there are two runs that have the same delays before and after the loops appearing in their common prefix. Thus, we can bound the distance between the two weights produced by those runs proportionally to the distance between the two input words, proving that the Lipschitz property is satisfied. □

4 Constructing a k-sequential weighted automaton

As explained in the introduction, the most intricate part in the proof of Theorem 2 is to prove that *ii*) implies *iii*). We give a constructive proof of this fact as stated in the following proposition.

Proposition 3. *Given a weighted automaton W satisfying the BTP_k, one can effectively build k sequential weighted automata whose union is equivalent to W.*

Let $W = (Q, t_{init}, t_{final}, T)$ be a weighted automaton that satisfies the BTP_k. The construction is done in two steps. First, we build an infinite sequential weighted automaton D_W equivalent to W, using the subset construction with delays presented in [4]. Then, by replacing infinite parts of D_W with finite automata, we build k sequential weighted automata whose union is equivalent to W.

Let us sketch the main ideas behind the construction of D_W. The states of D_W are the subsets S of $Q \times \mathbb{G}$. On input $u \in A^*$, D_W selects an initial run $\rho : \xrightarrow{\alpha_0} p_0 \xrightarrow{u|\alpha} p$ of W, outputs the corresponding $\alpha \in \mathbb{G}$, and, in order to keep track of all the runs $\rho' : \xrightarrow{\beta_0} q_0 \xrightarrow{u|\beta} q$ of W over the input u, stores in its state the corresponding pairs $(q', delay(\alpha_0\alpha, \beta_0\beta))$. The detailed construction, together with the proofs of its properties, adapted from [4] to fit our settings, can be found in the appendix.

If W is a transducer, i.e., a weighted automaton with weights in a free monoid, and W satisfies the BTP_1, which is equivalent to the twinning property, Lemma 17 of [4] proves that the trim part of D_W is finite. This lemma can be generalized to any kind of weighted automata, proving our proposition in the particular case $k = 1$. Let us now prove the general result by induction. Suppose that $k > 1$, and that the proposition is true for every integer strictly smaller than k. We begin by exposing two properties satisfied by D_W.

Since W satisfies the BTP_k, it also satisfies the notion of TP_k introduced in [9], and, by Proposition 1 of that paper, it is ℓ-valued for some integer ℓ effectively computable. Let $N_W = 2M_W|Q|^{\ell|Q|}$, let $S \in Q \times \mathbb{G}$ be a state of the trim part of D_W, and let $W_S = (Q, S, t_{final}, T)$ be the weighted automaton obtained by replacing the initial output relation of W with S. The following properties are satisfied.

P_1: The size of S is bounded by $\ell|Q|$;

P_2: If there exists a pair $(q, \alpha) \in S$ such that $|\alpha| > N_W$, $[\![W_S]\!]$ is k-sequential.

The proof of P_1 follows from the ℓ-valuedness of W. The main difficulty of the demonstration of Proposition 3 lies in the proof of P_2, which can be sketched as follows. Using the fact that there exists $(q, \alpha) \in S$ such that $|\alpha| > N_W$, we expose a partition of S into two subsets S' and S'' satisfying the $BTP_{k'}$, respectively the $BTP_{k''}$, for some $1 \leqslant k', k'' < k$ such that $k' + k'' \leqslant k$. This is proved by using the fact that W satisfies the BTP_k, and that the branching nature of the BTP allows us to combine unsatisfied instances of the BTP over $W_{S'}$ and $W_{S''}$ to build

unsatisfied instances of the BTP over W. Then, since $k' < k$ and $k'' < k$, $[\![S']\!]$ is k'-sequential and $[\![S'']\!]$ is k''-sequential by the induction hypothesis. Finally, as S is the union of S' and S'', W_S is equivalent to the union of $W_{S'}$ and $W_{S''}$, and $\mathbf{P}_2$ follows, since $k' + k'' \leqslant k$.

The properties $\mathbf{P}_1$ and $\mathbf{P}_2$ allow us to expose k sequential weighted automata $\overline{V}_1, \ldots, \overline{V}_k$ whose union is equivalent to W. Let U denote the set containing the accessible states S of D_W that contain only pairs (q, α) satisfying $|\alpha| \leqslant N_W$. As there are only finitely many $\alpha \in \mathbb{G}$ such that $|\alpha| \leqslant N_W$, $\mathbf{P}_1$ implies that U is finite. Moreover, as a consequence of $\mathbf{P}_2$, for every state $S \notin U$ in the trim part of D_W, W_S can be expressed as the union of k sequential weighted automata $V_i(S)$, with $1 \leqslant i \leqslant k$. For every $1 \leqslant i \leqslant k$, let $\overline{V}_i$ be the sequential weighted automaton that copies the behaviour of D_W as long as the latter stays in U, and swaps to $V_i(S)$ as soon as D_W enters a state $S \notin U$. Then D_W is equivalent to the union of the $\overline{V}_i$, $1 \leqslant i \leqslant k$, which proves the desired result, since D_W is equivalent to W. Once again, the detailed proofs can be found in the appendix.

5 Cost register automata with independent registers

Recently, a new model of machine, named cost register automata (CRA), has been introduced in [2]. We present in this section how the class of k-sequential relations is also characterized by a specific subclass of cost register automata.

A cost register automaton (CRA) [2] 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 relation in $A^* \times S$.

In this paper, we focus on a particular structure $(\mathbb{M}, \otimes c)$ defined over a monoid $(\mathbb{M}, \otimes, \mathbf{1})$. 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 class of automata is called additive cost register automata [3]. When $\mathbb{M}$ is the free monoid $(A^*, ., \varepsilon)$, this class is a subclass of streaming string transducers [1] 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 [2] 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.

Definition 5. *A cost register automaton on the alphabet A over the monoid $(\mathbb{M}, \otimes, \mathbf{1})$ is a tuple $(Q, q_{\text{init}}, \mathcal{X}, \delta, \mu)$ where Q is a finite set of states, $q_{\text{init}} \in Q$ 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, μ is a finite set of $Q \times \mathcal{X} \times \mathbb{M}$ (the output relation).*

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 and there are X, α such that $(q_{k+1}, X, \alpha) \in \mu$.

A cost register automaton C computes a relation $[\![C]\!] \subseteq A^* \times \mathbb{M}$ defined by the set of pairs $(w, \nu_{k+1}(X)\alpha)$ such that $(q_1, \nu_1)(q_2, \nu_2) \dots (q_{k+1}, \nu_{k+1})$ is an accepting run of C on w and $(q_{k+1}, X, \alpha) \in \mu$.

Definition 6. *A cost register automaton is said to be with independent registers if for any update function f which labels a transition, if $f(Y) = (X, \alpha)$ then $X = Y$.*

Example 4. Consider $A = \{a, b\}$ and $(\mathbb{M}, \otimes, \mathbf{1}) = (\mathbb{Z}, +, 0)$. The cost register automaton C_0 given in Figure 3 computes the function f_{last} introduced in Example 1. The register X_a (*resp.* X_b) stores the number of occurrences of the letter a (*resp.* b). Observe that these two registers are independent.

Independence of registers is tightly related to sequentiality of WA. We prove:

Proposition 4. *For all positive integers k, a relation is k-sequential if and only if it is computed by a cost register automaton with k independent registers.*

CRA are deterministic by definition, and a challenging minimisation problem is captured by the notion of *register complexity*. It is defined for a relation as the minimal integer k such that it can be defined by a CRA with k registers. By Proposition 4, results on the computation of the degree of sequentiality presented in Section 7 thus also allow to compute the register complexity for CRA with independent registers.

One can also show that the class of CRA with k independent registers is equivalent to the class of CRA with k registers, updates of the form $X := Y\alpha$, and that are copyless (every register appears at most once in the right-hand side of an update function).

Example 5. We have seen that the minimal k such that W_1 satisfies the BTP_k is $k = 4$. Thus it can be computed by a cost register automata with 4 independent registers. One can observe that the TP_2 from [9] is however satisfied. Indeed, there exists a cost register automata with only 2 registers realizing f^2_{last}, but the two registers are not independent (see Appendix Section A.1 for details).

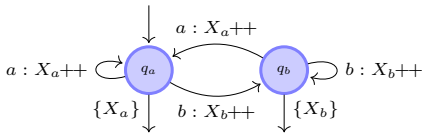


Fig. 3. Example of a cost register automaton C_0 computing the function f_{last}. The updates are abbreviated: $X_a{+}{+}$ means both $X_a := X_a + 1$ and $X_b := X_b$ (and conversely).

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 $\mathcal{F}(B)$. We say that a transducer T satisfies the branching twinning property of order k if, viewed as a weighted automaton over $\mathcal{F}(B)$, it satisfies the BTP_k. Similarly, a relation $R \subseteq A^* \times B^*$ is said to satisfy the Lipschitz property of order k iff it is the case when viewing R as a relation in $A^* \times \mathcal{F}(B)$.

A relation R of $A^* \times B^*$ is said to be *positive k-sequential* if it is computed by a k-sequential weighted automaton with weights in B^* (weights on the transitions in B^* and initial and final relations in $Q \times B^*$ where Q is its set of states). As for the general case, it is easy to see that a relation is positive k-sequential if and only if it is computed by a cost register automaton with k independent registers, with updates of the form $X := Xc$ where $c \in B^*$ and with an output relation $\mu \subseteq Q \times \mathcal{X} \times B^*$.

Theorem 3. *Let T be a transducer from A^* to B^*, and k be a positive integer. The following assertions are equivalent:*

i) $[\![T]\!]$ *satisfies the Lipschitz property of order k,*
ii) T *satisfies the branching twinning property of order k,*
iii) $[\![T]\!]$ *is positive k-sequential.*

The assertions *i)* and *ii)* are equivalent by Theorem 2. The fact that the assertion *iii)* implies the assertion *ii)* is also a consequence of Theorem 2 and of the fact that the branching twinning property of order k is a machine-independent characterization. Finally, it remains to prove that the assertion *ii)* implies the assertion *iii)*.

By hypothesis, $[\![T]\!] \subseteq A^* \times B^*$ is computed by a transducer that satisfies the branching twinning property of order k. Thus, by Theorem 2, it is computed by a cost register automaton over $\mathcal{F}(B)$ with k independent registers. We conclude using the:

Proposition 5. *A relation in $A^* \times B^*$ is computed by a cost register automaton over $\mathcal{F}(B)$ with k independent registers if and only if it is computed by a cost register automaton over B^* with k independent registers.*

7 Decidability of BTP_k and computation of the sequentiality degree

In this section, we prove the decidability of the following problem under some hypotheses on the group $\mathbb{G}$:
The BTP_k Problem: given a weighted automaton W over some group $\mathbb{G}$ and a number k, does W satisfy the BTP_k?

As a corollary of Theorem 2, this allows to compute the degree of sequentiality for weighted automata. We will consider two settings: first weighted automata over some computable commutative group and second, word transducers.

Our decision procedures non-deterministically guess a counter-example to the BTP_k. First, we show that if there exists such a counter-example with more than k loops, then there exists one with k loops. For simplicity, we can assume that the counter-example contains $k(k+1)/2$ loops *i.e.* exactly one loop per pair (j, j'), with $0 \leqslant j < j' \leqslant k$. This allows the procedure to first guess the "skeleton" of the counter example, and then check that this skeleton can be turned into a real counter-example. The skeleton consists of the vectors of states, and, for each pair (j, j') of run indices, indicates the index $\chi(j, j')$ of the last loop such that input words of runs j and j' are equal up to this loop, and the index $\eta(j, j')$ of the loop that induces a different delay (with $\eta(j, j') \leqslant \chi(j, j')$).

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

Theorem 4. *Let $\mathbb{G} = (G, \otimes)$ be a commutative group such that the operation $\otimes$ and the equality check are computable. Then the BTP_k problem is decidable.*

Proof (Sketch). It is easy to observe that for commutative groups, the constraint expressed on the delay in the BTP_k boils down to checking that loops have different weights.

The procedure first guesses the skeleton of a counter-example as explained above. The procedure then non-deterministically verifies that the skeleton can be completed into a concrete counter-example. To this end, it uses the information stored in this skeleton about how input words are shared between runs (indices $\chi(j, j')$) to identify the power $p \leqslant k+1$ of W in which the run should be identified. The procedure is based on the two following subroutines:

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

If we consider the group $(\mathbb{Z}, +)$, we can verify that the above procedure runs in PSPACE if k is given in unary. In addition, using ideas similar to a lower bound proved in [3], we can reduce the emptiness of k deterministic finite state automata to the BTP_k problem, yielding:

Theorem 5. *Over $(\mathbb{Z}, +)$, the BTP_k problem is* PSPACE*-complete (k given in unary).*

Case of transducers. For word transducers, the authors of [16] prove that a counter-example to the (classical) twinning property is either such that loops have output words of different length, or such that output words produced on the runs leading to the loops have a mismatch.

Inspired by this result, we show that the skeleton described above can be enriched with the information, for each pair of run indices (j, j'), whether one should look for a loop whose output words have distinct lengths, or for a mismatch on the paths leading to the loop. These different properties can all be checked in PSPACE, yielding:

Theorem 6. *Over $(B^*, \cdot)$, the BTP_k problem is* PSPACE*-complete (k is given in unary)*[4].

8 Conclusion

Multi-sequential machines are an interesting compromise between sequential and finite-valued ones. This yields the natural problem of the minimization of the size of the union. In this paper, we have solved this problem for weighted automata over an infinitary finitely generated group, a setting that encompasses standard groups. To this end, we have introduced a new twinning property, as well as a new Lipschitz property, and have provided an original construction from weighted automata to k-sequential weighted automata, extending the standard determinization of transducers in an intricate way. In addition, the characterization by means of a twinning property allows to derive efficient decision procedures, and all our results are also valid for word transducers.

As a complement, these results can be generalized to non finitely generated groups, using ideas similar to those developed in [9]. As future work, we plan to lift these results to other settings, like infinite or nested words. Another challenging research direction consists in considering other laws to aggregate weights of runs.

References

1. Alur, R., Cerný, P.: 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. LIPIcs, vol. 8, pp. 1–12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010)

[4] The transducer is viewed as a weighted automaton over $\mathcal{F}(B)$.

2. Alur, R., D'Antoni, L., Deshmukh, J.V., Raghothaman, M., Yuan, Y.: 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. pp. 13–22. IEEE Computer Society (2013)
3. Alur, R., Raghothaman, M.: 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. Lecture Notes in Computer Science, vol. 7966, pp. 37–48. Springer (2013)
4. Béal, M., Carton, O.: Determinization of transducers over finite and infinite words. Theor. Comput. Sci. 289(1), 225–251 (2002), http://dx.doi.org/10.1016/S0304-3975(01)00271-7
5. Berstel, J.: Transductions and context-free languages. Springer-Verlag (2013)
6. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. ACM Trans. Comput. Log. 11(4) (2010), http://doi.acm.org/10.1145/1805950.1805953
7. Choffrut, C.: Une caracterisation des fonctions sequentielles et des fonctions sous-sequentielles en tant que relations rationnelles. Theor. Comput. Sci. 5(3), 325–337 (1977), http://dx.doi.org/10.1016/0304-3975(77)90049-4
8. Choffrut, C., Schützenberger, M.P.: Décomposition de fonctions rationnelles. In: STACS 86, 3rd Annual Symposium on Theoretical Aspects of Computer Science, Orsay, France, January 16-18, 1986, Proceedings. Lecture Notes in Computer Science, vol. 210, pp. 213–226. Springer (1986)
9. Daviaud, L., Reynier, P.A., Talbot, J.M.: A Generalised Twinning Property for Minimisation of Cost Register Automata. In: 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2016. IEEE Computer Society (2016), to appear
10. Filiot, E., Gentilini, R., Raskin, J.F.: Quantitative languages defined by functional automata. Logical Methods in Computer Science 11(3:14), 1–32 (2015), http://arxiv.org/abs/0902.3958
11. Filiot, E., Gentilini, R., Raskin, J.: 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. LIPIcs, vol. 29, pp. 133–145. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2014)
12. Jecker, I., Filiot, E.: Multi-sequential word relations. In: Developments in Language Theory - 19th International Conference, DLT 2015, Liverpool, UK, July 27-30, 2015, Proceedings. Lecture Notes in Computer Science, vol. 9168, pp. 288–299. Springer (2015)
13. Lombardy, S., Sakarovitch, J.: Sequential? Theor. Comput. Sci. 356(1-2), 224–244 (2006), http://dx.doi.org/10.1016/j.tcs.2006.01.028
14. Sakarovitch, J.: Elements of Automata Theory. Cambridge University Press (2009), http://www.cambridge.org/uk/catalogue/catalogue.asp?isbn=9780521844253
15. Schützenberger, M.P.: On the definition of a family of automata. Information and Control 4 (1961)
16. Weber, A., Klemm, R.: Economy of description for single-valued transducers. Inf. Comput. 118(2), 327–340 (1995), http://dx.doi.org/10.1006/inco.1995.1071

Appendix

In all of the Appendix, A denotes a finite alphabet, $\mathbb{G}$ denotes an infinitary finitely generated group and Γ a finite set of generators of $\mathbb{G}$.

A Proofs of Section 3: Branching twinning and Lipschitz properties

A.1 Example

Example 6. The weighted automaton W_1 obtained by concatenating W_0 with itself is depicted in Figure 4 (left). This WA realizes the function f^2_{last}. As outlined in Example 5, f^2_{last} can be realized by a CRA with two registers. Indeed, the weighted automaton W_1 can be shown to satisfy the TP_2 of [9]. Figure 4 (right) shows such a 2-register machine C_1. (See [9] for more details.)

Note, however, that the two registers used in C_1 are not independent. (See the transitions reading #.) Actually, we need at least 4 independent registers to handle words like $a^n a \# a^m a$, $a^n b \# a^m a$, $a^n b \# b^m a$ and $a^n a \# b^m a$ which can be used to produce values arbitrarily far one to another. Finally, 4 independent registers are enough to realize f^2_{last} as we only have to guess one of the four combinations of last letters of the two words.

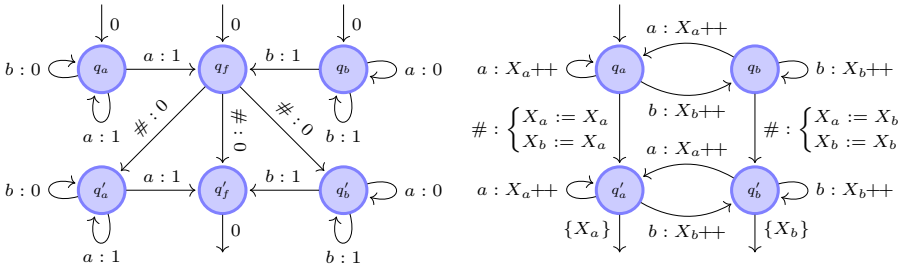


Fig. 4. A weighted automaton W_1 (left) and a cost register automaton C_1 (right) with 2 registers that compute f^2_{last}. The updates are abbreviated: $X_a{+}{+}$ means both $X_a := X_a + 1$ and $X_b := X_b$ (and conversely).

A.2 Lipschitz property of order k

We prove that k-sequential WA satisfies the Lipschitz property of order k. It is given by Proposition 1.

Proof of Proposition 1

Consider a weighted automaton W defined as the union of k sequential weighted automata $W_1, \ldots, W_k$. Let $(w_0, \alpha_0), \ldots, (w_k, \alpha_k) \in [\![W]\!]$. By the pigeon hole principle, there are $1 \leqslant j < j' \leqslant k$ and $1 \leqslant i \leqslant k$ such that $(w_j, \alpha_j) \in [\![W_i]\!]$ and $(w_{j'}, \alpha_{j'}) \in [\![W_i]\!]$. The result follows by sequentiality of W_i: there is a unique computation on the longest common prefix of w_j and $w_{j'}$ in W_i, thus:

$$d(\alpha_j, \alpha_{j'}) \leqslant 2M_W + M_W dist(w_j, w_{j'}) + 2M_W \leqslant 4M_W(dist(w_j, w_{j'}) + 1)$$

A.3 An alternative branching twinning property

Let us consider a similar definition of the BTP_k by increasing the number of loops. This leads to an alternative branching twinning property, that we will call BTP'_k, obtained from the BTP_k by requiring the property not only for k cycles, but for m cycles, for every $m \geqslant k$. We prove in Lemma 3 that BTP_k and BTP'_k are equivalent.

Definition 7. *A weighted automaton over a group $\mathbb{G}$ satisfies BTP'_k if:*

- *for all $m \geqslant k$*
- *for all states $\{q_{i,j} \mid i \in \{0, \ldots, m\}$ and $j \in \{0, \ldots, k\}\}$ with $q_{0,j}$ initial for all j,*
- *for all $\gamma_j \in \mathbb{G}$ such that $(q_{0,j}, \gamma_j) \in t_{init}$ with $j \in \{0, \ldots, k\}$,*
- *for all words $u_{i,j}$ and $v_{i,j}$ with $i \in \{1, \ldots, m\}$ and $j \in \{0, \ldots, k\}$ such that there are $k+1$ runs satisfying for all $i \in \{1, \ldots, m\}$, for all $j \in \{0, \ldots, k\}$, $q_{i-1,j} \xrightarrow{u_{i,j}|\alpha_{i,j}} q_{i,j}$ and $q_{i,j} \xrightarrow{v_{i,j}|\beta_{i,j}} q_{i,j}$ (see Figure 5),*

there are $j \neq j'$ such that for all $i \in \{1, \ldots, m\}$, if for every $1 \leqslant i' \leqslant i$, we have $u_{i',j} = u_{i',j'}$ and $v_{i',j} = v_{i',j'}$, then we have

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

Lemma 3. *For all positive integer k, a weighted automaton satisfies BTP_k if and only if it satisfies BTP'_k*

Proof. By definition, BTP'_k implies BTP_k. For the converse implication, suppose BTP'_k is not satisfied. That is, there are:

- an integer $m \geqslant k$
- states $\{q_{i,j} \mid i \in \{0, \ldots, m\}$ and $j \in \{0, \ldots, k\}\}$ with $q_{0,j}$ initial for all j,
- elements $\gamma_j \in \mathbb{G}$ such that $(q_{0,j}, \gamma_j) \in t_{init}$ with $j \in \{0, \ldots, k\}$,
- words $u_{i,j}$ and $v_{i,j}$ with $i \in \{1, \ldots, m\}$ and $j \in \{0, \ldots, k\}$ such that there are $k+1$ runs satisfying for all $i \in \{1, \ldots, m\}$, for all $j \in \{0, \ldots, k\}$, $q_{i-1,j} \xrightarrow{u_{i,j}|\alpha_{i,j}} q_{i,j}$ and $q_{i,j} \xrightarrow{v_{i,j}|\beta_{i,j}} q_{i,j}$ (see Figure 5),

such that for all $j \neq j'$ there exists $i \in \{1, \ldots, m\}$, also denoted $\zeta_{j,j'}$, such that

– for every $1 \leqslant i' \leqslant i$, we have $u_{i',j} = u_{i',j'}$ and $v_{i',j} = v_{i',j'}$, but
– $delay(\gamma_j \alpha_{1,j} \cdots \alpha_{i,j}, \gamma_{j'} \alpha_{1,j'} \cdots \alpha_{i,j'}) \neq delay(\gamma_j \alpha_{1,j} \cdots \alpha_{i,j} \beta_{i,j}, \gamma_{j'} \alpha_{1,j'} \cdots \alpha_{i,j'} \beta_{i,j'})$.

We prove now that we can only consider k loops and still preserve the property. We inductively build a partition P_i, for all $i \in \{0, \ldots, m\}$, of the set $\{0, \ldots, k\}$ of run indexes:

– $P_0 = \{\{0, \ldots, k\}\}$,
– P_{i+1} refines P_i such that j and j' remains in the same class if and only if

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

We know that P_m is the set of singleton sets. Moreover, since the partioned set contains $k+1$ elements, there are at most k indexes $i \in \{1, \ldots, m\}$ such that $P_{i-1} \neq P_i$. For all $j \neq j'$, consider i_s the smallest index such that j and j' are not in the same class in P_{i_s}. In particular, i_s is the smallest i such that

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

But then $i_s \leqslant \zeta_{j,j'}$ and thus for every $1 \leqslant i' \leqslant i_s$, we have $u_{i',j} = u_{i',j'}$ and $v_{i',j} = v_{i',j'}$. This proves that the BTP_k is not satisfied either. □

A.4 Equivalence of Lipschitz and branching twinning properties: proof of Proposition 2

BTP_k implies Lip_k. Let W denote a weighted automaton over $\mathbb{G}$ and Q its set of states.

We define an object, called a (k, m)-chunk, that has a shape that resembles to an input of the BTP'_k (with no initial outputs and additional runs at the right end after the last loops). Chunks will be assembled to build larger chunks.

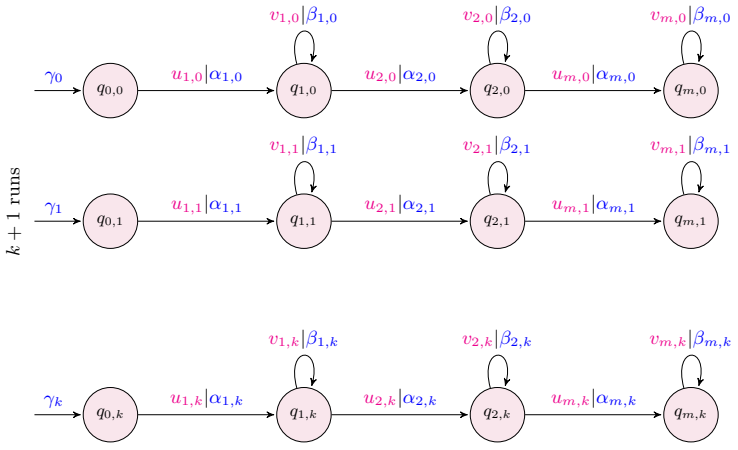


Fig. 5. Equivalent definition of the branching twinning property of order k

Definition 8 ((k,m)-chunk). *A* (k,m)-chunk *is a split of* k *runs* $(\rho_j)_{j\in\{1,\dots,k\}}$ *in* $2m+1$ *parts,* m *of which are loops, that is such that there are*

- *states* $\{q_{i,j} \mid i \in \{0,\dots,m+1\},$ *and* $j \in \{1,\dots,k\}\}$
- *words* $u_{i,j}$ *for all* $i \in \{1,\dots,m+1\}$ *and* $j \in \{1,\dots,k\}$,
- *words* $v_{i,j}$ *for all* $i \in \{1,\dots,m\}$ *and* $j \in \{1,\dots,k\}$

such that, for all $j \in \{1,\dots,k\}$, *we have*

- $\rho_{i,j} = q_{i-1,j} \xrightarrow{u_{i,j}|\alpha_{i,j}} q_{i,j}$ *and* $\rho'_{i,j} = q_{i,j} \xrightarrow{v_{i,j}|\beta_{i,j}} q_{i,j}$, *for all* $i \in \{1,\dots,m+1\}$,
- $\rho_j = \rho_{1,j}\rho'_{1,j}\cdots\rho_{m,j}\rho'_{m,j}\rho_{m+1,j}$,

The $\rho_{i,j}$ *are the backbone of the chunk and the* $\rho'_{i,j}$ *are its loops. The* backbone length *of this chunk is* $\max_{j\in\{1,\dots,k\}} \sum_{i\in\{1,\dots,m+1\}} u_{i,j}$.

The read words and produced weights inside a chunk may be the empty word and the neutral element of $\mathbb{G}$. Thus, if some of the runs are shorter in a chunk, they can be completed at will with trivial ε loops (that produce $\mathbf{1}$).

Also, one can join two chunks $\mathcal{C}_1$ and $\mathcal{C}_2$ when the last states of $\mathcal{C}_1$ are the same as the first states of $\mathcal{C}_2$. By fusing the last part of the runs of $\mathcal{C}_1$ with the first part of the runs of $\mathcal{C}_2$, we obtain a new chunk whose number of loops is the sum of the number of loops of $\mathcal{C}_1$ and $\mathcal{C}_2$, and whose backbone length is the sum of the backbone lengths of $\mathcal{C}_1$ and $\mathcal{C}_2$.

Lemma 4. *Let* ρ *be a run in* W *and let* $n = |Q|$. *If* $|in(\rho)| \geqslant |Q|$ *then there exist* $\rho_0,\dots,\rho_n$ *and* $\rho'_1,\dots,\rho'_n$, *such that* $\rho = \rho_0\rho'_1\rho_1\dots\rho'_n\rho_n$, *all the* ρ'_i *are loops and* $|in(\rho_0\dots\rho_n)| < |Q|$.

Lemma 5. *From* k *runs in* W *(*$k \geqslant 1$*), we can build a* $(k, k|Q|^k)$*-chunk whose backbone length is* $k|Q|^k$*-bounded.*

Proof. We proceed by induction on the number k of runs ρ_j ($j \in \{1,\dots,k\}$).

For the base case ($k = 1$), either $|in(\rho_1)| < |Q|$ and we can build a $(1,0)$-chunk $B = (\rho_1)$ which can be completed to a $(1,|Q|)$-chunk, or $|in(\rho_1)| \geqslant |Q|$ and by Lemma 4 we can find a split of ρ_1 that gives a $(1,|Q|)$-chunk. In both cases, the backbone length of this $(1,|Q|)$-chunk is $|Q|$-bounded.

For the induction case ($k > 1$), let $w = lcp_{j\in\{1,\dots,k\}}\{in(\rho_j)\}$ and let ρ'_j, ρ''_j for all $j \in \{1,\dots,k\}$ such that $\rho_j = \rho'_j\rho''_j$ and $in(\rho'_j) = w$. We will first build a $(k,|Q|^k)$-chunk for the k runs ρ'_j ($j \in \{1,\dots,k\}$) and then join on its right end some chunks for the ρ''_j runs ($j \in \{1,\dots,k\}$).

Either $|w| < |Q|^k$ and we can build a $(k,0)$-chunk $\left(\rho'_j\right)_{j\in\{1,\dots,k\}}$ that can be completed to a $(k,|Q|^k)$-chunk B'. Or $|w| \geqslant |Q|^k$ and, by Lemma 4 applied on W^k, for all $j \in \{1,\dots,k\}$ there exist $\rho'_{0,j},\dots,\rho'_{2n,j}$, where $n = |Q|^k$, such that $\rho'_j = \rho'_{0,j}\dots\rho'_{2n,j}$, for all $i \in \{0,\dots,n-1\}$ the $\rho'_{2i+1,j}$ are loops and $|in(\rho'_0\rho'_2\dots\rho'_{2n})| < |Q|^k$. This gives us a $(k,|Q|^k)$-chunk $B' = \left(\rho'_{i,j}\right)_{i\in\{0,\dots,2n\},j\in\{1,\dots,k\}}$. In both cases, the backbone of this $(k,|Q|^k)$-chunk is $|Q|^k$-bounded.

We partition the runs $(\rho''_j)_{j\in\{1,\dots,k\}}$ by the first letter they read. Each member of the partition is of size strictly less than k and their contained runs read inputs with a common prefix. We can apply the induction hypothesis to exhibit, for each member of the partition, a chunk that can be completed to a $(k-1,(k-1)|Q|^{k-1})$-chunk and has a backbone length bounded by $(k-1)|Q|^{k-1}$. By adequately (w.r.t. the order of the partition) joining those chunks on the right end of B' we obtain a $(k,|Q|^k+(k-1)|Q|^{k-1})$-chunk that has a backbone length bounded by $|Q|^k+(k-1)|Q|^{k-1}\leqslant k|Q|^k$. Furthermore, it can be completed to a $(k,k|Q|^k)$-chunk. □

Lemma 6. *If W satisfies BTP_k then $[\![W]\!]$ satisfies Lip_k.*

Proof. If W satisfies BTP_k then it also satisfies BTP'_k.

Let $(w_0,\alpha_0),\dots,(w_k,\alpha_k)\in[\![W]\!]$. There are initial states $p_0,\dots,p_k\in Q$, final states $q_0,\dots,q_k\in Q$, runs $\rho_0,\dots,\rho_k$, elements $\eta_j,\zeta_j,\theta_j\in\mathbb{G}$ for $j\in\{0,\dots,k\}$ such that, for all $j\in\{0,\dots,k\}$, $\alpha_j=\eta_j\zeta_j\theta_j$, $\rho_j=p_j\xrightarrow{w_j|\zeta_j}q_j$, $(p_j,\eta_j)\in t_{init}$ and $(q_j,\theta_j)\in t_{final}$.

Let $m=(k+1)|Q|^{k+1}$. By Lemma 5, we can build a $(k+1,m)$-chunk whose backbone length is bounded by m. That is there are:

- words $u_{i,j}$ for all $i\in\{0,\dots,m\},j\in\{0,\dots,k\}$ and words $v_{i,j}$ for $i\in\{1,\dots,m\},j\in\{0,\dots,k\}$ such that $w_j=u_{0,j}v_{1,j}u_{1,j}\cdots v_{m,j}u_{m,j}$ and $|u_{0,j}u_{1,j}\cdots u_{m,j}|<m$
- states $q_{i,j}$ for all $i\in\{0,\dots,m+1\},j\in\{0,\dots,k\}$ such that $q_{0,j}=p_j$ and $q_{m+1,j}=q_j$,
- elements of $\mathbb{G}$ $\alpha_{i,j}$ for all $i\in\{0,\dots,m\},j\in\{0,\dots,k\}$ and $\beta_{i,j}$ for all $i\in\{0,\dots,m\},j\in\{1,\dots,k\}$ such that $\zeta_j=\alpha_{0,j}\beta_{1,j}\alpha_{1,j}\cdots\beta_{m,j}\alpha_{m,j}$,

such that there are $k+1$ runs satisfying:

- for all $i\in\{0,\dots,m\},j\in\{0,\dots,k\}$, $q_{i,j}\xrightarrow{u_{i,j}|\alpha_{i,j}}q_{i+1,j}$ and
- for all $i\in\{1,\dots,m\},j\in\{0,\dots,k\}$, $q_{i,j}\xrightarrow{v_{i,j}|\beta_{i,j}}q_{i,j}$.

As W satisfies the BTP'_k, there are $j\neq j'$ such that for all $i\in\{1,\dots,m\}$, if for every $1\leqslant i'\leqslant i$, we have $u_{i'-1,j}=u_{i'-1,j'}$ and $v_{i',j}=v_{i',j'}$, then we have

$$\begin{aligned}&delay(\eta_j\alpha_{0,j}\cdots\alpha_{i-1,j},\eta_{j'}\alpha_{0,j'}\cdots\alpha_{i-1,j'})\\=\;&delay(\eta_j\alpha_{0,j}\cdots\alpha_{i-1,j}\beta_{i,j},\eta_{j'}\alpha_{0,j'}\cdots\alpha_{i-1,j'}\beta_{i,j'}).\end{aligned}$$

Let $i\in\{1,\dots,m\}$ be the minimum index such that $u_{i-1,j}\neq u_{i-1,j'}$ or $v_{i,j}\neq v_{i,j'}$. Then

$$\begin{aligned}&delay(\eta_j\zeta_j,\eta_{j'}\zeta_{j'})\\=\;&delay(\eta_j\alpha_{0,j}\beta_{1,j}\alpha_{1,j}\cdots\beta_{m,j}\alpha_{m,j},\eta_{j'}\alpha_{0,j'}\beta_{1,j'}\alpha_{1,j'}\cdots\beta_{m,j'}\alpha_{m,j'})\\=\;&delay(\eta_j\alpha_{0,j}\alpha_{1,j}\cdots\beta_{m,j}\alpha_{m,j},\eta_{j'}\alpha_{0,j'}\alpha_{1,j'}\cdots\beta_{m,j'}\alpha_{m,j'})\\&\vdots\\=\;&delay(\eta_j\alpha_{0,j}\cdots\alpha_{i-1,j}\beta_{i,j}\alpha_{i,j}\cdots\beta_{m,j}\alpha_{m,j},\eta_{j'}\alpha_{0,j'}\cdots\alpha_{i-1,j'}\beta_{i,j'}\alpha_{i,j'}\cdots\beta_{m,j'}\alpha_{m,j'}).\end{aligned}$$

Also we have that

$$dist(w_j, w_{j'}) = dist(u_{i-1,j}v_{i,j}u_{i,j}\cdots v_{m,j}u_{m,j}, u_{i-1,j'}v_{i,j'}u_{i,j'}\cdots v_{m,j'}u_{m,j'})$$

Therefore

$$\begin{aligned} &d(\mathbf{1}, \alpha_{i-1,j}\beta_{i,j}\alpha_{i,j}\cdots\beta_{m,j}\alpha_{m,j}) + d(\mathbf{1}, \alpha_{i-1,j'}\beta_{i,j'}\alpha_{i,j'}\cdots\beta_{m,j'}\alpha_{m,j'}) \\ &\leqslant M_W dist(u_{i-1,j}v_{i,j}u_{i,j}\cdots v_{m,j}u_{m,j}, u_{i-1,j'}v_{i,j'}u_{i,j'}\cdots v_{m,j'}u_{m,j'}) \\ &\leqslant M_W dist(w_j, w_{j'}) \end{aligned}$$

Since $|u_{0,j}u_{1,j}\cdots u_{i-2,j}| < m$ and $|u_{0,j'}u_{1,j'}\cdots u_{i-2,j'}| < m$, we have that

$$\begin{aligned} d(\alpha_j, \alpha_{j'}) &\leqslant 2M_W m + M_W dist(w_j, w_{j'}) + 2M_W \\ &\leqslant 2M_W(m+1)(dist(w_j, w_{j'}) + 1) \end{aligned}$$

This satisfies the lipschitz property of order k for $L = 2M_W((k+1)|Q|^{k+1}+1)$. □

Lip_k implies BTP_k. Consider a weighted automaton W that does not satisfy BTP_k. Let us prove that $[\![W]\!]$ does not satisfy Lip_k. It is a consequence of the following Lemma.

Lemma 7. *If W does not satisfy BTP_k, then for all positive integers L, there are $k+1$ words $w_0, \ldots, w_k$, initial states $q_0, \ldots, q_k$, with $(q_0, \gamma_0), \ldots, (q_k, \gamma_k) \in t_{init}$, states $p_0, \ldots, p_k$ and $k+1$ runs:*

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

such that for all $j \neq j'$, $d(\gamma_j\alpha_j, \gamma_{j'}\alpha_{j'}) > L(dist(w_j, w_{j'}) + 1)$.

Proof. The idea behind the proof is to consider a witness as described in Figure 2. If BTP_k is not satisfied, then one can pump the loops "the right number of times" to: (1) sufficiently increase the caley distance between the weights of the runs, (2) not increase to much the distance between the corresponding labelling words.

Let L be a positive integer. Since W does not satisfy BTP_k, then there are:

- states $\{q_{i,j} \mid i, j \in \{0, \ldots, k\}\}$ with $q_{0,j}$ initial for all j,
- pairs $(q_{0,j}, \gamma_j) \in t_{init}$ for $j \in \{0, \ldots, k\}$,
- words $u_{i,j}$ and $v_{i,j}$ with $i, j \in \{1, \ldots, k\}$ and $k+1$ runs

$$q_{i-1,j} \xrightarrow{u_{i,j}|\alpha_{i,j}} q_{i,j} \quad \text{and} \quad q_{i,j} \xrightarrow{v_{i,j}|\beta_{i,j}} q_{i,j} \quad \text{for} \quad 0 \leqslant j \leqslant k,\ 1 \leqslant i \leqslant k$$

such that for all $j \neq j'$, there is $i \in \{1, \ldots, k\}$ such that for all $1 \leqslant i' \leqslant i$, we have $u_{i',j} = u_{i',j'}$, $v_{i',j} = v_{i',j'}$ and:

$$delay(\gamma_j\alpha_{1,j}\cdots\alpha_{i,j}, \gamma_{j'}\alpha_{1,j'}\cdots\alpha_{i,j'}) \neq delay(\gamma_j\alpha_{1,j}\cdots\alpha_{i,j}\beta_{i,j}, \gamma_{j'}\alpha_{1,j'}\cdots\alpha_{i,j'}\beta_{i,j'})$$

We construct by induction (in decreasing order) a sequence of positive integers $t_k, \ldots, t_1$. Let us give the construction of t_i. Let L_i be the maximal length of

the words $u_{i+1,j}v_{i+1,j}^{t_{i+1}}\cdots u_{k,j}v_{k,j}^{t_k}$ over all $0 \leqslant j \leqslant k$. Consider T_i the set of pairs (j,j') such that for all $1 \leqslant i' \leqslant i$, we have $u_{i',j} = u_{i',j'}$, $v_{i',j} = v_{i',j'}$ and:

$$\mathit{delay}(\gamma_j\alpha_{1,j}\cdots\alpha_{i,j},\gamma_{j'}\alpha_{1,j'}\cdots\alpha_{i,j'}) \neq \mathit{delay}(\gamma_j\alpha_{1,j}\cdots\alpha_{i,j}\beta_{i,j},\gamma_{j'}\alpha_{1,j'}\cdots\alpha_{i,j'}\beta_{i,j'})$$

One can choose an integer N such that for all pairs $(j,j') \in T_i$,

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

Set $t_i = N$.

The words $w_j = u_{1,j}v_{1,j}^{t_1}\cdots u_{k,j}v_{k,j}^{t_k}$ and the corresponding runs fulfil the condition of the Lemma. Indeed, let $j \neq j'$, and i the minimal index such that $(j,j') \in T_i$. Such an index i exists by hypothesis. For $\ell \in \{j,j'\}$, set $\alpha_\ell = \alpha_{1,\ell}(\beta_{1,\ell})^{t_1}\alpha_{2,\ell}\cdots\alpha_{k,\ell}(\beta_{k,\ell})^{t_k}$ and $\overline{\alpha}_\ell = \alpha_{1,\ell}\alpha_{2,\ell}\cdots\alpha_{i-1,\ell}\alpha_{i,\ell}(\beta_{i,\ell})^{t_i}$.

We have:

$$\begin{aligned}
&\mathit{delay}(\gamma_j\alpha_j,\gamma_{j'}\alpha_{j'})\\
&= \mathit{delay}(\gamma_j\overline{\alpha}_j\alpha_{i+1,j}\cdots\alpha_{k,j}(\beta_{k,j})^{t_k},\gamma_{j'}\overline{\alpha}_{j'}\alpha_{i+1,j'}\cdots\alpha_{k,j'}(\beta_{k,j'})^{t_k})\\
&= (\alpha_{i+1,j}(\beta_{i+1,j})^{t_{i+1}}\cdots\alpha_{k,j}(\beta_{k,j})^{t_k})^{-1}\mathit{delay}(\gamma_j\overline{\alpha}_j,\gamma_{j'}\overline{\alpha}_{j'})\alpha_{i+1,j'}(\beta_{i+1,j'})^{t_{i+1}}\cdots\alpha_{k,j'}(\beta_{k,j'})^{t_k}
\end{aligned}$$

Moreover, $\mathit{dist}(w_j,w_{j'}) \leqslant 2L_i$ by definition of L_i. Then:

$$d(\gamma_j\alpha_j,\gamma_{j'}\alpha_{j'}) > 2L_i(M_W+L)+L-2M_WL_i \geqslant L(\mathit{dist}(w_j,w_{j'})+1)$$

□

Lemma 8. *If W does not satisfy BTP_k then $[\![W]\!]$ does not satisfy Lip_k.*

Proof. Let L be a positive interger and $L' = L(2N+1)$ where N is the number of states of W. By Lemma 7, there are $k+1$ words $w_0,\ldots,w_k$, initial states $q_0,\ldots,q_k$, with $(q_0,\gamma_0),\ldots,(q_k,\gamma_k) \in t_{\mathit{init}}$, states $p_0,\ldots,p_k$ and $k+1$ runs:

$$q_j \xrightarrow{w_j|\alpha_j} p_j \quad \text{for all } j \in \{0,\ldots,k\},$$

such that for all $j \neq j'$, $d(\gamma_j\alpha_j,\gamma_{j'}\alpha_{j'}) > L'(\mathit{dist}(w_j,w_{j'})+1)$. These $k+1$ runs can be completed into accepting runs ending in accepting states $r_0,\ldots,r_k$ labeled by $w_0w_0',\ldots,w_kw_k'$ with weights $\alpha_0\alpha_0',\ldots,\alpha_k\alpha_k'$ such that for all $0 \leqslant j \leqslant k$, $|w_j'| \leqslant N$. Consider $(r_0,\beta_0),\ldots,(r_k,\beta_k) \in t_{\mathit{final}}$.

For all $j \neq j'$,

$$\begin{aligned}
d(\gamma_j\alpha_j\alpha_j'\beta_j,\gamma_{j'}\alpha_{j'}\alpha_{j'}'\beta_{j'}) &> L'(\mathit{dist}(w_j,w_{j'})+1)+2NM_W+2M_W\\
&\geqslant L(\mathit{dist}(w_j,w_{j'})+2N+1)\\
&\geqslant L(\mathit{dist}(w_jw_j',w_{j'}w_{j'}')+1)
\end{aligned}$$

□

B Proofs of Section 4: Main result

We now prove that the branching twinning property of order k implies the k-sequentiality (Proposition 3). It is developed in the rest of this Section.

Proof of Proposition 3

We present here the elements missing from the sketch of proof. First, we expose the subset construction with delays, together with the proofs of its properties, adapted from [4] to fit our settings. Then, we present in details the proof of the properties $\mathbf{P}_1$ and $\mathbf{P}_2$. Finally, we give the formal construction of the sequential weighted automata $\overline{V}_i$, $1 \leqslant i \leqslant k$, whose union is equivalent to W.

Subset construction with delays. Given $W = (Q, t_{init}, t_{final}, T)$ a weighted automaton, we construct an equivalent infinite sequential automaton $D_W = (Q', t'_{init}, t'_{final}, T')$ as follows. The states of D_W are the subsets of $Q \times \mathbb{G}$. For every $S \in Q'$ and $a \in A$, we define the single transition starting from S, and labelled by the input a, as follows.

1. First, the elements of S are updated with respect to W. Let
$$S' = \{(q, \beta\gamma) | \exists p \in Q \text{ s.t. } (p, \beta) \in S, (p, a, \gamma, q) \in T\} \subseteq Q \times \mathbb{G}.$$
2. Then a reference pair $(q, \alpha) \in S'$ is picked, and S' is normalized with respect to it. Let $(p, \alpha) = \mathsf{select}(S') \in Q \times \mathbb{G}$, where select is a choice function. Let
$$S'' := \{(q, \alpha^{-1}\beta) | (q, \beta) \in S'\} \subseteq Q \times \mathbb{G}.$$
Note that, in particular, $(q, \mathbf{1}) \in S''$.

Then $(S, a, \alpha, S'') \in T'$.

For every state $S \in Q'$, let W_S denote the weighted automaton obtained by replacing the initial output relation of W with S, i.e., $W_S = (Q, S, t_{final}, T)$. Note that $W_{t_{init}} = W$. Before defining the initial relation and the final relation of D_W, let us prove two lemmas that follow from the definition of T'.

Lemma 9. *For every accessible state S of D_W that is not the initial state, there exists $q \in Q$ such that $(q, \mathbf{1}) \in S$.*

Proof. This follows immediately from the definition of T'.

Lemma 10. *Let $S \subset Q \times \mathbb{G}$ and let $u \in A^*$, and consider the run $S \xrightarrow{u|\alpha} S'$ of D. Then*
$$S' = \{(q, \beta) | \text{ there exists a path } \rho :\xrightarrow{\gamma} p \xrightarrow{u|\delta} q \text{ in } W_S \text{ s.t. } \alpha\beta = \gamma\delta\}.$$

Proof. The proof is done by induction over the length of the input word u. If $u = \epsilon$, then $S = S'$, $\alpha = \mathbf{1}$, and the result follows immediately. Now suppose that $u = va$ for some $v \in A^*$ and $a \in A$, and that the lemma is true for the input word v. Note that the initial run can be split as follows.

$$S \xrightarrow{v|\alpha_1} S_1 \xrightarrow{a|\alpha_2} S'.$$

By induction hypothesis,

$$S_1 = \{(q_1, \beta_1) | \text{ there exists a path } \rho :\xrightarrow{\gamma} p \xrightarrow{v|\delta_1} q_1 \text{ in } W_S \text{ s.t. } \alpha_1\beta_1 = \gamma\delta_1\}.$$

Moreover, by definition of the transition relation of D_W,

$$S' = \{(q, \alpha_2^{-1}\beta_1\gamma_2) | \exists q_1 \in Q \text{ s.t. } (q_1, \beta_1) \in S_1, (q_1, a, \gamma_2, q) \in T\} \subseteq Q \times \mathbb{G}.$$

The desired result follows, since we obtain, by applying those two equalities,

$$\begin{aligned}
S' &= \{(q, \alpha_2^{-1}\beta_1\gamma_2) | \exists q_1 \in Q \text{ s.t. } (p_1, \beta_1) \in S_1, (q_1, a, \gamma_2, q) \in T\} \\
&= \{(q, \alpha_2^{-1}\beta_1\gamma_2) | \exists \rho :\xrightarrow{\gamma} p \xrightarrow{v|\delta_1} q_1 \text{ in } W_S \text{ s.t. } \alpha_1\beta_1 = \gamma\delta_1 \text{ and } (q_1, a, \gamma_2, q) \in T\} \\
&= \{(q, \alpha_2^{-1}\beta_1\gamma_2) | \exists \rho :\xrightarrow{\gamma} p \xrightarrow{u|\delta} q \text{ in } W_S \text{ s.t. } \alpha_1\beta_1\gamma_2 = \gamma\delta\} \\
&= \{(q, \beta) | \exists \rho :\xrightarrow{\gamma} p \xrightarrow{u|\delta} q \text{ in } W_S \text{ s.t. } \alpha_1\alpha_2\beta = \gamma\delta\} \\
&= \{(q, \beta) | \exists \rho :\xrightarrow{\gamma} p \xrightarrow{u|\delta} q \text{ in } W_S \text{ s.t. } \alpha\beta = \gamma\delta\}.
\end{aligned}$$

We can now define the initial relation and the final relation of D_W. Since Lemma 10 guarantees that, starting from state S, D_W accurately simulates the weighted automaton W_S, and $W_{t_{init}} = W$, by setting $t'_{init} = (t_{init}, \mathbf{1})$, and by setting

$$t'_{final} = \{(S, \alpha\beta) | \text{ there exists } q \in Q \text{ s.t. } (q, \alpha) \in S \text{ and } (q, \beta) \in t_{final}\},$$

we ensure that D_W is equivalent to W.

Proof of $\mathbf{P_1}$ We now demonstrate $\mathbf{P}_1$, using Lemma 10 and the fact that W is ℓ-valued.

Lemma 11. *Let S be an accessible state of D_W. Then $|S| \leqslant \ell|Q|$.*

Proof. For every $q \in Q$, let $n_q \in \mathbb{N}$ denote the number of times that q appears in S, i.e., $n_q = |\{\beta \in \mathbb{G} | (q, \beta) \in S\}|$. Using the fact that W is ℓ-valued, we now prove that $n_q \leqslant \ell$, which implies the desired result, since $n = \Sigma_{q \in Q} n_q$.

Let $\beta_1, \ldots, \beta_{n_q}$ be an enumeration of the elements of $\{\beta \in \mathbb{G} | (q, \beta) \in S\}$. Since S is accessible, there exists a run $t_{init} \xrightarrow{u|\alpha} S$ in D_W, and, by Lemma 10, for every $0 \leqslant j \leqslant n_q$, since $(q, \beta_j) \in S$, there exists an initial run

$$\rho_j :\xrightarrow{\gamma_j} p_j \xrightarrow{u|\delta_j} q$$

of $W_{t_{init}} = W$ such that $\alpha\beta_j = \gamma_j\delta_j$. Moreover, since W is trim by supposition, it admits a final run

$$\rho : q \xrightarrow{v|\delta} q_f \xrightarrow{\delta_f} .$$

Then, for every $0 \leqslant j \leqslant n_q$, $(uv, \alpha\beta_j\delta\delta_f) = (uv, \gamma_j\delta_j\delta\delta_f) \in [\![C]\!]$. Since the β_j are distinct elements of the group $\mathbb{G}$, so are the elements $\alpha\beta_j\delta\delta_f \in \mathbb{G}$, hence, since W is ℓ-valued by supposition, $n_q \leqslant \ell$.

Proof of $\mathbf{P_2}$ In order to prove $\mathbf{P}_2$, we first introduce a new definition.

Definition 9. *For every state $S \in Q'$, let the rank of S, denoted by $\mathsf{rank}(S)$, be the minimal integer k' such that W_S satisfies the $BTP_{k'}$, where W_S is the weighted automaton obtained by replacing the initial output relation of W with S, i.e., $W_S = (Q, S, t_{final}, T)$.*

Lemma 12. *Let $N_W = 2M_W|Q|^{\ell|Q|}$. Let S be an accessible state of D_W that contains a pair (q, α) such that $|\alpha| > N_W$. Then there exists a partition of S into two subsets S' and S'' such that $\mathsf{rank}(S') + \mathsf{rank}(S'') \leqslant k$.*

Proof. By Lemma 9, we know that there exists a pair $(q_1, \mathbf{1}) \in S$. Moreover, by supposition, there exists a pair $(q_2, \alpha_2) \in S$ such that $|\alpha_2| > N_W$.

Let $(q_3, \alpha_3), \ldots, (q_m, \alpha_m)$ be an enumeration of the elements of S distinct from $(q_1, \mathbf{1})$ and (q_2, α_2). By Lemma 11, $m \leqslant \ell|Q|$. Since S is accessible, there exists a run $t_{init} \xrightarrow{u|\alpha} S$ in D_W. Then, by Lemma 10, for every $1 \leqslant j \leqslant m$, since $(q_j, \alpha_j) \in S$, there exist a run

$$\rho_j : \xrightarrow{\gamma_{0.j}} q_{0.j} \xrightarrow{u|\gamma_j} q_j$$

of W over u such that $\alpha\alpha_j = \gamma_{0.j}\gamma_j$.

The proof is now done in two steps. First, we expose a decomposition of u into three words $wv_\tau w'$ such that every run ρ_j loops over v_τ, and the delay between the outputs of the runs ρ_1 and ρ_2 is modified along v_τ. This allows us to define a partition $\{S', S''\}$ of S, splitting the elements (q_j, α_j) of S depending on whether or not the delay between ρ_1 and ρ_j changes along v_τ. Then, we prove that $\mathsf{rank}(S') + \mathsf{rank}(S'') \leqslant k$, using the following idea. Let $r' = \mathsf{rank}(S') - 1$, $r'' = \mathsf{rank}(S'') - 1$. By combining a witness of the non satisfaction of the $BTP_{r'}$ by $W_{S'}$ and a witness of the non satisfaction of the $BTP_{r''}$ by $W_{S''}$, we build a witness of the non satisfaction of the $BTP_{r'+r''+1}$ by W_S. This implies that

$$\mathsf{rank}(S') + \mathsf{rank}(S'') - 1 = r' + r'' + 1 < k,$$

and the desired result follows.

1. By applying Lemma 4 to the product of m copies of W, we obtain a subdivision $u_0v_1u_1 \cdots v_{|Q|^m}u_{|Q|^m}$ of the word u such that each run ρ_j loops over each input v_s, and $|u_0 \cdots u_{|Q|^m}| < |Q|^m \leqslant |Q|^{\ell|Q|}$.

Note that the distance between the outputs of ρ_1 and ρ_2 after reading the input u is

$$d(\alpha_1, \alpha_2) = d(\mathbf{1}, \alpha_2) > N_W = 2M_W|Q|^{\ell|Q|}.$$

In order for the distance to increase by at least N_W, there has to exist an integer $1 \leqslant \tau \leqslant m$ such that the delay between the outputs of ρ_1 and ρ_2 changes along v_τ, since M_W is greater than or equal to the maximal size of the output of a transition of W, and $|u_0 \cdots u_m| < |Q|^{\ell|Q|}$. Let $w = u_0 v_1 u_1 \cdots v_{\tau-1} u_{\tau-1}$, $w' = u_\tau v_{\tau+1} u_{\tau+1} \cdots v_{|Q|^m} u_{|Q|^m}$, and for every $1 \leqslant j \leqslant m$, consider the following decomposition of the run ρ_j:

$$\rho_j : \xrightarrow{\gamma_{0,j}} q_{0,j} \xrightarrow{w|\gamma_{1,j}} p_j \xrightarrow{v_\tau|\gamma_{2,j}} p_j \xrightarrow{w'|\gamma_{3,j}} q_j.$$

Let S' be the set of pairs (q_j, α_j) corresponding to the indices j such that the delay between ρ_1 and ρ_j stays the same along v_τ, i.e.,

$$delay(\gamma_{0,1}\gamma_{1,1}, \gamma_{0,j}\gamma_{1,j}) = delay(\gamma_{0,1}\gamma_{1,1}\gamma_{2,1}, \gamma_{0,j}\gamma_{1,j}\gamma_{2,j}),$$

Then, let $S'' = S \setminus S'$. By definition of v_τ, S'' is not empty since it contains (q_2, α_2), hence $\{S', S''\}$ is a partition of S.

2. Let $r' = \mathsf{rank}(S') - 1$, $r'' = \mathsf{rank}(S'') - 1$, and $r = r' + r''$. By definition of the rank, there exists an unsatisfied instance of the $BTP_{r'}$ over the weighted automaton $W_{S'}$. Let $\phi_0, \ldots, \phi_{r'}$ be the runs of $W_{S'}$ forming this instance. Similarly, there exists an unsatisfied instance of the $BTP_{r''}$ over the weighted automaton $W_{S''}$. Let $\phi_{r'+1}, \ldots, \phi_{r+1}$ be the runs of $W_{S''}$ forming this instance. For every $1 \leqslant \eta \leqslant r+1$, we add loops over the empty word at the end of the run ϕ_η in order for it to contain exactly r loops, yielding the run

$$\phi'_\eta : \xrightarrow{\delta_{0,\eta}} p_{0,\eta} \xrightarrow{w_{1,\eta} \mid \delta_{1,\eta}} \underset{\curvearrowright\, x_{1,\eta} \mid \zeta_{1,\eta}}{p_{1,\eta}} \xrightarrow{w_{2,\eta} \mid \delta_{2,\eta}} \cdots \xrightarrow{w_{r,\eta} \mid \delta_{r,\eta}} \underset{\curvearrowright\, x_{r,\eta} \mid \zeta_{r,\eta}}{p_{r',\eta}}.$$

Note that, since both S' and S'' are subsets of S, each ϕ'_η can be seen as a run over the weighted automaton W_S. By extending those runs on the left, we now construct an instance of the BTP_{r+1} for W that is not satisfied. For every $0 \leqslant \eta \leqslant r$, $(p_{0,\eta}, \delta_{0,\eta}) \in S$, therefore there exists $1 \leqslant j_\eta \leqslant m$ such that $(p_{0,\eta}, \delta_{0,\eta}) = (q_{j_\eta}, \alpha_{j_\eta})$, hence we can compose the run ρ_{j_η} with ϕ_η, as follows.

$$\psi_\eta : \xrightarrow{\gamma_{0,j_\eta}} q_{0,j_\eta} \xrightarrow{w \mid \gamma_{1,j_\eta}} \underset{\curvearrowright\, v_\tau \mid \gamma_{2,j_\eta}}{p_{j_\eta}} \xrightarrow{w_\eta \mid \gamma_\eta} \underset{\curvearrowright\, x_{1,\eta} \mid \zeta_{1,\eta}}{p_{1,\eta}} \xrightarrow{w_{2,\eta} \mid \delta_{2,\eta}} \cdots \xrightarrow{w_{r,\eta} \mid \delta_{r,\eta}} \underset{\curvearrowright\, x_{r,\eta} \mid \zeta_{r,\eta}}{p_{r,\eta}},$$

where $w_\eta = v_\tau w' w_{1,\eta}$ and $\gamma_\eta = \gamma_{2,j_\eta}\gamma_{3,j_\eta}\delta_{1,\eta}$. Note that ψ_η is a run of W. In order to conclude the proof, we need to prove that the instance of the BTP_{r+1} for W formed by the runs $\psi_0, \ldots, \psi_r$ is not satisfied. For every $0 \leqslant \eta < \eta' \leqslant r+1$, we now expose a loop differentiating ψ_η and $\psi_{\eta'}$, i.e., a loop along which the delay between the outputs of ψ_η and $\psi_{\eta'}$ changes, and that occurs on the part of the runs where the inputs are identical. We consider three possibilities.

– If $0 \leqslant \eta \leqslant r' < \eta' \leqslant r+1$, then $(q_{j_\eta}, \alpha_{j_\eta}) \in S'$ and $(q_{j_{\eta'}}, \alpha_{j_{\eta'}}) \in S''$, and, by definition of the partition $\{S', S''\}$ of S,

$$delay(\gamma_{0.j_\eta}\gamma_{1.j_\eta}, \gamma_{0.j_{\eta'}}\gamma_{1.j_{\eta'}}) \neq delay(\gamma_{0.j_\eta}\gamma_{1.j_\eta}\gamma_{2.j_\eta}, \gamma_{0.j_{\eta'}}\gamma_{1.j_{\eta'}}\gamma_{2.j_{\eta'}}),$$

Therefore, the first loop of ψ_η and $\psi_{\eta'}$ can be used to differentiate them.
– If $0 \leqslant \eta < \eta' \leqslant r'$, since the runs $\phi_0, \ldots, \phi_{r'}$ form a non satisfied instance of the $BTP_{r'}$ for $W_{S'}$, we can find a loop differentiating ϕ_η and $\phi_{\eta'}$, and use the corresponding loop to differentiate ψ_η and $\psi_{\eta'}$.
– If $r'+1 \leqslant \eta < \eta' \leqslant r+1$, since the runs $\phi_{r'+1}, \ldots, \phi_{r+1}$ form a non satisfied instance of the $BTP_{r''}$ for $W_{S''}$, we can find a loop differentiating ϕ_η and $\phi_{\eta'}$, and use the corresponding loop to differentiate ψ_η and $\psi_{\eta'}$.

Finally, $\mathbf{P}_2$ is obtained as a corollary.

Corollary 1. *Let S be an accessible state of D_W that contains a pair (q, α) such that $|\alpha| > N_W$. Then W_S is k-sequential.*

Proof. By the previous lemma, there exists a partition of S into two subsets S' and S'' such that $\mathsf{rank}(S') + \mathsf{rank}(S'') \leqslant k$. Note that $\mathsf{rank}(S') \geqslant 1$ and $\mathsf{rank}(S'') \geqslant 1$, hence $\mathsf{rank}(S'') < k$ and $\mathsf{rank}(S') < k$ Therefore, the induction hypothesis can be applied, proving that $W_{S'}$ is $\mathsf{rank}(S')$-sequential, and $W_{S''}$ is $\mathsf{rank}(S'')$-sequential. Finally, as S is equal to the union of S' and S'', W_S is equivalent to the union of $W_{S'}$ and $W_{S''}$. Therefore W_S is k-sequential.

Final construction We construct k sequential weighted automata $\overline{V}_1, \ldots, \overline{V}_k$ whose union is equivalent to W. Let U denote the set containing the accessible states S of D_W that contain only pairs (q, α) satisfying $|\alpha| \leqslant N_W$. Moreover, let U' be the set of states of D_W accessible in one step from U, i.e.,

$$U' = \{S' | \exists S \in U, a \in A, \alpha \in \mathbb{G} \text{ s.t. } (S, a, \alpha, S') \in T'\}.$$

As there are only finitely many $\alpha \in \mathbb{G}$ such that $|\alpha| \leqslant N_W$, $\mathbf{P}_1$ implies that U is finite. Note that this implies the finiteness of U'. By $\mathbf{P}_2$, for every state $S \in U'$ that is not in U, W_S can be expressed as the union of k sequential weighted automata $V_i(S)$, with $1 \leqslant i \leqslant k$. For every $1 \leqslant i \leqslant k$, let $\overline{V}_i$ be defined as the union of D_W restricted to the states $S \in U$, and all the $V_i(S')$, for $S' \in U' \setminus U$, with the two following differences. First, the only initial state of $\overline{V}_i$ is the initial state of D_W. Second, for every transition (S, a, α, S') of D_W between states $S \in U$ and $S' \in U' \setminus U$, we add a transition $(S, a, \alpha\alpha_{S'.i}, q_{S'.i})$, where $\{(q_{S'.i}, \alpha_{S'.i})\}$ is the initial relation of $V_i(S')$.

Proposition 6. *The weighted automaton W is equivalent to the union of the $\overline{V}_i$, $1 \leqslant i \leqslant k$.*

Proof. We prove that D_W is equivalent to the union of the $\overline{V}_i$, $1 \leqslant i \leqslant k$, which implies the desired result, since W is equivalent to D_W.

First, we show that the relation defined by D_W is included into the union of the $[\![\overline{V}_i]\!]$. Let $\rho :\to S_0 \xrightarrow{u|\alpha} S_f \xrightarrow{\beta}$ be an accepting run of D_W. We now expose an integer $i \in \{1, \ldots, k\}$ such that the output of the run of $\overline{V}_i$ over the input u is $\alpha\beta$. If all the states visited by ρ are in U, ρ is present in each $\overline{V}_i$, and we are done. Otherwise, let us split ρ as follows.

$$\rho :\to S_0 \xrightarrow{u_1|\alpha_1} S \xrightarrow{a|\alpha'} S' \xrightarrow{u_2|\alpha_2} S_f \xrightarrow{\beta},$$

where S' is the first state encountered along ρ that is not in U. Then $S' \in U'$. Moreover, by Lemma 10, the definition of the final relation of D_W, and the fact that $W_{S'}$ is equivalent to the union of the $V_i(S')$, there exists $1 \leqslant i \leqslant k$ and a run

$$\rho' :\xrightarrow{\gamma_0} q_0 \xrightarrow{u_2|\gamma_1} q_f \xrightarrow{\gamma_2}$$

in $V_i(S')$ such that $\gamma_0\gamma_1\gamma_2 = \alpha_2\beta$. Then the run

$$\to S_0 \xrightarrow{u_1|\alpha_1} S \xrightarrow{a|\alpha'\gamma_0} q_0 \xrightarrow{u_2|\gamma_1} q_f \xrightarrow{\gamma_2}$$

over the input u is in $\overline{V}_i$, and the associated output is $\alpha_1\alpha'\gamma_0\gamma_1\gamma_2 = \alpha\beta$, which proves the desired result.

Conversely, we can prove, using similar arguments, that for every $1 \leqslant i \leqslant k$, $[\![\overline{V}_i]\!]$ is included into $[\![D_W]\!]$, which concludes the proof.

C Proofs of Section 5: Cost register automata with independent registers

Proof of Proposition 4.

Proof. From a k-sequential weighted automaton $\cup_{i\in\{1,\ldots,k\}} W_i$, we can build a CRA C_i with 1 register for each of the W_i. We obtain a CRA with k independent registers by making the product of those C_i. From a CRA C with k independent registers $\{X_i\}_{i\in\{1,\ldots,k\}}$, for each i we can produce a trim projection of C on the register X_i. Each of these 1-register machines can be expressed as a weighted automaton, and their union is a k-sequential weighted automaton. □

D Proofs of Section 6: The case of transducers

Notations. Let $\mathbb{M} = (M, \otimes, \mathbf{1})$ be a monoid. 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 $O^{\leqslant k} = O^{<k} \cup O^k$.

Moreover, if $\mathbb{M}$ is a group and $O \subseteq M$, O^{-1} denotes the set $\{\alpha^{-1} \mid \alpha \in O\}$.

Proof of Proposition 5. Let us sketch the main ideas. Consider a cost register automaton C over $\mathcal{F}(B)$ that computes a relation in $A^* \times B^*$. One can prove that there is a bound N' such that along the runs of C, 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 the fact that C uses k independent registers.

Let us give a formal proof now. Let $(Q, q_{init}, \mathcal{X}, \delta, \mu)$ denote a register automaton over $\mathcal{F}(B) = (B \cup B^{-1})^*$ computing a relation $F \subseteq A^* \times B^*$. The Cayley distance is defined in $(\mathcal{F}(B), B)$.

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

- m is the maximum of the set:

$$\{|\alpha| \ |\delta(q, a) = (p, h), h(Y) = (X, \alpha), q, p \in Q, a \in A, X, Y \in \mathcal{X}\}$$

- s is the maximum of the set:

$$\{|\alpha| \mid (q, X, \alpha) \in \mu, X \in \mathcal{X}, q \in Q\}$$

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

The set of states of C' 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 relation μ' is defined by the triplets $((q, r), X, r(X)\alpha)$ for all $(q, X, \alpha) \in \mu$.

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

Lemma 13. *For all words w, there is a run in C on w from (q_{init}, ν_{init}) to some (q, ν) if and only if there is a run in C' 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 C on w from (q_{init}, ν_{init}) to (q, ν). Set (q', ν') the configuration such that there is a run in C on w' from (q_{init}, ν_{init}) to (q', ν') and $\delta(q', a) = (q, g)$. By induction hypothesis, there is a run in C' 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 C 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 C 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 C' 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 C' 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 C 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 C 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. □

By the previous lemma, we can now prove that $[\![C]\!] = [\![C']\!]$. Consider a run in C on a word w from (q_{init}, ν_{init}) to (q, ν) and $(q, Y, \alpha) \in \mu$. Then, by Lemma 13, there is a run in C' 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, $((q, r), Y, r(Y)\alpha) \in \mu'$.

Conversely, suppose that there is a run in C' on w from $((q_{init}, r_{init}), \nu_{init})$ to $((q, r), \sigma)$ and a register Y such that $((q, r), Y, r(Y)\alpha) \in \mu'$. Then by Lemma 13, there is a run in C 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, $(q, Y, \alpha) \in \mu$.

Thus, C' computes the same relation as C. Moreover, by construction, C' uses k independent registers. 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 $\alpha \in \mathcal{F}(B)$ such that $(q, X, \alpha) \in \mu$,

– $(q, X) \in E$ 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 $\alpha \in \mathcal{F}(B)$.

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

Lemma 14. *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 $\nu(X) = c \in \mathcal{F}(B)$. 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 $\nu'(Y) = c\alpha$, $(q', Y, \beta) \in \mu$ and thus $c\alpha\beta \in B^*$. Finally, $c \in B^*(B \cup B^{-1})^{\leqslant N}$. □

To prove that the updates of C' 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}) by 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 $v' \in \mathcal{F}(B)$, and $v \notin (B \cup B^{-1})^{<N}$. By induction hypothesis and completing the run, there is a word w such that $(w, u'ua^{-1}vv') \in [\![C]\!]$ 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 $v' \in \mathcal{F}(B)$, $v \notin (B \cup B^{-1})^{<N}$ and $v' \in (B \cup B^{-1})^{\leqslant N}$. That is a contradiction.
– The word u is empty. Since $v \neq av'$ for all $v' \in \mathcal{F}(B)$, $v \notin (B \cup B^{-1})^{<N}$ 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 C, 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 14. ($\star$)

Finally, we need to prove that the output relation of C' also uses only elements in B^*. Thus, let us prove that for all accessible (q, r) and $(q, X, \alpha) \in \mu$, $r(X)\alpha$ 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, $v \in \mathcal{F}(B), v \neq av'$ for some $v' \in \mathcal{F}(B)$. Let us treat the two following cases:

– The word u ends with a letter $b \neq a$. In this case, by completing the run, one of the images of a word is of the form $\beta r(X)\alpha$ with $\beta \in B^*$. But in this case, $\beta r(X)\alpha$ would not belong to B^*. Thus, C would not compute a relation in $A^* \times B^*$, that is a contradiction.

– The word u is empty. By completing the run into an accepting run in C', 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 $(\star)$.

E Proofs of Section 7: Decidability

First observe that when we described the skeleton of a counter-example, we claimed that we can look for a counter-example with more than k loops. This claim is exactly Lemma 3, proven in Appendix, Section A.3.

Case of commutative groups. We omitted the proof of the lower bound, so we give details now.

Lemma 15. *Over the group $(\mathbb{Z}, +)$, the BTP_k problem is* PSPACE*-hard (k given in unary).*

Proof. We present a reduction of the emptiness of k deterministic finite state automata to the BTP_k problem, using similar ideas to a lower bound proved in [3]. Let $D_1, \ldots, D_k$ be k deterministic finite state automata over some alphabet A. Let $\#$ and \$ be two fresh symbols not in A. For each i, we build a deterministic weighted automaton W_i over the group $(\mathbb{Z}, +)$. The semantics of W_i is defined as:

$$[\![W_i]\!] = \{(u\#\$^m, i * m) \mid u \in \mathsf{dom}(D_i), m \in \mathbb{N}\}$$

We consider now the weighted automaton defined as $W = \bigcup_i W_i$. We claim that W does not satisfy BTP_{k-1} iff the intersection of the languages defined by the D_i's is non-empty.

First, suppose that the intersection of the languages is empty. We describe the construction of a CRA with $k-1$ independent registers $X_1, \ldots, X_{k-1}$ that realizes the relation $[\![W]\!]$. We consider the deterministic finite state automaton D obtained as the product of the D_i's. We add to D the set of states $Q' = 2^{\{1,\ldots,k\}}$. From each state $(q_1, \ldots, q_k)$ of D, we add a transition on symbol $\#$ that goes to the state $\{i \mid q_i \text{ is a final state of } D_i\} \in Q'$. As the intersection of the languages is empty, every reachable state $q' \in Q'$ is such that $|q'| \leqslant k-1$. We can thus restrict D to elements of Q' of size at most $k-1$. Last, given a state $q' \in Q'$, we add a self-loop labelled by the symbol \$.

We describe now the updates of the registers. The only non-trivial updates are on self-loops of states in Q'. Let $q' = \{i_1, \ldots, i_m\} \in Q'$. As explained above, we have $m \leqslant k-1$. We suppose that indices i_j's are sorted by increasing order. Intuitively, we use register X_j to represent index i_j, so we define the update $X_j := X_j * i_j$. The final output function of state q' is simply defined as the set $\{X_1, \ldots, X_m\}$. It is easy to verify that this CRA realizes the relation $[\![W]\!]$.

Conversely, suppose now that the intersection of the languages is non-empty, and let u be a word in this intersection. We proceed by contradiction, and

suppose that there exists an equivalent CRA with $k-1$ independent registers. By definition, the relation $\{(u\#\$^m, i*m) \mid m \geqslant 0, i \in \{1,\dots,k\}\}$ is included in $[\![W]\!]$. In particular we can find input words with k output values that are pairwise arbitrarily far. It is easy to show that a CRA with $k-1$ registers is unable to accept such a relation, hence a contradiction. □

Case of transducers. We describe the condition on the skeleton that we can ensure when we are considering transducers.

Let L be a positive integer. We say that two runs ρ_1 and ρ_2 on the same input word u are L-close if for every prefix u' of u, the restrictions ρ_1' and ρ_2' of the two runs on the input u' are such that $d(\alpha_1, \alpha_2) \leqslant L$, where α_i is the output word produced by the run ρ_i'.

Lemma 16 (Nice shape). *Let T be a transducer, then T violates BTP_k iff there exists a counter example given as follows: there are*

- *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 for all j,*
- *words $\{u_{i,j}\}_{0\leqslant i\leqslant m, 0\leqslant j\leqslant k}$ and $\{v_{i,j}\}_{1\leqslant i\leqslant m, 0\leqslant j\leqslant k}$ such that there are $k+1$ runs satisfying for all $0 \leq j \leq k$, for all $1 \leq i \leq m$, $q_{i-1,j} \xrightarrow{u_{i,j}|\alpha_{i,j}} q_{i,j}$ and $q_{i,j} \xrightarrow{v_{i,j}|\beta_{i,j}} q_{i,j}$*

and such that for all $0 \leqslant j < j' \leqslant k$, there exists $1 \leqslant i \leqslant m$ such that for all $1 \leqslant i' \leqslant i$, we have $u_{i',j} = u_{i',j'}$ and $v_{i',j} = v_{i',j'}$, and

a) either $|\beta_{i,j}| \neq |\beta_{i,j'}|$,
b) or $|\beta_{i,j}| = |\beta_{i,j'}| \neq 0$, the words $\alpha_{1,j} \dots \alpha_{i,j}$ and $\alpha_{1,j'} \dots \alpha_{i,j'}$ have a mismatch, and the runs $q_{0,j} \xrightarrow{u_1 \dots u_i} q_{i,j}$ and $q_{0,j'} \xrightarrow{u_1 \dots u_i} q_{i,j'}$ are $M_T.n^{k+1}$-close.

Proof (Sketch). The reverse implication is trivial, so we focus on the direct one. We consider a counter-example to the BTP_k and aim at deriving a counter example satisfying the above properties.

Let us consider a pair (j, j') of runs indices with $0 \leqslant j < j' \leqslant k$. By definition, there exists an index i (satisfying $i \leqslant \chi(j, j')$) such that the loop i induces a different delay. If this is due to the length of the output words, then we are done as we are in case *a*).

Otherwise, let us assume that every loop of index at most $\chi(j, j')$ has output words of same length on components j and j'. Consider a loop that induces different delays. Then it induces a mismatch between the non-empty output words of the loop on components j and j'. This loop can be unfolded to move the mismatch on output words of the runs leading to the loop. It remains to show that the runs are $M_T.n^{k+1}$-close. If this is the case, then we are done and have proven that case *b*) is satisfied. Otherwise, we can prove that the input word has length at least n^{k+1} and thus that there exists a synchronized loop (on the k runs) whose output words on components j and j' have distinct length. But then we are back in case *a*). □