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 operation 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); sum is the set union and product 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:

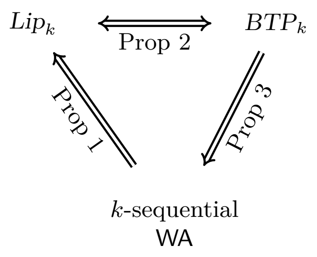
Theorem 2. *Let W be a weighted automaton with set semantics over an infinitary finitely generated group and k be a positive integer. The following assertions are equivalent:*

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, the 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 $\otimes$ 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 $\mathit{dist}(u, v) = |u| + |v| - 2 * |\mathit{lcp}(u, v)|$ where $\mathit{lcp}(u, v)$ is the longest common prefix of u and v.

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

Lemma 1. *Given a group $\mathbb{G}$, for all $\alpha, \alpha', \beta, \beta', \gamma, \gamma' \in \mathbb{G}$,*
1. *$\mathit{delay}(\alpha, \beta) = \mathbf{1}$ if and only if $\alpha = \beta$,*
2. *if $\mathit{delay}(\alpha, \alpha') = \mathit{delay}(\beta, \beta')$ then $\mathit{delay}(\alpha\gamma, \alpha'\gamma') = \mathit{delay}(\beta\gamma, \beta'\gamma')$.*

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 $\alpha, \beta \in \mathbb{G}$, $d(\alpha, \beta) = |\mathit{delay}(\alpha, \beta)|$.*

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 $(Q, t_{\mathit{init}}, t_{\mathit{final}}, T)$ where Q is a finite set of states, $t_{\mathit{init}} \subseteq Q \times \mathbb{M}$ (resp. $t_{\mathit{final}} \subseteq Q \times \mathbb{M}$) 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 $(q, \alpha) \in t_{\mathit{init}}$ (*resp.* $(q, \alpha) \in t_{\mathit{final}}$), 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 each of its states appears in some accepting run. W.l.o.g., we assume that the automata we consider are trimmed.

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

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 $(Q, t_{\mathit{init}}, t_{\mathit{final}}, T)$ over $\mathbb{M}$ is said to be sequential if $|t_{\mathit{init}}| = 1$ 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 allow finite sets of weights to be associated with final states, and not only singletons. This seems more appropriate to us regarding the parallel evaluation model for multi-sequential weighted automata: we prefer to merge threads that only differ by their final outputs. If we define $OutMax = \max_{q \in Q} |\{(q, \alpha) \in t_{\mathit{final}}\}|$, then the standard definition of sequential machines requires $OutMax = 1$. Being k-sequential implies being $(k \cdot OutMax)$-valued. Hence, multi-sequential weighted automata are included in finite-valued ones. However, multi-sequential weighted automata are strictly less expressive than finite-valued ones.

Allowing a final output relation obviously has an impact on the sequentiality degree. We believe that it is possible to fit the usual setting by appropriately reformulating our characterizations. However, this cannot be directly deduced from our current 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(a) 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 $f_{last}(u_1) + \cdots + f_{last}(u_n)$ is not multi-sequential (see for instance [12]) whereas it is single-valued.

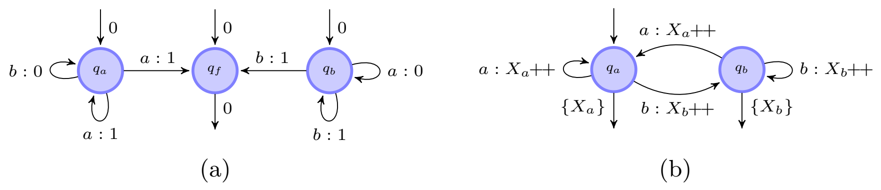


Fig. 1. (a) Example of a weighted automaton W_0 computing the function f_{last}. (b) Example of a cost register automaton C_0 computing the function f_{last}. The updates are abbreviated: $X_a{+}{+}$ means both $X_a := X_a + 1$ and $X_b := X_b$ (and conversely).

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 $\mathit{dist}(,)$, 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 $(w_0, \alpha_0), \ldots, (w_k, \alpha_k) \in R$, there are two indices i, j such that $0 \leqslant i < j \leqslant k$ and $d(\alpha_i, \alpha_j) \leqslant L\,(\mathit{dist}(w_i, w_j) + 1)$.*

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. This property is 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 $(q_{0,j}, \gamma_j) \in t_{\mathit{init}}$ 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

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

Example 3. The weighted automaton W_0, given in Figure 1(a), 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}.

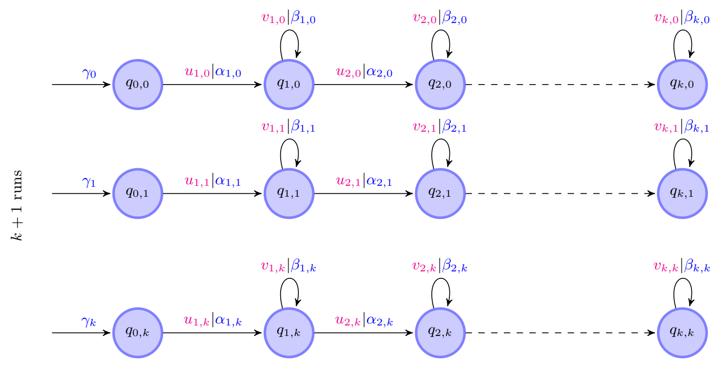


Fig. 2. Branching twinning property of order k

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 $W = (Q, t_{init}, t_{\mathit{final}}, T)$ 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 $(q', \mathit{delay}(\alpha_0\alpha, \beta_0\beta))$. 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 $W_S = (Q, S, t_{\mathit{final}}, T)$ 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 P_2 follows, since $k' + k'' \leqslant k$.

The properties P_1 and 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 $|\alpha| \leqslant N_W$, P_1 implies that U is finite. Moreover, as a consequence of 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 $(q_1, \nu_1)(q_2, \nu_2) \ldots (q_{k+1}, \nu_{k+1})$ 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 $(q_1, \nu_1)(q_2, \nu_2) \ldots (q_{k+1}, \nu_{k+1})$ 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 1(b) 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).

The class of CRA with k non-independent registers was characterized in [9] using the twinning property of order k. This property is weaker than our branching twinning property of order k as it requires the same conclusion but only for runs labeled by the same input words.

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 $W = (Q, t_{init}, t_{final}, T)$ 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 $delay(\alpha_j, \alpha_{j'}) \neq \mathbf{1}$. The procedure non-deterministically guesses the cycle in W^p (its length can be bounded by $2n^p$) and computes incrementally the value of $delay(\alpha_j, \alpha_{j'})$. $\square$

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 operations 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)
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)

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

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