# Pebble Weighted Automata and Weighted Logics

Benedikt Bollig, Paul Gastin, Benjamin Monmege, Marc Zeitoun

A

# Pebble Weighted Automata and Weighted Logics

BENEDIKT BOLLIG, PAUL GASTIN, BENJAMIN MONMEGE, LSV, ENS Cachan, CNRS, Inria
MARC ZEITOUN, Université de Bordeaux, LaBRI, CNRS, Inria

We introduce new classes of weighted automata on words. Equipped with pebbles, they go beyond the class of recognizable formal power series: they capture weighted first-order logic enriched with a quantitative version of transitive closure. In contrast to previous work, this calculus allows for unrestricted use of existential and universal quantifications over positions of the input word. We actually consider both two-way and one-way pebble weighted automata. The latter class constrains the head of the automaton to walk left-to-right, resetting it each time a pebble is dropped. Such automata have already been considered in the Boolean setting, in the context of data words. Our main result states that two-way pebble weighted automata, one-way pebble weighted automata, and our weighted logic are expressively equivalent. We also give new logical characterizations of standard recognizable series.

Categories and Subject Descriptors: F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs; F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages

General Terms: Algorithms, Languages, Theory, Verification

Additional Key Words and Phrases: Formal power series, Pebble automata, Quantitative properties, Weighted automata, Weighted logics

## 1. INTRODUCTION

Connections between logical and state-based formalisms have always been a fascinating research area in theoretical computer science, which produced some fundamental theorems. The line of classical results started with the equivalence of MSO logic and finite automata [Büchi 1959; Elgot 1961; Trakhtenbrot 1961] and gave rise to natural generalizations to trees in terms of logics for tree automata [Thatcher and Wright 1968; Rabin 1969; Comon-Lundh et al. 2008], tree-walking automata, and pebble tree automata [Bojańczyk et al. 2006; Samuelides and Segoufin 2007; Bojańczyk 2008]. Such automata models recently attracted significant interest, for instance in the context of manipulating XML documents and evaluating XPath queries [ten Cate and Segoufin 2010].

Other extensions of finite automata are of quantitative nature and include timed automata, probabilistic systems, and transducers, which all come with more or less natural, specialized logical characterizations. A generic concept of adding weights to qualitative systems is provided by the theory of weighted automata [Droste et al. 2009], first introduced by Schützenberger [Schützenberger 1961]. The output of a weighted automaton running on a

---

This is an extended and enhanced version of results published in [Bollig et al. 2010]. This paper was written when the last author was visiting LSV as a full time Inria researcher.
This work was partly supported by ANR 2010 BLAN 0202 01 FREC, ARCUS Île-de-France–Inde, and LIA InForMel.
Authors' addresses: B. Bollig, P. Gastin, and B. Monmege: LSV, ENS Cachan, CNRS & Inria, 61, Av. du Président Wilson, 94235 Cachan Cedex, France. Email: {bollig,gastin,monmege}@lsv.ens-cachan.fr; M. Zeitoun: LaBRI, Université de Bordeaux, CNRS & Inria, 351 cours de la Libération, 33405 Talence Cedex, France. Email: mz@labri.fr.

DOI:http://dx.doi.org/10.1145/0000000.0000000

word is no longer a Boolean value discriminating between accepted and rejected behaviors. A word is rather mapped to a weight from a semiring, summing over the weights of all possible runs, each calculated as the product of its transition outcomes. Indeed, probabilistic automata and word transducers appear as instances of that framework, which found its way into numerous application areas such as natural language processing and speech recognition or digital image compression (see [Droste et al. 2009, Part IV]).

A logical characterization of weighted automata, however, was established only recently [Droste and Gastin 2009], in terms of a (restricted) weighted MSO logic capturing the recognizable formal power series (i.e., the behaviors of finite weighted automata). The key idea is to interpret existential and universal quantifications as the operations sum and product from a semiring. The original work on a logical characterization of the recognizable series on words has been extended to more general structures such as ranked and unranked trees [Droste and Vogler 2006; Droste and Vogler 2009], nested words [Mathissen 2010], pictures [Fichtner 2011], and Mazurkiewicz traces [Meinecke 2006], to mention only a few. In all these cases, however, one has to restrict the universal first-order quantification to stay within the class of recognizable series.

In the present paper, we follow a different approach. Instead of restricting the logic, we define an extended automaton model that naturally *reflects* it. We will indeed show that universal quantification as interpreted in [Droste and Gastin 2009] is essentially captured by a pebble (two-way) mechanism in the automata-theoretic counterpart. Inspired by the theory of two-way and pebble automata on words and trees [Rabin and Scott 1959; Engelfriet and Hoogeboom 1999; Bojańczyk et al. 2006; Samuelides and Segoufin 2007; Bojańczyk 2008], we actually define weighted generalizations that preserve their natural connections with logic. More precisely, we introduce pebble weighted automata on words and establish expressive equivalence to weighted first-order logic with a quantitative extension of the Boolean transitive closure, extending the classical Boolean case for words [Engelfriet and Hoogeboom 2007]. Our equivalence proof makes a detour via one-way pebble weighted automata, which were already considered in the Boolean context for infinite alphabets [Neven et al. 2004] and restrict pebble weighted automata to left-to-right runs. Note that fragments of our logic also yield alternative characterizations of the (classical) recognizable formal power series.

**Related work.** There has been a wide range of quantitative extensions of first-order and temporal logic that are quite different from the logics studied in this paper. Most of them are specialized and cannot be instantiated by arbitrary semirings.

The formalisms from [Immerman and Lander 1990; Etessami 1997; Libkin 2000; Hella et al. 2001] consider counting and aggregate quantifiers for first-order logic to talk about two-sorted structures. Roughly speaking, they allow us to specify how often a Boolean formula shall be satisfied, or to sum up over all elements satisfying a property. However, a sentence still defines a language (of two-sorted structures) rather than a formal power series. Moreover, the focus in these works has been on (non-)definability and locality properties in the sense of Hanf and Gaifman. Connections with automata (not to mention weighted automata) were not studied, and there seems to be no immediate correspondence.

When weighted monadic second-order logic is provided with a threshold operator, it embeds probabilistic CTL [Hansson and Jonsson 1994], as observed in [Bollig and Gastin 2009]. In principle, this also applies to counting versions of the temporal logics CTL and LTL [Laroussinie et al. 2012; Laroussinie et al. 2010], which can reason about how often a formula is satisfied along a path. Again, a formula from these logics defines a language of words or trees rather than a series, which leads to classical (Boolean) satisfiability and model checking questions. Contrary to this, we are aiming in this paper at a Büchi-like correspondence of logic and automata for *formal power series*, which associate a *quantity* with every word.

A *quantitative* semantics of structures is given in [Fischer et al. 2010] for an extension of the $\mu$-calculus, though not in terms of a formal power series over a semiring. Disjunction and conjunction correspond to maximum and minimum, respectively, and temporal as well as fixed-point operators are defined using supremum, infimum, and discount factors. As opposed to our setting, the structures themselves are branching and quantitative (like in probabilistic CTL).

**Outline.** In Section 2, we recall the classical concept of recognizable formal power series and recall basics about logics. Weighted logics are introduced in Section 3. In Section 4, the class of recognizable series is revisited in the light of these weighted logics, which yields new logical characterizations. In Section 5, we go beyond recognizable series by introducing pebble weighted automata (two-way and one-way), and we show that both formalisms are expressively equivalent to weighted first-order logic augmented with transitive closure. Finally, Section 6 studies some algorithmic questions such as evaluation (the weighted version of the membership problem), and satisfiability. We conclude in Section 7.

## 2. NOTATION AND BACKGROUND

In this section we set up the notation and we recall some basic results on weighted automata and logics. We refer the reader to [Droste and Gastin 2009; Droste et al. 2009] for details.

Throughout the paper, $A$ denotes a finite alphabet and $A^*$ (respectively, $A^+$) is the free monoid (respectively, semigroup) over $A$, i.e., the set of words (respectively, nonempty words). The length of $u \in A^*$ is denoted by $|u|$. If $|u| = n \geqslant 1$, we usually write $u = u_1 \cdots u_n$ with $u_i \in A$ and we let $\mathrm{pos}(u) = \{1, \ldots, n\}$. For $1 \leqslant i \leqslant j \leqslant n$, we denote by $u[i..j]$ the factor $u_i u_{i+1} \cdots u_j$ of $u$. Finally, we let $A^{\leqslant k} = \bigcup_{0 \leqslant i \leqslant k} A^i$.

### 2.1. Formal power series

A *semiring* is a structure $(\mathbb{S}, +, \times, 0, 1)$ where $(\mathbb{S}, +, 0)$ is a commutative monoid, $(\mathbb{S}, \times, 1)$ is a monoid, $\times$ distributes over $+$, and 0 is an absorbing element for $\times$. We say that $\mathbb{S}$ is *commutative* if so is $(\mathbb{S}, \times, 1)$.

*Example* 2.1. We shall refer in the examples to the usual Boolean semiring $(\{0,1\}, \vee, \wedge, 0, 1)$, denoted by $\mathbb{B}$, the semiring $(\mathbb{N}, +, \times, 0, 1)$ of natural numbers, the semiring $(\mathbb{Z}, +, \times, 0, 1)$ of integers, the tropical semiring $(\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$, denoted by $\mathbb{T}$, the arctic semiring $(\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$, denoted by $\mathbb{A}$, and the probabilistic semiring $(\mathbb{R}^+, +, \times, 0, 1)$, denoted by $\mathbb{P}$.

A *formal power series* (or *series*, for short) is a mapping $f : A^+ \to \mathbb{S}$. The set of formal power series is denoted by $\mathbb{S}\langle\langle A^+ \rangle\rangle$. We denote again by $+$ and $\times$ the pointwise addition and multiplication (called the *Hadamard product*) on $\mathbb{S}\langle\langle A^+ \rangle\rangle$, and by 0 and 1 the constant series with values 0 and 1, respectively.

### 2.2. Weighted automata

All automata we consider are finite. A *(one-way) weighted automaton* (1WA) over $(\mathbb{S}, +, \times, 0, 1)$ is a tuple $\mathcal{A} = (Q, A, I, F, \Delta, \mathsf{weight})$, where $Q$ is the set of states, $A$ is a finite alphabet, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, $\Delta \subseteq Q \times A \times Q$ is the set of transitions, and $\mathsf{weight} : \Delta \to \mathbb{S}$ assigns a weight to every transition. A *run* on a word $u = u_1 \cdots u_n$ is a sequence of transitions $\rho = q_0 \xrightarrow{u_1} q_1 \xrightarrow{u_2} \cdots \xrightarrow{u_n} q_n$. The *weight* of the run $\rho$ is

$$\mathsf{weight}(\rho) \stackrel{\text{def}}{=} \prod_{i=1}^{n} \mathsf{weight}(q_{i-1}, u_i, q_i)\,.$$

Such a run $\rho$ is said to be *accepting* if $q_0 \in I$ and $q_n \in F$. The *weight* $[\![\mathcal{A}]\!](u)$ of $u$ is the sum of the weights of all accepting runs on $u$.

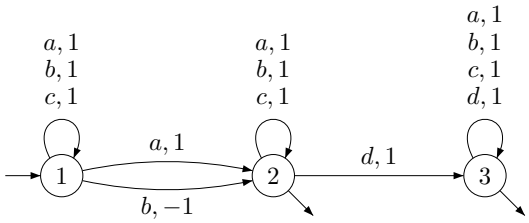


Fig. 1. Automaton for Example 2.3

One may also describe a weighted automaton by matrices: for all $a \in A$, the transition matrix $\mu(a) \in \mathbb{S}^{Q\times Q}$ is defined by $\mu(a)_{q,q'} = \mathsf{weight}(q, a, q')$ for all $q, q' \in Q$ if $(q, a, q') \in \Delta$, and 0 otherwise. Moreover, one defines matrices $\lambda \in \mathbb{S}^{1\times Q}$ and $\gamma \in \mathbb{S}^{Q\times 1}$ as characteristic matrices of the sets $I$ and $F$ respectively, i.e., for every state $q \in Q$, $\lambda_q = 1$ if $q \in I$, 0 otherwise, and $\gamma_q = 1$ if $q \in F$, 0 otherwise. The mapping $\mu : A \to \mathbb{S}^{Q\times Q}$ extends uniquely to a morphism $\mu : A^+ \to \mathbb{S}^{Q\times Q}$ (where $\mathbb{S}^{Q\times Q}$ is endowed with the usual multiplication of matrices). One can then verify that $[\![\mathcal{A}]\!](u) = \lambda \times \mu(u) \times \gamma$ (see, e.g., [Berstel and Reutenauer 2010, Prop. 6.1] for more details).

We call $[\![\mathcal{A}]\!] \in \mathbb{S}\langle\langle A^+\rangle\rangle$ the *behavior*, or *semantics* of $\mathcal{A}$. A formal power series $f \in \mathbb{S}\langle\langle A^+\rangle\rangle$ is called *recognizable* if it is the behavior of some weighted automaton. We let $\mathbb{S}^{\mathrm{rec}}\langle\langle A^+\rangle\rangle$ be the collection of all recognizable formal power series.

*Example* 2.2. In the semiring $\mathbb{N}$, let $\mathcal{A}$ be the automaton with a single state $q$, both initial and final, and transitions $(q, a, q)$ of weight 2 for all $a \in A$. Then, $[\![\mathcal{A}]\!](u) = 2^{|u|}$ for all $u \in A^+$.

*Example* 2.3. Let $A = \{a, b, c, d\}$. Suppose we want to compute, for a word $u \in A^+$, the difference between the number of $a$'s and the number of $b$'s before the first occurrence of $d$. Over the semiring $\mathbb{Z}$, the automaton $\mathcal{A}$ represented in Figure 1 computes this difference relying on the non-deterministic move from state 1 to state 2.

It is well-known [Berstel and Reutenauer 2010] that $\mathbb{S}^{\mathrm{rec}}\langle\langle A^+\rangle\rangle$ is stable under addition and, if $\mathbb{S}$ is commutative, also under Hadamard product.

### 2.3. Monadic second-order logic

Let us fix infinite supplies of first-order variables $\mathrm{Var} = \{x, y, z, t, x_1, x_2, \ldots\}$, and of second-order variables $\mathrm{VAR} = \{X, Y, X_1, X_2, \ldots\}$. The set $\mathrm{MSO}(A)$ (or simply MSO) of monadic second-order formulae over $A$ is given by the grammar:

$$\varphi ::= P_a(x) \mid x \leqslant y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\, \varphi \mid \exists X\, \varphi$$

where $a \in A$, $x, y \in \mathrm{Var}$ and $X \in \mathrm{VAR}$. As usual, the set $\mathrm{FO}(A)$ (or simply FO) of first-order formulae over $A$ is the fragment of $\mathrm{MSO}(A)$ without second-order quantifications $\exists X$.

For $\varphi \in \mathrm{MSO}(A)$, let $\mathrm{Free}(\varphi)$ denote the set of free variables of $\varphi$. If $\mathrm{Free}(\varphi) = \emptyset$, then $\varphi$ is called a *sentence*. For a finite set $\mathcal{V} \subseteq \mathrm{Var} \cup \mathrm{VAR}$ and a word $u \in A^+$, a $(\mathcal{V}, u)$-*assignment* is a function $\sigma$ that maps first-order variables in $\mathcal{V}$ to elements of $\mathrm{pos}(u)$ and second-order variables in $\mathcal{V}$ to subsets of $\mathrm{pos}(u)$. For $x \in \mathrm{Var}$ and $i \in \mathrm{pos}(u)$, $\sigma[x \mapsto i]$ denotes the $(\mathcal{V} \cup \{x\}, u)$-assignment that maps $x$ to $i$ and, otherwise, coincides with $\sigma$. For $X \in \mathrm{VAR}$ and $I \subseteq \mathrm{pos}(u)$, the $(\mathcal{V} \cup \{X\}, u)$-assignment $\sigma[X \mapsto I]$ is defined similarly.

A pair $(u, \sigma)$, where $\sigma$ is a $(\mathcal{V}, u)$-assignment, can be encoded as a word over the extended alphabet $A_{\mathcal{V}} \stackrel{\text{def}}{=} A \times \{0, 1\}^{\mathcal{V}}$. We write a word $(u_1, \sigma_1) \cdots (u_n, \sigma_n) \in A_{\mathcal{V}}^+$ as $(u, \sigma)$ where $u = u_1 \cdots u_n$ and $\sigma = \sigma_1 \cdots \sigma_n$. We call $(u, \sigma)$ *valid* if, for each first-order variable $x \in \mathcal{V}$, the $x$-row of $\sigma$ contains exactly one 1. If $(u, \sigma)$ is valid, then $\sigma$ encodes the $(\mathcal{V}, u)$-assignment

that maps a first-order variable $x \in \mathcal{V}$ to the unique position carrying 1 in the $x$-row, and a second-order variable $X \in \mathcal{V}$ to the set of positions carrying 1 in the $X$-row.

We fix a finite set $\mathcal{V}$ of variables such that $\mathrm{Free}(\varphi) \subseteq \mathcal{V}$. For every $u \in A^+$ and every $(\mathcal{V}, u)$-assignment $\sigma$, we define $u, \sigma \models \varphi$ by induction over the formula $\varphi$, as shown in Table I. Note in particular that the semantics of $\varphi$ only depends on the restriction of $\sigma$ to free variables of $\varphi$ in $\mathcal{V}$.

Table I. Semantics of MSO

| | |
|---|---|
| $u, \sigma \models P_a(x)$ | if $u_{\sigma(x)} = a$ |
| $u, \sigma \models x \leqslant y$ | if $\sigma(x) \leqslant \sigma(y)$ |
| $u, \sigma \models x \in X$ | if $\sigma(x) \in \sigma(X)$ |
| $u, \sigma \models \neg\varphi$ | if $u, \sigma \not\models \varphi$ |
| $u, \sigma \models \varphi_1 \vee \varphi_2$ | if $u, \sigma \models \varphi_1$ or $u, \sigma \models \varphi_2$ |
| $u, \sigma \models \exists x\, \varphi$ | if there exists $i \in \mathrm{pos}(u)$ such that $u, \sigma[x \mapsto i] \models \varphi$ |
| $u, \sigma \models \exists X\, \varphi$ | if there exists $I \subseteq \mathrm{pos}(u)$ such that $u, \sigma[X \mapsto I] \models \varphi$ |

We use abbreviations for constants, modulo constraints and comparisons, like for example, $x \leqslant y + 2$. We use first and last as abbreviations for the first and last positions of a word. All of these shortcuts can be replaced by suitable FO-formulae, except modulo constraints such as "$x \equiv_\ell m$" with $1 \leqslant m \leqslant \ell$, which are MSO-definable.

In the sequel, we let FO+mod be the fragment of MSO consisting of FO augmented with modulo constraints $x \equiv_\ell m$ for constants $1 \leqslant m \leqslant \ell$ (since the positions of words start with 1, it is more convenient to compute modulo as a value between 1 and $\ell$). The semantics is given by

$$[\![x \equiv_\ell m]\!](u, \sigma) = \begin{cases} 1 & \text{if } \sigma(x) \equiv m \bmod \ell \\ 0 & \text{otherwise} \end{cases}$$

and it is MSO-definable by

$$x \equiv_\ell m \stackrel{\text{def}}{=} \forall X \Big( \big[ (x \in X) \wedge \big( \forall y (y \in X \wedge y > \ell) \implies y - \ell \in X \big) \big] \implies m \in X \Big).$$

## 3. WEIGHTED LOGICS

Our aim is to define a denotational model to describe the behavior of weighted automata. This has already been done by Droste and Gastin [Droste and Gastin 2009]: they have introduced weighted logics with syntax close to monadic second-order logic, extending the semantics by using addition and product of a semiring to evaluate disjunctions/existential quantifications, and conjunctions/universal quantifications, respectively. At the price of strong restrictions on the shape of the formulae, they succeeded to prove an expressiveness result (in our formalism, this result is stated in Theorem 3.8).

The syntax of [Droste and Gastin 2009] is purely quantitative, though Boolean connectives can be expressed indirectly. As it may be somewhat confusing to interpret purely logical formulae in a weighted manner, we slightly modify the original syntax, by clearly separating the logical and the quantitative parts: our weighted logic consists of a Boolean kernel, such as MSO or FO, augmented with quantitative operators (addition, multiplication, sum and product quantifications, and possibly weighted transitive closure). Thus, our language will allow us to test explicitly for Boolean monadic second-order properties, and to perform computations. We shall see that both formalisms are equivalent. However, in addition to being more intuitive, the new syntax allows flexibility to study expressiveness. For instance, one can investigate how the underlying Boolean logic influences the computational power (see Lemma 3.3).

### 3.1. General definitions

*Definition* 3.1 (*Weighted logics*). Given a class $\mathcal{L}$ of Boolean formulae, we denote by wMSO($\mathcal{L}$) the class of *weighted monadic second-order logic* defined by

$$\Phi ::= s \mid \varphi \mid \Phi \oplus \Phi \mid \Phi \otimes \Phi \mid \bigoplus_x \Phi \mid \bigotimes_x \Phi \mid \bigoplus_X \Phi \mid \bigotimes_X \Phi$$

where $s \in \mathbb{S}$, $\varphi \in \mathcal{L}$, $x \in \mathrm{Var}$ and $X \in \mathrm{VAR}$. Disabling the sum and product indexed by set variables, we define the class wFO($\mathcal{L}$) of *weighted first-order logic* by

$$\Phi ::= s \mid \varphi \mid \Phi \oplus \Phi \mid \Phi \otimes \Phi \mid \bigoplus_x \Phi \mid \bigotimes_x \Phi$$

where $s \in \mathbb{S}$, $\varphi \in \mathcal{L}$ and $x \in \mathrm{Var}$.

We denote again by $\mathrm{Free}(\Phi)$ the set of free variables of a formula $\Phi \in \mathrm{wMSO}(\mathcal{L})$. Let again $\mathcal{V}$ be a finite set of variables such that $\mathrm{Free}(\Phi) \subseteq \mathcal{V}$. The semantics $[\![\Phi]\!]_{\mathcal{V}} \in \mathbb{S}\langle\langle A_{\mathcal{V}}^+ \rangle\rangle$ of $\Phi$ wrt. $\mathcal{V}$ is defined as follows: if $(u, \sigma)$ is not valid, we set $[\![\Phi]\!]_{\mathcal{V}}(u, \sigma) = 0$. Otherwise $[\![\Phi]\!]_{\mathcal{V}}$ is given in Table II. Hereby, if $\mathbb{S}$ is not commutative, the products follow the natural order on $\mathrm{pos}(u)$ and the lexicographic order on the power set $\{0,1\}^{\mathrm{pos}(u)}$.

Table II. Semantics of wMSO($\mathcal{L}$)

$$[\![s]\!](u,\sigma) = s \qquad [\![\varphi]\!](u,\sigma) = \begin{cases} 1 & \text{if } u, \sigma \models \varphi \\ 0 & \text{otherwise} \end{cases}$$

$$[\![\Phi_1 \oplus \Phi_2]\!](u,\sigma) = [\![\Phi_1]\!](u,\sigma) + [\![\Phi_2]\!](u,\sigma) \qquad [\![\Phi_1 \otimes \Phi_2]\!](u,\sigma) = [\![\Phi_1]\!](u,\sigma) \times [\![\Phi_2]\!](u,\sigma)$$

$$[\![\textstyle\bigoplus_x \Phi]\!](u,\sigma) = \sum_{i \in \mathrm{pos}(u)} [\![\Phi]\!](u, \sigma[x \mapsto i]) \qquad [\![\textstyle\bigotimes_x \Phi]\!](u,\sigma) = \prod_{i \in \mathrm{pos}(u)} [\![\Phi]\!](u, \sigma[x \mapsto i])$$

$$[\![\textstyle\bigoplus_X \Phi]\!](u,\sigma) = \sum_{I \subseteq \mathrm{pos}(u)} [\![\Phi]\!](u, \sigma[X \mapsto I]) \qquad [\![\textstyle\bigotimes_X \Phi]\!](u,\sigma) = \prod_{I \subseteq \mathrm{pos}(u)} [\![\Phi]\!](u, \sigma[X \mapsto I])$$

*Example* 3.2. In the Boolean semiring $\mathbb{B}$, recognizable and wMSO(MSO)-definable series coincide. In contrast, for the natural semiring $\mathbb{N}$, the definition yields $[\![\bigotimes_x \bigotimes_y 2]\!](u) = 2^{|u|^2}$, which is not recognizable [Droste and Gastin 2009]. Indeed, let $\mathcal{A} = (Q, A, I, F, \Delta, \mathsf{weight})$. Then, $[\![\mathcal{A}]\!](u) = O((M|Q|)^{|u|+1})$ for $M = \max(\mathsf{weight}(\Delta))$, since there are $O(|Q|^{|u|+1})$ runs on $u$, each of weight $O(M^{|u|})$. Also observe that the behavior of the automaton of Example 2.2 is $[\![\bigotimes_y 2]\!]$. Therefore, recognizable series are not stable under first-order product $\bigotimes_x$.

We denote by AP the class of atomic propositions, i.e., that contains only formulae $P_a(x)$, $x \leqslant y$, $x \in X$ and their negations. As explained at the beginning of the section, the logic wMSO($\mathbb{S}$) (respectively, wFO($\mathbb{S}$)) introduced in [Droste and Gastin 2009] is exactly the class of formulae wMSO(AP) (respectively, wFO(AP)). Transposed into our syntax, the transformation described in their Definition 4.3 and proved in Lemma 4.4, permits to state:

Lemma 3.3. *For all formulae $\varphi \in$ MSO (respectively, $\varphi \in$ FO), there exists a formula $\Phi \in$ wMSO(AP) (respectively, $\Phi \in$ wFO(AP)), with the same set of free variables, such that $[\![\varphi]\!] = [\![\Phi]\!]$.*

Hence, we can always replace a Boolean FO or MSO formula by its wFO(AP) or wMSO(AP) equivalent and we obtain:

Corollary 3.4. *Let $\mathbb{S}$ be a (possibly non-commutative) semiring and $f \in \mathbb{S}\langle\langle A^+ \rangle\rangle$. Then,*

— *$f$ is* wMSO(MSO)*-definable if and only if $f$ is* wMSO(AP)*-definable.*

— *$f$ is* wFO(FO)*-definable if and only if $f$ is* wFO(AP)*-definable.*

For a Boolean formula $\varphi \in \mathcal{L}$ and a formula $\Phi \in \mathrm{wMSO}(\mathcal{L})$, we define the macro

$$\varphi \ominus \Phi \stackrel{\text{def}}{=} \neg\varphi \oplus (\varphi \otimes \Phi)$$

as a natural generalization of the Boolean implication ($\varphi \to \varphi'$ for $\varphi, \varphi' \in \mathcal{L}$): its semantics is $[\![\Phi]\!]$ if $\varphi$ holds, and 1 otherwise.

*Example* 3.5. The series recognized by the weighted automaton in Example 2.3 can also be described by the following formula in wFO(FO):

$$\bigoplus_y \left[\left(\neg\exists x\ (x \leqslant y \wedge P_d(x))\right) \otimes \left(P_a(y) \oplus (P_b(y) \otimes (-1))\right)\right].$$

*Example* 3.6. Consider now a machine with one counter that can take non-negative values, being either incremented, decremented or zero-tested on each step. We encode the behavior of such a machine by recording the string of instructions applied on the counter, letters *Inc*, *Dec* and *ZTest* stand for increment, decrement and zero-test respectively. We will show in Theorem 6.2 that satisfiability of a formula in the logic wFO(FO) over the semiring $\mathbb{Z}$ is undecidable, by reducing the problem of emptiness of a two-counter machine. In order to prepare this theorem, we design formulae that map a word $u \in \{Inc, Dec, ZTest\}^+$ to weight 0 when $u$ encodes a non-valid behavior of a non-negative counter.

First, we want to make sure that the counter always stays non-negative. For this, considering a word $u \in \{Inc, Dec, ZTest\}^+$, and assuming that the counter starts with value 0, the value[1]

$$\prod_{i \in \mathrm{pos}(u) | u_i = Dec} \left(|u[1..i-1]|_{Inc} - |u[1..i-1]|_{Dec}\right)$$

is 0 if and only if the counter becomes negative along the sequence of instructions encoded by $u$, since at the first position $i$ where it becomes negative, we have $|u[1..i-1]|_{Inc} = |u[1..i-1]|_{Dec}$. By using a simpler version of the formula described in Example 3.5, we can compute this value with the following formula of wFO(AP):

$$\bigotimes_x \left[P_{Dec}(x) \ominus \bigoplus_y \left[y < x \otimes \left(P_{Inc}(y) \oplus (P_{Dec}(y) \otimes (-1))\right)\right]\right].$$

Next, assuming that the counter always stays non-negative, we want to check that a zero-test is used only when the counter is indeed 0. For a word $u \in \{Inc, Dec, ZTest\}^+$, let us consider the value

$$\prod_{i \in \mathrm{pos}(u) | u_i = ZTest} \ \prod_{1 \leqslant k < i} \left(|u[1..i-1]|_{Inc} - |u[1..i-1]|_{Dec} - k\right).$$

Note that, assuming that the counter is 0 initially and always stays non-negative, this value is 0 if and only if some zero-test occurs when the counter is not zero. Note that this series can be defined by the following formula in wFO(AP):

$$\bigotimes_x \left[P_{ZTest}(x) \ominus \bigotimes_z \left[z < x \ominus \bigoplus_y \left[(y \leqslant z \otimes (-1)) \oplus \left(y < x \otimes \left(P_{Inc}(y) \oplus (P_{Dec}(y) \otimes (-1))\right)\right)\right]\right]\right]$$

### 3.2. Previous expressiveness result

For $\mathcal{L} \subseteq \mathrm{MSO}$ closed under $\vee$, $\wedge$ and $\neg$, an *$\mathcal{L}$-step formula* is a formula obtained from the grammar

$$\Phi ::= s \mid \varphi \mid \Phi \oplus \Phi \mid \Phi \otimes \Phi, \qquad \text{with } s \in \mathbb{S} \text{ and } \varphi \in \mathcal{L}. \tag{1}$$

The following lemma shows in particular that an $\mathcal{L}$-step formula assumes a finite number of values, each of which corresponds to an $\mathcal{L}$-definable language.

[1] Here, and in the following, $|u|_a$ stands for the number of occurrences of letter $a$ in word $u$.

LEMMA 3.7. *For every $\mathcal{L}$-step formula $\Phi$, one can construct an equivalent formula $\Psi = \bigoplus_{i\in I}(\psi_i \otimes s_i)$ with $I$ finite, $\psi_i \in \mathcal{L}$ and $s_i \in \mathbb{S}$. More precisely,* $\mathrm{Free}(\Phi) = \mathrm{Free}(\Psi)$ *and for every $\mathcal{V}$ containing* $\mathrm{Free}(\Phi)$, $[\![\Phi]\!]_{\mathcal{V}}(u,\sigma) = [\![\Psi]\!]_{\mathcal{V}}(u,\sigma)$ *for all* $(u,\sigma) \in A^+_{\mathcal{V}}$.

PROOF. Call $\varphi_1, \dots, \varphi_p$ the $\mathcal{L}$-formulae occurring in the expression of $\Phi$ given by the grammar (1). For $I \subseteq \{1,\dots,p\}$, let $\Phi_I$ be the formula obtained by replacing in $\Phi$ each $\varphi_i$ by 1 if $i \in I$ and by 0 otherwise, so that $\Phi_I$ has no free variable and $[\![\Phi_I]\!]$ is a constant $s_I \in \mathbb{S}$. Let $\psi_I = \bigwedge_{i\in I}\varphi_i \wedge \bigwedge_{i\notin I}\neg\varphi_i$, which is an $\mathcal{L}$-formula, since $\mathcal{L}$ is closed under $\wedge$ and $\neg$. Let $\Psi = \bigoplus_I(\psi_I \otimes s_I)$. Clearly, $\mathrm{Free}(\Phi) = \mathrm{Free}(\Psi)$.

Let $\mathcal{V} \supseteq \mathrm{Free}(\Phi)$. Fix a word $u$ and a $(\mathcal{V},u)$-assignment $\sigma$. Let $J = \{i \mid (u,\sigma) \models \varphi_i\}$, so that $(u,\sigma) \models \psi_J$ and $(u,\sigma) \not\models \psi_I$ for $I \neq J$. Therefore, $[\![\Psi]\!]_{\mathcal{V}}(u,\sigma) = s_J = [\![\Phi_J]\!]_{\mathcal{V}}(u,\sigma) = [\![\Phi]\!]_{\mathcal{V}}(u,\sigma)$, where the last equality comes from the definition of $J$. □

From now on, we freely use Lemma 3.7, using the special form it provides for $\mathcal{L}$-step formulae. All MSO-step formulae are clearly recognizable. By [Droste and Gastin 2009], $\bigotimes_x \Phi$ is recognizable for any MSO-step formula $\Phi$. A restricted fragment sREMSO of wMSO(AP) (called the syntactically restricted formulae), was defined in [Droste and Gastin 2009]. Essentially, sREMSO consists of wMSO(MSO) formulae of the form $\bigoplus_{X_1}\cdots\bigoplus_{X_n}\Phi$ where $\Phi$ is a wFO(MSO) formula such that if it contains $\Psi \otimes \Psi'$ as a subformula then the values in $\mathbb{S}$ appearing in $\Psi$ and $\Psi'$ commute element-wise, and such that the use of first-order product $\bigotimes_x$ is restricted to MSO-step formulae. Note that if the semiring is commutative, the requirement over the product $\Psi \otimes \Psi'$ is trivially verified. A consequence of their work is the following statement.

THEOREM 3.8 ([DROSTE AND GASTIN 2009]). *A formal power series is recognizable if and only if it is definable in* sREMSO.

### 3.3. Weighted transitive closure

We now define other fragments of wMSO(MSO), which also carry enough expressiveness to generate all recognizable series. Contrary to sREMSO that allows second-order quantitative operators, and finally restricts them, we rather extend the fragment wFO(MSO), containing only first-order quantitative operators, with a weighted equivalent of the Boolean transitive closure.

We first define a very general weighted transitive closure operator before presenting some restrictions over it that will be useful in the sequel.

*Weighted transitive closure.* For a formula $\Phi(x,y)$ with at least two free variables $x$ and $y$, we introduce a weighted transitive closure operator $[\mathrm{TC}_{x,y}\Phi](x',y')$ having as free variables the fresh variables $x'$ and $y'$, and all the free variables of $\Phi$ except $x$ and $y$. Its semantics is defined by

$$[\![[\mathrm{TC}_{x,y}\Phi](x',y')]\!](u,\sigma) = \sum_{\sigma(x')=i_0,i_1,\dots,i_m=\sigma(y')} \prod_{0\leqslant k\leqslant m-1} [\![\Phi]\!](u,\sigma[x\mapsto i_k, y\mapsto i_{k+1}]) \quad (2)$$

where the sum runs over all $m > 0$ and all sequences $(i_k)_{0\leqslant k\leqslant m}$ of positions of the word $u$ with $i_0 = \sigma(x')$ and $i_m = \sigma(y')$ such that either $m = 1$, or $m > 1$ and all positions of the sequence are pairwise distinct. This sum is finite as the word has finitely many positions. Notice that if $\sigma(x') = \sigma(y') = i$, then the semantics is $[\![\Phi]\!](u,\sigma[x\mapsto i, y\mapsto i])$.

To ease notation, we now write $[\![\Phi(x,y)]\!](u,i,j)$ instead of $[\![\Phi(x,y)]\!](u,[x\mapsto i, y\mapsto j])$.

*Example* 3.9. Let $\Psi = \mathrm{TC}_{x,y}1$ over $\mathbb{N}$. Then for a word $u$ of length $n$, we have $[\![\Psi]\!](u,1,n) = \sum_{m=0}^{n-2} m!\binom{n-2}{m}$, since the sum in (2) ranges over all sequences of pairwise distinct positions in $\{1,\dots,n\}$ starting in 1 and ending in $n$.

*Ordered weighted transitive closure.* Since later we consider one-way weighted automata, it is natural to restrict this operator to a left-to-right version. Hence, we introduce the operator $\mathrm{TC}^{<}_{x,y}\Phi$ whose semantics consists in restricting the sum of (2) to *non-decreasing* sequences $(i_k)_{0\leqslant k\leqslant m}$ of positions in the word. Equivalently, we can define it by

$$\mathrm{TC}^{<}_{x,y}\Phi \stackrel{\text{def}}{=} \mathrm{TC}_{x,y}(x \leqslant y \otimes \Phi)\,.$$

Intuitively, the $\mathrm{TC}^{<}_{x,y}$ operator generalizes the forward transitive closure operator of the Boolean case: a formula $\varphi(x,y)$ with two free variables defines a binary relation on positions of a word $u$, namely $\{(i,j) \mid (i \leqslant j) \wedge u,i,j \models \varphi\}$. The relation defined by $\mathrm{TC}^{<}_{x,y}\varphi$ is the transitive closure of this relation.

Throughout this paper, we will mostly use ordered transitive closure. For this reason, we define recursively the $m$-th power of a formula $\Phi$, with this ordered restriction

$$\Phi^1(x,y) \stackrel{\text{def}}{=} (x \leqslant y) \otimes \Phi(x,y),$$
$$\Phi^{m+1}(x,y) \stackrel{\text{def}}{=} \bigoplus\nolimits_z \big[(x < z < y) \otimes \Phi(x,z) \otimes \Phi^m(z,y)\big], \quad \text{for } m \geqslant 1.$$

The left-to-right restriction of the transitive closure permits finally to give a third equivalent definition, *a priori* simpler to use:

$$[\![\mathrm{TC}^{<}_{x,y}\Phi]\!] = \sum_{m\geqslant 1} [\![\Phi^m(x,y)]\!]\,.$$

This infinite sum is well-defined: $[\![\Phi^m(x,y)]\!](u,\sigma) = 0$ if $m \geqslant |u|$, i.e., on each pair $(u,\sigma)$, only finitely many terms assume a nonzero value.

*Bounded weighted transitive closure.* We also introduce a *bounded* weighted transitive closure operation $\mathrm{TC}^{N}_{x,y}$ for each integer $N > 0$. Its semantics consists in restricting the sequences of (2) so that $|i_{k+1} - i_k| \leqslant N$ for every $k \in \{0,\dots,m-1\}$. Equivalently,

$$\mathrm{TC}^{N}_{x,y}\Phi \stackrel{\text{def}}{=} \mathrm{TC}_{x,y}(|y-x| \leqslant N \otimes \Phi)\,.$$

*Bounded ordered weighted transitive closure.* Finally, we can combine both restrictions and introduce a *bounded ordered* weighted transitive closure operation. For each integer $N > 0$, the $\mathrm{TC}^{N,<}_{x,y}$ operator is given by

$$\mathrm{TC}^{N,<}_{x,y}\Phi = \mathrm{TC}_{x,y}\big((x \leqslant y \leqslant x+N) \otimes \Phi\big)\,. \tag{3}$$

*Definition* 3.10. Given a class $\mathcal{L}$ of Boolean formulae, we denote by $\mathrm{wFOTC}(\mathcal{L})$ the class of weighted formulae defined by

$$\Phi ::= s \mid \varphi \mid \Phi \oplus \Phi \mid \Phi \otimes \Phi \mid \bigoplus\nolimits_x \Phi \mid \bigotimes\nolimits_x \Phi \mid \mathrm{TC}_{x,y}\Phi$$

with $s \in \mathbb{S}$, $\varphi \in \mathcal{L}$, $x,y \in \mathrm{Var}$. A series $f \in \mathbb{S}\langle\!\langle A^+ \rangle\!\rangle$ is said to be $\mathrm{wFOTC}(\mathcal{L})$-definable if there is a formula $\Phi \in \mathrm{wFOTC}(\mathcal{L})$ such that $[\![\Phi]\!] = f$. We define similar notions for the restricted operators $\mathrm{TC}^{<}$, $\mathrm{TC}^{N}$ and $\mathrm{TC}^{N,<}$, denoting them $\mathrm{wFOTC}^{<}(\mathcal{L})$, $\mathrm{wFOTC}^{\mathrm{b}}(\mathcal{L})$ and $\mathrm{wFOTC}^{\mathrm{b},<}(\mathcal{L})$, respectively: in particular, for the bounded restrictions, we allow all constants $N$ in the transitive closure operators.

Notice that if $\Phi(x)$ is a formula from $\mathrm{wFOTC}(\mathcal{L})$ with a free variable $x$, then, for a fixed integer $k$, the formula $\bigoplus_y [(y = x+k) \otimes \Phi(y)]$ is again a formula of the same fragment, that we will denote $\Phi(x+k)$ in the following.

*Example* 3.11. Let $\Phi(x,y) \stackrel{\text{def}}{=} \bigotimes_z 2$ over $\mathbb{N}$. Let $u = u_1 \cdots u_n$ with $n \geqslant 2$. We have $[\![\mathrm{TC}^{1,<}_{x,y}\Phi]\!](u,1,n) = \prod_{i=1}^{n-1} [\![\Phi]\!](u,i,i{+}1)$ due to the bounded and ordered weighted transitive closure. Now, $[\![\Phi]\!](u,i,i+1) = 2^{|u|}$ if $i \geqslant 1$, so $[\![\mathrm{TC}^{1,<}_{x,y}\Phi]\!](u,1,n) = 2^{|u|(|u|-1)}$. Noticing that the series $u \mapsto 2^{|u|}$ is recognizable by a weighted automaton, but not the series $u \mapsto 2^{|u|(|u|-1)}$

(similarly to Example 3.2), this example shows in particular that the class of recognizable series is not closed under application of bounded ordered weighted transitive closures.

*Example* 3.12. Consider the formula $\Phi = \mathrm{TC}^{2,<}_{x,y}(2 \otimes y = x + 2)$ over $A$ and $\mathbb{N}$. Then

$$[\![\Phi]\!](u, 1, |u|) = \begin{cases} 2^n & \text{if } |u| = 2n+1 \text{ with } n \geqslant 1 \\ 0 & \text{otherwise.} \end{cases}$$

Notice that the support of $[\![\Phi]\!]$ (i.e., the language of all words mapped to nonzero values) is not FO-definable. Therefore, $[\![\Phi]\!]$ is not wFO(FO)-definable, since otherwise, we would have $[\![\Phi]\!] = [\![\Psi]\!]$ for a wFO(FO)-formula $\Psi$, and the FO formula obtained from $\Psi$ by changing all nonzero constants of $\mathbb{N}$ to 1 and replacing $\oplus$, $\otimes$, $\bigoplus_x$ and $\bigotimes_x$ by $\vee$, $\wedge$, $\exists x$ and $\forall x$ respectively would define the support of $[\![\Phi]\!]$ (this holds because $\mathbb{N}$ is a positive semiring).

*Example* 3.13. The *modulo* predicate can easily be expressed in $\mathrm{wFOTC}^{\mathrm{b},<}(\mathrm{FO})$. For $1 \leqslant m \leqslant \ell$, the predicate is expressed by

$$x \equiv_\ell m \stackrel{\text{def}}{=} (x = m) \oplus \left[\mathrm{TC}^{\ell,<}_{x',y'}(y' = x' + \ell)\right](m, x)$$

while the binary relation can be expressed with

$$x \equiv_\ell y \stackrel{\text{def}}{=} (y = x) \oplus \left[\mathrm{TC}^{\ell,<}_{x',y'}(y' = x' + \ell)\right](x, y) \oplus \left[\mathrm{TC}^{\ell,<}_{x',y'}(y' = x' + \ell)\right](y, x)\,.$$

## 4. EXPRESSIVENESS OF WEIGHTED LOGICS

We now consider syntactical restrictions of $\mathrm{wFOTC}^{<}(\mathrm{FO})$ and $\mathrm{wFOTC}^{\mathrm{b},<}(\mathrm{FO})$, inspired by normal form formulae of [Neven and Schwentick 2003] where only one "external" weighted transitive closure operator is allowed.

In the following, sentences like "$f$ is definable by a formula $\mathrm{TC}_{x,y}\Psi$" means that for every word $u \in A^+$, we have

$$f(u) = [\![\mathrm{TC}_{x,y}\Psi]\!](u, 1, |u|)$$

The following result characterizes the expressive power of weighted automata.

THEOREM 4.1. *Let $\mathbb{S}$ be a (possibly non-commutative) semiring and $f \in \mathbb{S}\langle\!\langle A^+ \rangle\!\rangle$. The following assertions are equivalent over $\mathbb{S}$ and $A$.*

*(1) $f$ is recognizable.*
*(2) $f$ is definable by a formula $\mathrm{TC}^{N,<}_{x,y}\Phi$ with $\Phi$ (FO+mod)-step.*
*(3) $f$ is definable by a formula $\mathrm{TC}^{N,<}_{x,y}\Phi$ with $\Phi$ MSO-step.*
*(4) $f$ is definable by a formula $\mathrm{TC}^{<}_{x,y}\Phi$ with $\Phi$ (FO+mod)-step.*
*(5) $f$ is definable by a formula $\mathrm{TC}^{<}_{x,y}\Phi$ with $\Phi$ MSO-step.*
*(6) $f$ is definable by a formula $\bigoplus_X \bigotimes_x \Phi(x, X)$ with $\Phi$ FO-step.*
*(7) $f$ is definable by a formula $\bigoplus_X \bigotimes_x \Phi(x, X)$ with $\Phi$ MSO-step.*

PROOF. The implications 2 ⇒ 3, 4 ⇒ 5 and 6 ⇒ 7 are clear since FO+mod ⊆ MSO, and 7 ⇒ 1 follows from Theorem 3.8 because formulae described in 7 are included in sREMSO: indeed, because of Lemma 3.7, every product $\Psi \otimes \Psi'$ in $\Phi$ is such that $\Psi \in \mathrm{MSO}$ and $\Psi' = s \in \mathbb{S}$, henceforth the property on elementwise commutation is trivially verified. The implications 2 ⇒ 4 and 3 ⇒ 5 are also obvious since $\mathrm{TC}^{N,<}$ can be defined with $\mathrm{TC}^{<}$. We prove 1 ⇒ 2, 1 ⇒ 6, and 5 ⇒ 7.

For the first two implications, let us fix a weighted automaton $\mathcal{A} = (Q, A, I, F, \Delta, \mathsf{weight})$ with $Q = \{1, \dots, m\}$. Recall that $\mu$ denotes the morphism from $A^+$ to $\mathbb{S}^{Q \times Q}$ as defined in Section 2.2. The general idea behind the constructions we will present is that the semantics $[\![\mathcal{A}]\!](u)$ can be computed by splitting the word $u$ in small slices $u_1, \dots, u_k$ by $[\![\mathcal{A}]\!](u) =$

$\lambda \times \mu(u_1) \times \cdots \times \mu(u_k) \times \gamma$. As it is a priori not possible to compute a matrix of weight in the logic without changing the semiring, we must moreover partition the accepting runs considering the sequence of states they encounter at every positions between slice $u_i$ and $u_{i+1}$. Each state must then be encoded into the word so that a formula can find it to check the validity of the run and its weight: to do so, we consider portions $u_1, \ldots, u_k$ to be of size $m$ roughly, and position $q \in Q$ of $u_i$ encodes that state $q$ were encountered before reading $u_i$ in the current run. It is then necessary to compute the weight of runs over a slice. For $v \in A^+$ and $p, q \in Q$, we use an FO-step formula $\Xi^v_{p,q}(x)$ to compute the weight of a factor starting at position $x - p + 1$, labeled by the word $v$, when $\mathcal{A}$ goes from $p$ to $q$:

$$\Xi^v_{p,q}(x) \stackrel{\text{def}}{=} \mu(v)_{p,q} \otimes \bigwedge_{1 \leqslant k \leqslant |v|} P_{v_k}(x - p + k)\,.$$

We easily see that for every word $u$, and every positions $i, j \in \mathrm{pos}(u)$ with $i \leqslant j$ and $i + p - 1 \leqslant |u|$,

$$[\![\Xi^{u[i..j]}_{p,q}(x)]\!](u, i + p - 1) = \mu(u[i..j])_{p,q}\,. \tag{4}$$

Let us show 1 $\Rightarrow$ 2. We construct an (FO+mod)-step formula $\Phi_{\mathrm{long}}(x, y)$, such that $[\![\mathcal{A}]\!](u) = [\![\mathrm{TC}^{2m,<}_{x,y}\Phi_{\mathrm{long}}]\!](u, 1, |u|)$ for every word $u$ *sufficiently long* (recall that $m$ is the number of states of $\mathcal{A}$). As already mentioned, the idea, inspired by [Thomas 1982], consists in making the transitive closure operator pick positions $z_\ell = \ell m + q_\ell$, with $1 \leqslant q_\ell \leqslant m$, for successive values of $\ell$, to encode runs of $\mathcal{A}$ going through state $q_\ell$ just before reading the letter at position $\ell m + 1$. Consider slices $[\ell m + 1, (\ell + 1)m]$ of positions in the word where we evaluate the formula (the last slice might be incomplete). Each position $x$ is located in exactly one such slice, and the state $p$ encoded by such a position is given by the FO+mod formula $x \equiv_m p$. Moreover, the first position of the slice in which a position $x$ is located can be computed by $x - p + 1$, if $x \equiv_m p$. Since the transitive closure starts from the first position of the word, to make this still work for $\ell = 0$, let us assume without loss of generality that $I = \{1\}$.

Our $\mathrm{TC}^{N,<}$-formula picks positions $x$ and $y$ marked $\bullet$ in Figure 2, and computes the weight of the factor of length $m$ between positions $x - p + 1$ and $x - p + m$ (included), assuming states $x \equiv_m p$ and $y \equiv_m q$ just before and after these positions respectively.

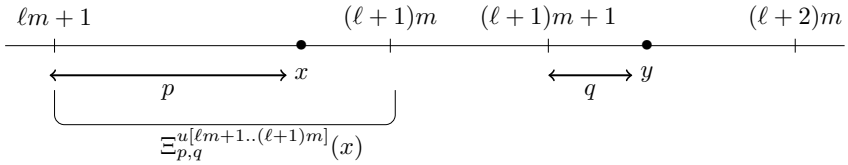


Fig. 2. Positions picked by the $\mathrm{TC}^{N,<}$-formula

The formula $\Phi_{\mathrm{long}}$ distinguishes the cases where $x$ is located far or near from the last position. More precisely, in the following expression of $\Phi_{\mathrm{long}}$, the first sum corresponds to the case where an entire slice is still available after the slice of $x$ (as in Figure 2), while the second sum captures the case where $x$ is located in the last complete slice.

$$\begin{aligned}\Phi_{\mathrm{long}}(x, y) = &\bigoplus_{p,q \in Q,\, v \in A^m} \big(x - p + 2m < \mathsf{last} \wedge x \equiv_m p \wedge y - q = x - p + m\big) \otimes \Xi^v_{p,q}(x) \\ &\oplus \bigoplus_{\substack{p \in Q, q_f \in F \\ m < d \leqslant 2m,\, v \in A^d}} \big(x - p + d = \mathsf{last} \wedge x \equiv_m p \wedge y = \mathsf{last}\big) \otimes \Xi^v_{p,q_f}(x)\,.\end{aligned}$$

Using (4), this definition implies that, for every word $u$ and all positions $\ell m+p$, $\ell' m+q$ in $u$, with $p, q \in Q$:

$$[\![\Phi_{\mathrm{long}}]\!](u, \ell m{+}p, \ell' m{+}q) = \begin{cases} \mu(u[\ell m + 1..\ell' m])_{p,q} & \text{if } \ell' = \ell + 1 \wedge (\ell+2)m < |u|, \\ \sum_{q_f \in F} \mu(u[\ell m + 1..|u|])_{p,q_f} & \text{if } \ell' = \ell + 1 \wedge \ell' m + q = |u|, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Notice that in the second case, we have $(\ell+2)m = \ell' m + m \geqslant \ell' m + q = |u|$. This ensures that the first two cases are disjoint. Now, by definition of the $\mathrm{TC}^{2m,<}_{x,y}$ operator,

$$[\![\mathrm{TC}^{2m,<}_{x,y}\Phi_{\mathrm{long}}(x,y)]\!](u,1,|u|) = \sum_{k \geqslant 1} [\![\Phi^k_{\mathrm{long}}(x,y)]\!](u,1,|u|)\,.$$

Observe that $\Phi_{\mathrm{long}}$ syntactically ensures the bounded and ordered condition of the transitive closure operator.

Suppose now that $|u| \geqslant m+1$. We show that there is at most one nonzero value in this sum: $[\![\Phi^k_{\mathrm{long}}(x,y)]\!](u,1,|u|)$ with $k = \lfloor \frac{|u|-1}{m} \rfloor$. So let $k$ be the largest integer $\ell$ such that $\ell m < |u|$.

Let $1 = i_0 < \ldots < i_h = |u|$ be a sequence of positions chosen in (2) such that $[\![\Phi_{\mathrm{long}}(x,y)]\!](u, i_\ell, i_{\ell+1}) \neq 0$ for all $\ell \in \{0, \ldots, h-1\}$. For $\ell \leqslant h-1$, we denote by $q_\ell \in Q$ the state such that $i_\ell \equiv_m q_\ell$. The first case of (5) must apply for $0 \leqslant \ell \leqslant h-2$, and it yields by induction $i_\ell = \ell m + q_\ell$ if $0 \leqslant \ell \leqslant h-1$. The second case of (5) applied with $\ell = h-1$ yields $|u| \leqslant (h+1)m$, whence $h = k = \lfloor \frac{|u|-1}{m} \rfloor$.

In particular, if $|u| \leqslant m$, then $[\![\mathrm{TC}^{2m,<}_{x,y}\Phi_{\mathrm{long}}(x,y)]\!](u,1,|u|) = 0$. If $|u| > m$, we deduce that $[\![\mathrm{TC}^{2m,<}_{x,y}\Phi_{\mathrm{long}}(x,y)]\!](u,1,|u|) = [\![\Phi^k_{\mathrm{long}}(x,y)]\!](u,1,|u|)$ for $k = \lfloor \frac{|u|-1}{m} \rfloor \geqslant 1$. The assignment $1 = i_0 < i_1 < \cdots < i_k = |u|$ is uniquely defined by the sequence $\overline{q} = (q_1, \ldots, q_{k-1}) \in Q^{k-1}$ (note that if $k = 1$, then $\overline{q}$ is empty). The possible choices of this tuple induce the sum in the following evaluation of $[\![\Phi^k_{\mathrm{long}}(x,y)]\!](u,1,|u|)$, where we omit the variable names $(x,y)$ in the right hand side. Recall that $q_0 = 1$ is the unique initial state giving a nonzero value.

$$\begin{aligned}
&[\![\mathrm{TC}^{2m,<}_{x,y}\Phi_{\mathrm{long}}(x,y)]\!](u,1,|u|) = [\![\Phi^k_{\mathrm{long}}]\!](u,1,|u|) \\
&\quad = \sum_{\overline{q} \in Q^{k-1}} \left( \Big[ \prod_{\ell=1}^{k-1} [\![\Phi_{\mathrm{long}}]\!]\big(u, (\ell-1)m + q_{\ell-1}, \ell m + q_\ell\big) \Big] \times [\![\Phi_{\mathrm{long}}]\!](u, (k-1)m + q_{k-1}, |u|) \right) \\
&\quad = \sum_{\overline{q} \in Q^{k-1}} \left( \Big[ \prod_{\ell=1}^{k-1} \mu\big(u[(\ell-1)m + 1..\ell m]\big)_{q_{\ell-1}, q_\ell} \Big] \times \sum_{q_f \in F} \mu(u[(k-1)m+1..|u|])_{q_{k-1}, q_f} \right) \\
&\quad = \sum_{q_f \in F} \mu(u)_{q_0, q_f} = [\![\mathcal{A}]\!](u)\,.
\end{aligned}$$

Finally, if $|u| \leqslant m$, we can design an FO-step formula $\Phi_{\mathrm{small}}$ dealing with small words:

$$\Phi_{\mathrm{small}} \stackrel{\mathrm{def}}{=} (x = \mathsf{first} \wedge y = \mathsf{last}) \otimes \bigoplus_{\substack{1 \leqslant d \leqslant m \\ v \in A^d, q_f \in F}} (\mathsf{last} = d) \otimes \Xi^v_{q_0, q_f}(\mathsf{first})\,.$$

Only the term for $v = u$ yields a nonzero value. Hence, $\mathrm{TC}^{2m,<}_{x,y}(\Phi_{\mathrm{small}} \oplus \Phi_{\mathrm{long}})$ computes $[\![\mathcal{A}]\!]$, as required.

Let us now outline the proof of 1 ⇒ 6, which relies on a similar technique. We use the monadic second-order variable $X$ to partition the runs along the portions $u = u_1 \cdots u_k$

considering the states in-between each portion. The set $X$ will contain for all $i$: $x_i$, the first position of $u_i$ and $t_i$, the position in $u_i$ used to encode the state of $Q$ reached before reading $u_i$. To distinguish $x_i$'s and $t_i$'s, we consider portions of size $2m+1$, so that between $x_i$ and $t_i$ the distance is no more than $m$, whereas the distance between $t_i$ and $x_{i+1}$ is at least $m$.

More formally, we encode runs of $\mathcal{A}$ by a formula $\bigoplus_X \bigotimes_x \Phi(x, X)$ with $\Phi$ an FO-step formula, enforcing $X$ to represent positions $x_0 < t_0 < x_1 < t_1 < \dots$ with $x_\ell = (2m+1)\ell+1$, where the distance $t_\ell - x_\ell$ encodes the state of $\mathcal{A}$ reached just before position $x_\ell$. For instance, since we assume as above that the only initial state yielding a nonzero value is $q_0 = 1$, we enforce $x_0 = 1$ and $t_0 = 2$, so that $t_0 - x_0 = 1$ encodes that state. We enforce $1 \leqslant t_\ell - x_\ell \leqslant m$ (so that $t_\ell - x_\ell \in Q$) but we make a gap between $t_\ell$ and $x_{\ell+1}$, namely $x_{\ell+1} - t_\ell > m$.

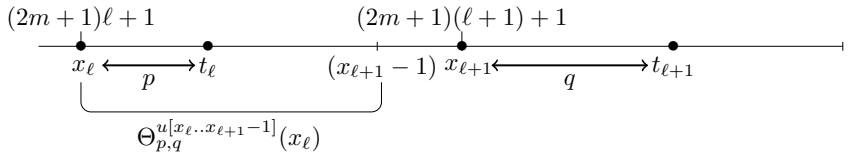


Fig. 3. Positions represented by $X$, marked $\bullet$

We use shortcuts like $\{z,t\} \subseteq X$, or $X \cap ]z,t] = \emptyset$, where $]z,t] = \{z' \mid z < z' \leqslant t\}$, which are easily FO-definable. Let us define

$$\begin{aligned}\Phi(x,X) = \quad & (\mathsf{last} > m \wedge \{1,2\} \subseteq X) \otimes (\Phi_{\mathrm{far}}(x,X) \oplus \Phi_{\mathrm{near}}(x,X)) \\ & \oplus (\mathsf{last} \leqslant m \wedge X = \emptyset) \otimes \big[x = \mathsf{first} \ominus \bigoplus_{\substack{1 \leqslant d \leqslant m \\ v \in A^d,\ q_f \in F}} (\mathsf{last} = d \otimes \Theta_{q_0,q_f}^v(x))\big].\end{aligned}$$

with $\Theta_{p,q}^v(x)$ a slightly modified version of $\Xi_{p,q}^v(x)$:

$$\Theta_{p,q}^v(x) \stackrel{\text{def}}{=} \mu(v)_{p,q} \otimes \bigwedge_{1 \leqslant k \leqslant |v|} P_{v_k}(x+k-1)$$

verifying, for every word $u$, and every positions $i, j \in \mathrm{pos}(u)$ with $i \leqslant j$,

$$[\![\Theta_{p,q}^{u[i..j]}(x)]\!](u,i) = \mu(u[i..j])_{p,q}\,. \tag{6}$$

The outermost sum in $\Phi$ distinguishes the cases of long or short words, while the sum $\Phi_{\mathrm{far}}(x,X) \oplus \Phi_{\mathrm{near}}(x,X)$ discriminates between positions $x \in X$ located far/near from the last position. For the case of short words, the product resulting from the quantification $\bigotimes_x$ has value $[\![\mathcal{A}]\!](u)$ thanks to the premise $x = \mathsf{first}$ of the implication $\ominus$. Moreover, the sum resulting from $\bigoplus_X$ has a single nonzero value thanks to $X = \emptyset$. Therefore, the overall semantics is indeed $[\![\mathcal{A}]\!](u)$. We define $\Phi_{\mathrm{far}}$ as

$$\begin{aligned}\Phi_{\mathrm{far}}(x,X) = (x+3m+1 \leqslant \mathsf{last}) \otimes \Big( & (x \in X \wedge X \cap ]x, x+m] \neq \emptyset) \ominus \\ & \bigoplus_{\substack{p,q \in Q \\ v = v_1 \cdots v_{2m+1}}} X \cap ]x, x+3m+1] = \{x+p, x+2m+1, x+2m+1+q\} \otimes \Theta_{p,q}^v(x)\Big).\end{aligned}$$

Note that $p$ and $q$ are uniquely defined by the set $X$. To produce a nonzero value, we must also have $x+3m+1 \leqslant \mathsf{last}$, so that letting $y = x+2m+1$, we have $y+m \leqslant \mathsf{last}$. Furthermore,

$y \in X \wedge X \cap ]y, y+m] \neq \emptyset$, hence the pattern repeats. We define $\Phi_{\text{near}}$ as

$$\Phi_{\text{near}}(x,X) = (x+3m+1 > \mathsf{last}) \otimes \Big( (x \in X \wedge X \cap ]x, x+m] \neq \emptyset) \ominus \bigoplus_{\substack{p\in Q, q_f \in F \\ m<d\leqslant 3m+1 \\ v=v_1\cdots v_d}} (X \cap ]x, \mathsf{last}] = \{x+p\} \wedge x+d-1 = \mathsf{last}) \otimes \Theta^v_{p,q_f}(x) \Big).$$

As FO-step formulae are closed by sum and product, the formula $\Phi(x,X)$ is an FO-step formula. Let us prove that $[\![\bigoplus_X \bigotimes_x \Phi(x,X)]\!](u) = [\![\mathcal{A}]\!](u)$. This equality has already been checked for words $u$ of length at most $m$, so let us assume that $|u| > m$. In the following, we denote by $\mathrm{Pat}(u)$ the set of all subsets $\{x_0 < t_0 < \ldots < x_r < t_r\}$ with $t_0 = 2$, $x_\ell = (2m+1)\ell+1$ and $1 \leqslant t_\ell - x_\ell \leqslant m$ for all $\ell \in \{0,\ldots,r\}$, and $x_r + m \leqslant |u| < x_r + 3m + 1$. Let $q_\ell = t_\ell - x_\ell$. We have already explained that if $[\![\bigotimes_x \Phi(x,X)]\!](u,I) \neq 0$, then we must have $I \in \mathrm{Pat}(u)$. Then, for a word $u$ of length greater than $m$, we get:

$$\begin{aligned}
&[\![\textstyle\bigoplus_X \bigotimes_x \Phi(x,X)]\!](u) \\
&\quad = \sum_{I\in\mathrm{Pat}(u)} \prod_{1\leqslant i\leqslant |u|} \big( [\![\Phi_{\text{far}}]\!](u,i,I) + [\![\Phi_{\text{near}}]\!](u,i,I) \big) \\
&\quad = \sum_{I\in\mathrm{Pat}(u)} \Big( \prod_{1\leqslant i\leqslant |u|-3m-1} [\![\Phi_{\text{far}}]\!](u,i,I) \times \prod_{\max(1,|u|-3m)\leqslant i\leqslant |u|} [\![\Phi_{\text{near}}]\!](u,i,I) \Big) \\
&\quad = \sum_{I\in\mathrm{Pat}(u)} \Big( \prod_{0\leqslant \ell\leqslant r-1} [\![\Theta^{u[x_\ell..x_{\ell+1}-1]}_{q_\ell,q_{\ell+1}}(x)]\!](u,x_\ell) \times \sum_{q_f\in F} \big( [\![\Theta^{u[x_r..|u|]}_{q_r,q_f}(x)]\!](u,x_r) \big) \Big).
\end{aligned}$$

The last equality holds since $[\![\Phi_{\text{near}}]\!](u,x_r,I) = \sum_{q_f\in F} \big( [\![\Theta^{u[x_r..|u|]}_{q_r,q_f}(x)]\!](u,x_r) \big)$ and since $[\![\Phi_{\text{near}}]\!](u,i,I) = 1$ if $i \in \{\max(1,|u|-3m),\ldots,|u|\} \setminus \{x_r\}$. Finally, by (6),

$$\begin{aligned}
[\![\textstyle\bigoplus_X \bigotimes_x \Phi(x,X)]\!](u) &= \sum_{\substack{q_0,\ldots,q_r\in Q \\ q_0=1, q_f\in F}} \Big( \prod_{0\leqslant\ell\leqslant r-1} \mu(u[x_\ell..x_{\ell+1}-1]])_{q_\ell,q_{\ell+1}} \times \mu(u[x_r..|u|])_{q_r,q_f} \Big) \\
&= \sum_{q_f\in F} \mu(u)_{q_0,q_f} = [\![\mathcal{A}]\!](u).
\end{aligned}$$

As in [Thomas 1982], we could use a more compact formula by encoding states in binary.

We finally prove 5 ⇒ 7. Assume that $\Phi$ is of the form $\Phi(x,y) = \bigoplus_{i\in I} \varphi_i(x,y) \otimes s_i$. Again, we use the monadic second-order variable $X$ to encode the *runs* of the weighted transitive closure operator. Let $u \in A^+$ be a word, $j \in \mathrm{pos}(u)$ and $J \subseteq \mathrm{pos}(u)$. If $j < \max(J)$ then we let $\mathrm{next}(j,J) = \inf\{\ell \in J \mid j < \ell\}$. We define the MSO-step formula

$$\Phi'(x,X) \stackrel{\text{def}}{=} \bigoplus_{i\in I} s_i \otimes \exists y (x < y \wedge X \cap ]x,y] = \{y\} \wedge \varphi_i(x,y))$$

and we can easily check that

$$[\![\Phi']\!](u,j,J) = \begin{cases} [\![\Phi]\!](u,j,\mathrm{next}(j,J)) & \text{if } j < \max(J) \\ 0 & \text{otherwise.} \end{cases}$$

The formula $\mathrm{TC}^<_{x,y}\Phi(\mathsf{first},\mathsf{last})$ is then equivalent to $\bigoplus_X \bigotimes_x (\Psi_{\text{small}} \oplus \Psi)$, with

$$\begin{aligned}
\Psi &\stackrel{\text{def}}{=} \big(\mathsf{first} \neq \mathsf{last} \wedge \{\mathsf{first},\mathsf{last}\} \subseteq X\big) \otimes \big( (x \in X \wedge x < \max(X)) \ominus \Phi'(x,X) \big) \\
\Psi_{\text{small}} &\stackrel{\text{def}}{=} \big(X = \emptyset \wedge \mathsf{first} = \mathsf{last}\big) \otimes \big( (x = \mathsf{first}) \ominus \Phi(x,x) \big).
\end{aligned}$$

Note that MSO-step formulae are closed under sum and products so the formula above is indeed in the required fragment. Let $u \in A^+$. In case, $|u| = 1$, we have $[\![\bigoplus_X \bigotimes_x (\Psi_{\text{small}} \oplus$

$\Psi)]\!](u) = [\![\Phi]\!](u,1,1) = [\![\mathrm{TC}^{<}_{x,y}\Phi]\!](u,1,|u|)$. Otherwise, with the conditions on $X$ in the formulae $\Psi_{\mathrm{small}}$ and $\Psi$ we have

$$[\![\bigoplus_X \bigotimes_x (\Psi_{\mathrm{small}} \oplus \Psi)]\!](u) = \sum_{\{1,|u|\}\subseteq J\subseteq\{1,\dots,|u|\}} [\![\bigotimes_x \Psi]\!](u,[X\mapsto J])\,.$$

For a choice of $J=\{j_0,j_1,\dots,j_k\}$ with $1=j_0<j_1<\cdots<j_k=|u|$, by definition of $\Psi$ we have $[\![\bigotimes_x \Psi]\!](u,[X\mapsto J]) = \prod_{0\leqslant r<k}[\![\Phi]\!](u,j_r,j_{r+1})$. Finally, we obtain

$$\begin{aligned}[\![\bigoplus_X \bigotimes_x (\Psi_{\mathrm{small}} \oplus \Psi)]\!](u) &= \sum_{k>0} \sum_{1=j_0<j_1<\cdots<j_k=|u|} \prod_{0\leqslant r<k} [\![\Phi]\!](u,j_r,j_{r+1}) \\ &= [\![\mathrm{TC}^{<}_{x,y}\Phi]\!](u,1,|u|)\end{aligned}$$

where the last equality comes from the definition of the ordered weighted transitive closure operator. □

## 5. PEBBLE WEIGHTED AUTOMATA

Example 3.2 shows that weighted automata lack closure properties to capture the logic $\mathrm{wFOTC}^{\mathrm{b},<}(\mathrm{FO})$. We introduce a notion of two-way pebble weighted automata making up for this gap. In the Boolean setting, two-way automata over strings have been first considered in [Rabin and Scott 1959], whereas pebble automata have been introduced in [Blum and Hewitt 1967].

### 5.1. General definitions

We now consider *pebble 2-way weighted automata* (P2WA). A P2WA has a read-only tape. At each step, it can move its head one position to the left or to the right (within the boundaries of the input tape), or drop a pebble at the current head position, or lift a pebble and resume the tape to its position. Applicable transitions and weights depend on the current letter and state, and on which pebbles are carried by the current position. Pebbles are numbered $p,\dots,2,1$, and are handled using a stack policy: if pebbles $p,\dots,\ell$ have already been dropped, the automaton can either lift pebble $\ell$ (if $\ell\leqslant p$), drop pebble $\ell-1$ (if $\ell>1$), or move.

As these automata can go in either direction, we add two fresh symbols $\triangleright$ and $\triangleleft$ to mark the beginning and the end of an input word. Let $\widetilde{A}=A\uplus\{\triangleright,\triangleleft\}$. To compute the value of $u=u_1\cdots u_n\in A^+$, a P2WA works on a tape holding $\triangleright u\triangleleft$. For convenience, we number the letters of $\triangleright u\triangleleft$ from 0, setting $u_0=\triangleright$ and $u_{n+1}=\triangleleft$. We denote by $\widetilde{\mathrm{pos}}(u)$ the set $\{0,1,\dots,n,n+1\}$.

*Definition* 5.1. Let $p\geqslant 0$. A *pebble two-way weighted automaton (*P2WA*) with $p$ pebbles* over $\mathbb{S}$ is a tuple $\mathcal{A}=(Q,A,I,F,\Delta,\mathsf{weight})$ where $Q$ is a finite set of states, $A$ is a finite alphabet, $I\subseteq Q$ is the set of initial states, $F\subseteq Q$ is the set of final states, $\Delta\subseteq Q\times\widetilde{A}\times 2^{\{1,\dots,p\}}\times D\times Q$ is the set of transitions, with $D=\{\rightarrow,\leftarrow,\mathsf{drop},\mathsf{lift}\}$ and $\mathsf{weight}:\Delta\to\mathbb{S}$ is the weight function.

A *configuration* of a P2WA $\mathcal{A}$ with $p$ pebbles on a word $u\in A^+$ of length $n$ is a triple $(q,\pi,i)\in Q\times\mathrm{pos}(u)^{\leqslant p}\times\widetilde{\mathrm{pos}}(u)$. In such a configuration, $q$ denotes the current state of $\mathcal{A}$, while $i$ is the current head position, and $\pi=\pi_p\cdots\pi_\ell$ with $1\leqslant\ell\leqslant p+1$ encodes the locations of the $p+1-\ell$ currently dropped pebbles. More precisely, $\pi_m\in\{1,\dots,n\}$ is the position of pebble $m$, while pebbles $\ell-1,\dots,1$ are currently not on the tape. Given such a word of pebbles $\pi=\pi_p\cdots\pi_\ell$, we denote by $\mathrm{peb}(\pi,i)$ the set of pebbles currently dropped on position $i\in\widetilde{\mathrm{pos}}(u)$, i.e., $\mathrm{peb}(\pi,i)=\{j\mid p\geqslant j\geqslant\ell\wedge\pi_j=i\}$.

There is a *step* of weight $s$ from configuration $(q, \pi, i)$ to configuration $(q', \pi', i')$ if there exists $d \in D$ such that $\delta \stackrel{\text{def}}{=} (q, u_i, \mathrm{peb}(\pi, i), d, q') \in \Delta$ with $s = \mathsf{weight}(\delta)$, and

$$\begin{cases} \pi' = \pi \text{ and } i' = i+1 & \text{if } d = \rightarrow \\ \pi' = \pi \text{ and } i' = i-1 & \text{if } d = \leftarrow \\ \pi' = \pi \cdot i \text{ and } i' = 0 & \text{if } d = \mathsf{drop} \text{ and } i \in \mathrm{pos}(u) \\ \pi' \cdot i' = \pi & \text{if } d = \mathsf{lift}\,. \end{cases} \tag{7}$$

Note that a drop simply records the current position in $u$, with a fresh pebble (such a pebble should be available) and returns to the beginning of the tape, and that a lift returns on the last dropped pebble and pops it. This model corresponds to a weighted extension of the well-known *weak pebble automata* (as described, e.g., in [Bojańczyk et al. 2006]), as we can easily design *drop* transitions that do not reset the tape, and force *lift* transitions to occur only at the position of the topmost pebble. Notice that if it exists, the direction $d$ is determined by the two configurations.

A *run* $\rho$ of $\mathcal{A}$ on $u$ is a sequence of configurations related by steps. We denote by $\mathsf{weight}(\rho)$ the product of the weights of the steps of the run $\rho$ (from left to right, but we will mainly work with commutative semirings in this section). A run is said to be *accepting* if it leads from a configuration $(q, \varepsilon, 0)$, with $q \in I$, to a configuration $(q', \varepsilon, n+1)$, with $q' \in F$ (at the end, no pebble is left on the tape). The run $\rho$ is *simple* if whenever two configurations $\alpha$ and $\beta$ appear in $\rho$, we have $\alpha \neq \beta$. The series $[\![\mathcal{A}]\!] \in \mathbb{S}\langle\langle A^+ \rangle\rangle$ is defined by $[\![\mathcal{A}]\!](u) = \sum \mathsf{weight}(\rho)$ where the sum ranges over all *simple* accepting runs $\rho$ on $u$. Note that a P2WA with 0 pebble is in fact a 2-way weighted automaton. It follows from Proposition 5.9 below that this class of automata has the same expressive power as 1WA.

*Example* 5.2. Let us sketch a P2WA $\mathcal{A}$ with 1 pebble recognizing the series $u \mapsto 2^{|u|^2}$ over $\mathbb{N}$. The idea is that $\mathcal{A}$ drops its pebble successively on every position of the input word. Transitions for reallocating the pebble have weight 1. When a pebble is dropped, $\mathcal{A}$ scans the whole word from left to right where every transition has weight 2. As this scan happens $|u|$ times, we obtain $[\![\mathcal{A}]\!](u) = 2^{|u|^2}$.

### 5.2. Pebble one-way weighted automata

The notion of simplicity is semantic. Our goal in this section is to find a syntactic restriction of P2WA enforcing simplicity. Mainly, it will consist in restricting to *1-way* runs, disabling left moves.

*Definition* 5.3. A *pebble one-way weighted automaton (*P1WA*)* over $\mathbb{S}$ is a P2WA $\mathcal{A} = (Q, A, I, F, \Delta, \mathsf{weight})$ which satisfies the following conditions:

(1) There are no left transitions: $(q, a, P, d, q') \in \Delta$ implies $d \neq \leftarrow$;
(2) Lift transitions may not be followed by drop transitions: $(q, a, P, \mathsf{lift}, q') \in \Delta$ and $(q', a', P', d', q'') \in \Delta$ imply $d' \neq \mathsf{drop}$;

To justify (2), note that a lift transition followed by a drop transition would just result in resetting the current position to the beginning of the word, without dropping a new pebble.

The P2WA described in Example 5.2 is in fact a P1WA by definition.

As a fragment of P2WA, the semantics of a P1WA is well defined. However, the semantic restriction (considering only simple runs to define the weight of a word) is now useless, since every run of a P1WA is ensured to be simple, as shown in the next lemma.

Lemma 5.4. *All runs of a* P1WA *are simple.*

Proof. Consider a P1WA $\mathcal{A} = (Q, A, I, F, \Delta, \mathsf{weight})$. Given a word $u$, we define a total order on the set of configurations of $\mathcal{A}$ over $u$, so that runs of $\mathcal{A}$ over $u$ form an increasing sequence of configurations.

To do so, we consider the *measure* of a configuration $(q, \pi, i)$ to be the word over the alphabet $\widetilde{\mathrm{pos}}(u) \uplus \{\bot, \top\}$ defined by

$$\mathrm{meas}(q, \pi, i) = \begin{cases} \pi \cdot i \cdot \bot & \text{if } \exists\, (q, a, P, \mathsf{drop}, q') \in \Delta \\ \pi \cdot i \cdot \top & \text{otherwise.} \end{cases}$$

We order the set $\widetilde{\mathrm{pos}}(u) \uplus \{\bot, \top\}$ by $\bot \prec 0 \prec 1 \prec \cdots \prec |u| \prec |u|+1 \prec \top$. Henceforth, finite words over this set are ordered with the associated lexicographic (total) order, denoted $\prec$.

We now show that for every run $\rho = (q_m, \pi_m, i_m)_{0 \leqslant m \leqslant h}$ of $\mathcal{A}$ over $u$, the sequence of measures of the configurations is increasing, i.e., $\mathrm{meas}(q_m, \pi_m, i_m) \prec \mathrm{meas}(q_{m+1}, \pi_{m+1}, i_{m+1})$ for all $0 \leqslant m < h$. This ensures that the run is simple. Let $0 \leqslant m < h$. The step from configuration $\gamma_m = (q_m, \pi_m, i_m)$ to configuration $\gamma_{m+1} = (q_{m+1}, \pi_{m+1}, i_{m+1})$ is enabled with a transition of the form $(q_m, u_{i_m}, \mathrm{peb}(\pi_m, i_m), d, q_{m+1})$ with $d$ defined as in (7) (specialized to the one-way case):

— If $d = \rightarrow$, then $\pi_{m+1} = \pi_m$ and $i_{m+1} = i_m + 1$. Hence, for some $\alpha, \beta \in \{\bot, \top\}$ we have

$$\mathrm{meas}(\gamma_m) = \pi_m i_m \alpha \prec \pi_m (i_m + 1)\beta = \mathrm{meas}(\gamma_{m+1})\,.$$

— If $d = \mathsf{drop}$, then $\pi_{m+1} = \pi_m i_m$ and $i_{m+1} = 0$. Hence, for some $\alpha \in \{\bot, \top\}$ we have

$$\mathrm{meas}(\gamma_m) = \pi_m i_m \bot \prec \pi_m i_m 0 \alpha = \mathrm{meas}(\gamma_{m+1})\,.$$

— if $d = \mathsf{lift}$, then $\pi_m = \pi_{m+1} i_{m+1}$. Hence, for some $\alpha \in \{\bot, \top\}$ we have

$$\mathrm{meas}(\gamma_m) = \pi_{m+1} i_{m+1} i_m \alpha \prec \pi_{m+1} i_{m+1} \top = \mathrm{meas}(\gamma_{m+1})\,.$$

Notice that the $\top$ in $\mathrm{meas}(\gamma_{m+1})$ comes from the fact that lift transitions may not be followed by drop transitions, by definition of P1WA.

In all cases, we have $\mathrm{meas}(\gamma_m) \prec \mathrm{meas}(\gamma_{m+1})$, which concludes the proof. □

*Example* 5.5 (*Example 3.6 continued*). The value computed to check that the counter always stays non-negative can also be calculated with the P1WA with 1 pebble of Figure 4: this automaton drops a pebble on each letter *Dec* and then computes the difference between the number of *Inc*'s and the number of *Dec*'s before the pebble. This is done as in Example 2.3. In the transitions of Figure 4 (the highlighted zone will be explained below), $*$ stands for any letter. We fix a word $u$, and denote by $i_1 < i_2 < \cdots < i_m$ the positions of $u$ labelled by *Dec*. Notice that each accepting run of the automaton drops the pebble on each position $i_k$ (for every $1 \leqslant k \leqslant m$), and, after each such drop, must choose a position $j_k$ to follow the transition from state 2 to state 3: moreover, we have $1 \leqslant j_k < i_k$ and $u_{j_k} \in \{Inc, Dec\}$. Denoting $J_k$ the set of such positions $j_k$ for every $k$, the set of accepting runs is therefore in bijection with the set $J = J_1 \times \cdots \times J_m$. The weight of the run defined by the tuple $(j_k)_{1 \leqslant k \leqslant m}$ is then given by

$$\prod_{k=1}^{m} f(j_k)$$

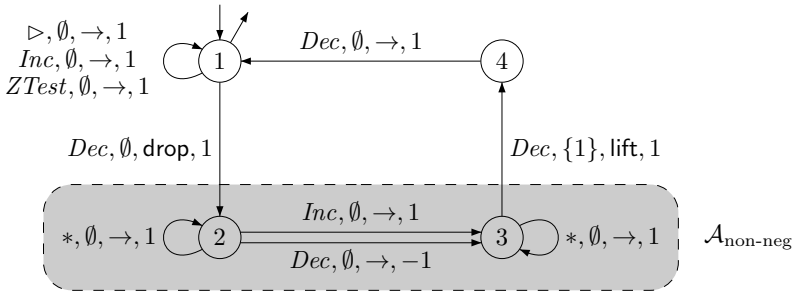


Fig. 4. Automaton checking that the counter is non-negative

where $f(j) = 1$ if $u_j = Inc$, $-1$ if $u_j = Dec$, and 0 otherwise. Hence, the semantics of the automaton over the word $u$ is given by

$$\begin{aligned}\sum_{(j_k)_{1\leqslant k\leqslant m}\in J}\prod_{k=1}^{m} f(j_k) &= \sum_{j_1\in J_1}\sum_{j_2\in J_2}\cdots\sum_{j_m\in J_m}\prod_{k=1}^{m} f(j_k)\\ &= \prod_{k=1}^{m}\sum_{j_k\in J_k} f(j_k) \qquad \text{(by distributivity of } \times \text{ over } +)^{2}\\ &= \prod_{k=1}^{m}\big(|u[1..i_k-1]|_{Inc} - |u[1..i_k-1]|_{Dec}\big)\end{aligned}$$

Next, assuming that the counter always stays non-negative, the value checking that a zero-test is used only when the counter is indeed 0 can be computed with the P1WA with 2 pebbles of Figure 5 (highlighted zone to be explained below). Intuitively, it drops pebble 2 sequentially on every position performing action *ZTest* to compute the external product. When this pebble is dropped (over a position $j$), on every position $k$ before the one holding the pebble (i.e., $k \leqslant j-1$), it drops pebble 1 choosing non-deterministically either to compute

$$|u[1..j-1]|_{Inc} - |u[1..j-1]|_{Dec}$$

with the upper part, or $-k$ with the lower part. The proof of correctness is done similarly to the argument for the previous automaton, and strongly relies again on the distributivity of the multiplication over the addition.

### 5.3. One-way versus two-way

We proceed with the most difficult result of this paper: the equivalence between P2WA and P1WA when the semiring is commutative.

[2]Notice the transformation from the term on the left of the equality (a sum of products of weights) to this term (a product of sums of weights). The same transformation is actually behind the two alternative definitions of the semantics of a weighted automaton (see Section 2.2): whereas the definition based on runs is a sum of products, the one based on the morphism $\mu$ is a product of matrices. The idea yielding the second one from the first is to gather together runs with respect to common configurations: the weight of a word split into two factors $uv$ is the product of the weights over $u$ and $v$, each produced by summing the weights of all sub-runs over $u$ and $v$, respectively. In the setting of pebble automata, we cannot simply split the word since the automaton may navigate and drop pebbles. However, if every accepting run goes through a given configuration exactly once, we may consider the sub-runs that start and end in this configuration, and group the weights similarly. In the present computation, we consider $m$ such configurations, namely those where the pebble has just been dropped.

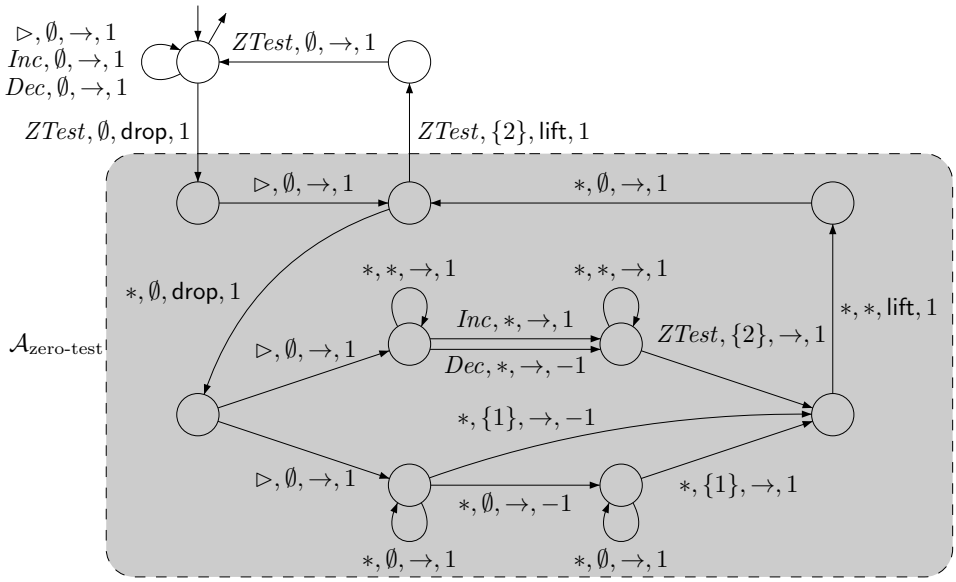


Fig. 5. Automaton checking the zero-tests

THEOREM 5.6. *Let* $\mathbb{S}$ *be a* commutative *semiring and* $f \in \mathbb{S}\langle\langle A^+ \rangle\rangle$. *Then,* $f$ *is recognizable by a* P2WA *if and only if* $f$ *is recognizable by a* P1WA.

One direction directly follows from the definitions of pebble weighted automata. The proof of the converse is by induction on the number of pebbles. We start with the base case of automata without pebbles. We simply write 2WA instead of "P2WA with 0 pebble".

The next proposition is a generalization of the result, originally proved in [Rabin and Scott 1959], stating that two-way feature does not add expressive power to non-deterministic finite automata over finite strings. There are mainly two proof techniques for this result. The proof in [Shepherdson 1959] starts with a two-way automaton and constructs an equivalent one-way automaton by enriching the state with a relation table recording for every pair of states $(q, q')$, whether it is possible to reach $q'$ from $q$ with a loop on the current prefix. The key argument is that there is a finite number of such tables and that it can be computed simultaneously to the main run.

Observe that this method is not applicable in our weighted setting. Indeed, the relation table should now be enriched with the precomputed weights of the subruns, which cannot be recorded within a finite memory (as the weights may grow with the length of the prefix read so far). Instead, we will use the crossing-sequence method which is closer to the proof technique in [Rabin and Scott 1959], and fully explained in [Hopcroft and Ullman 1979]. For the sake of clarity, we provide a complete proof below in our weighted setting.

PROPOSITION 5.7. *Let* $\mathbb{S}$ *be a* commutative *semiring,* $f \in \mathbb{S}\langle\langle A^+ \rangle\rangle$. *If* $f$ *is recognizable by a* 2WA, *then* $f$ *is recognizable by a* 1WA.

PROOF. Let $\mathcal{A} = (R, A, I, F, \Delta, \mathsf{weight})$ be a 2WA. We omit the (empty) set of pebbles in the transitions of the P2WA $\mathcal{A}$, i.e., $\Delta \subseteq R \times \widetilde{A} \times D \times R$. We also assume that $\mathsf{weight}$ is fully defined on $R \times \widetilde{A} \times D \times R$ by setting $\mathsf{weight}(\delta) = 0$ if $\delta \notin \Delta$.

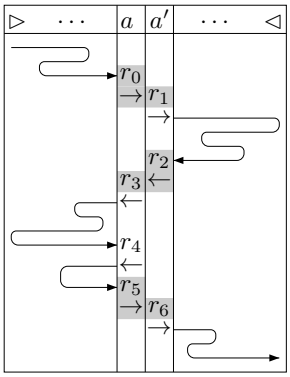


Fig. 6. Run of a 2WA and some crossing-sequences

We construct an equivalent 1WA (actually a P1WA without pebble[3]) $\mathcal{B} = (Q', A, I', F', \Delta', \mathsf{weight}')$ such that $[\![\mathcal{B}]\!] = [\![\mathcal{A}]\!]$. Again, we omit the (empty) set of pebbles and the $(\to)$ direction in the transitions of the P1WA $\mathcal{B}$, i.e., $\Delta' \subseteq Q' \times \widetilde{A} \times Q'$.

Intuitively, the idea is to record in a state of $\mathcal{B}$ the *crossing-sequence* of states of $\mathcal{A}$ which is observed in some accepting run $\rho$ while scanning the current position of the input word, see Figure 6. Since the semantics of a 2WA is computed as the sum over *simple* accepting runs, a crossing-sequence consists of pairwise distinct states and is therefore bounded by the number of states of $\mathcal{A}$. Finally, the commutativity of the semiring allows us to compute the weight of an accepting run $\rho$ along a corresponding run of $\mathcal{B}$.

In order to ease the matching of consecutive crossing-sequences of $\mathcal{A}$, we also record in a state of $\mathcal{B}$ the current input letter from $\widetilde{A} = A \cup \{\rhd, \lhd\}$ and the sequence of moves from $\{\leftarrow, \to\}$ performed by the 2WA. Therefore, we will define $Q' = Q'_\rhd \cup Q'_A \cup Q'_\lhd$ as a *finite* subset of $T = \widetilde{A}(R\{\leftarrow, \to\})^*(R \cup R\to)$. We say that a word in $T$ is *R-simple* if it does not contain two occurrences of some state in $R$.

While scanning $\rhd$ only right moves are possible hence we let $Q'_\rhd$ be the (finite) set of *R-simple* words in $\rhd(R\to)^+$. Similarly, while scanning $\lhd$, only left moves are possible until $\mathcal{A}$ finally halts on this position. Hence we let $Q'_\lhd$ be the (finite) set of *R-simple* words in $\lhd(R\leftarrow)^*R$. Finally, we let $Q'_A$ be the (finite) set of *R-simple* words in $A(R\{\leftarrow, \to\})^*R\to$. For instance, Figure 6 exhibits two crossing-sequences in $Q'_A$ which are $q = ar_0\to r_3\leftarrow r_4\leftarrow r_5\to$ and $q' = a'r_1\to r_2\leftarrow r_6\to$.

We describe now the set $\Delta' \subseteq Q' \times \widetilde{A} \times Q'$ of transitions of $\mathcal{B}$. Two states $a\tau$ and $a'\tau'$ in $Q'$ can be *matched* in a transition $(a\tau, a, a'\tau') \in \Delta'$ of $\mathcal{B}$ if $a \neq \lhd$, $a' \neq \rhd$ and $\tau$ has one more right moves than the number of left moves in $\tau'$: $|\tau|_\to = 1 + |\tau'|_\leftarrow$.

The weight of a transition $(q, a, q') \in \Delta'$ is the product of the weights of the transitions of $\mathcal{A}$ *contained* in the pair $(q, q')$, i.e., right transitions from $q$ to $q'$ and left transitions from $q'$ to $q$. On Figure 6, these transitions are on a gray background.

Formally, a right transition $(r, a, \to, r')$ of $\mathcal{A}$ is *contained* in $(q, q')$ if $r$ is the state just before the $k$th occurrence of $\to$ in $q$ and $r'$ the state just after the $(k-1)$th occurrence[4] of $\leftarrow$ in $q'$, for some $1 \leqslant k \leqslant |q|_\to$. Similarly, a left transition $(r', a', \leftarrow, r)$ of $\mathcal{A}$ is contained in $(q, q')$ if $r'$ the state just before the $k$th occurrence of $\leftarrow$ in $q'$ and $r$ is the state just after the $k$th occurrence of $\to$ in $q$, for some $1 \leqslant k \leqslant |q'|_\leftarrow$. For instance in Figure 6, the

[3]Note that, formally, a P1WA without pebble is not a 1WA as defined in Section 2.2. This is mainly due to the $\rhd$ marker. But it should be clear that the two models are equivalent.

[4]In case $k = 1$, the state just after the 0th occurrence of $\leftarrow$ in $q'$ is simply the first state of $q'$.

two right transitions from $\mathcal{A}$ contained in the depicted pair of states are $(r_0, a, \rightarrow, r_1)$ and $(r_5, a, \rightarrow, r_6)$, whereas the only left transition is $(r_2, a', \leftarrow, r_3)$.

We let $\mathsf{Trans}(q, q')$ be the set of (left or right) transitions of $\mathcal{A}$ contained in $(q, q')$. We then define the weight of $\rightarrow$ moves of $\mathcal{B}$ induced by the matching states $(q, q')$ as the product of the weights of the (left or right) transitions of $\mathcal{A}$ contained in $(q, q')$:

$$\mathsf{weight}'(q, a, q') = \prod_{\delta \in \mathsf{Trans}(q,q')} \mathsf{weight}(\delta)\,.$$

For instance, transition from $q = ar_0{\rightarrow}r_3{\leftarrow}r_4{\leftarrow}r_5{\rightarrow}$ to $q' = a'r_1{\rightarrow}r_2{\leftarrow}r_6{\rightarrow}$ on Figure 6 has weight

$$\mathsf{weight}'(q, a, q') = \mathsf{weight}(r_0, a, \rightarrow, r_1) \times \mathsf{weight}(r_2, a', \leftarrow, r_3) \times \mathsf{weight}(r_5, a, \rightarrow, r_6)\,.$$

Finally, the set $I'$ of initial states of $\mathcal{B}$ is the set of $R$-simple words in $\triangleright I{\rightarrow}(R{\rightarrow})^*$ and the set $F'$ of final states of $\mathcal{B}$ is the set of $R$-simple words in $\triangleleft(R{\leftarrow})^*F$.

Lemma 5.8 below proves that there is a weight preserving bijection between the simple accepting runs of $\mathcal{A}$ and the accepting runs of $\mathcal{B}$. We conclude that $[\![\mathcal{B}]\!] = [\![\mathcal{A}]\!]$ since for all $u \in A^+$, $[\![\mathcal{A}]\!](u)$ is the sum of the weights of the simple accepting runs of $\mathcal{A}$ over $u$, whereas $[\![\mathcal{B}]\!](u)$ is the sum of the weights of the accepting runs of $\mathcal{B}$ over $u$. □

LEMMA 5.8. *For every word $u \in A^+$, there is a weight preserving bijection between the simple accepting runs of $\mathcal{A}$ over $u$ and the accepting runs of $\mathcal{B}$ over $u$.*

PROOF. Fix a word $u = u_1 \cdots u_n \in A^+$ of length $n \geqslant 1$. We first show how to map a simple accepting run $\rho = \rho_0\rho_1 \cdots \rho_m$ of $\mathcal{A}$ over $u$ (where each $\rho_j$ is a configuration) to an accepting run of $\mathcal{B}$ over $u$.

For $i \in \{0, \ldots, n+1\}$, we construct the crossing-sequence $q_i = u_i\tau_i \in Q'$ of $\mathcal{B}$ associated with position $i$ in $\triangleright u \triangleleft$, as illustrated in Figure 6. Recall that $u_0 = \triangleright$ and $u_{n+1} = \triangleleft$. Formally, we define $\tau_i$ as the "projection" of the configurations visiting position $i$, where configuration $(r, i)$ is "projected" to $r{\rightarrow}$ (respectively, $r{\leftarrow}$ or $r$) if it is followed by some $(r', i+1)$ (respectively, $(r', i-1)$ or is the last configuration). Note that the word $\tau_i$ is $R$-simple. In particular, we have $q_0 \in I'$, $q_{n+1} \in F'$, and $q_i \in Q'_A$ for every $1 \leqslant i \leqslant n$.

Observe also that $|\tau_i|_{\rightarrow} = 1 + |\tau_{i+1}|_{\leftarrow}$ for every $0 \leqslant i \leqslant n$. Indeed, $\rho$ must reach position $i+1$ for a first time, and then, each time the run $\rho$ goes to the left of position $i+1$, it will in the future come back to $i+1$ from $i$. Therefore, $(q_i, u_i, q_{i+1}) \in \Delta'$ for all $0 \leqslant i \leqslant n$. We deduce that $f(\rho) = (q_0, 0)(q_1, 1) \cdots (q_{n+1}, n+1)$ is an accepting run of $\mathcal{B}$.

We now show that $f$ preserves the weights, i.e., $\mathsf{weight}(\rho) = \mathsf{weight}'(f(\rho))$ for all simple accepting run $\rho$ of $\mathcal{A}$ over $u$. Indeed, $\mathsf{weight}(\rho)$ is the product of the weights of (left or right) transitions appearing in $\rho$. With $f(\rho) = (q_0, 0)(q_1, 1) \cdots (q_{n+1}, n+1)$, every right transition of $\rho$ starting from position $i$ $(0 \leqslant i \leqslant n)$ is contained in the pair $(q_i, q_{i+1})$, so that its weight is computed in the transition from $i$ to $i+1$ in $f(\rho)$, whereas every left transition of $\rho$ starting from position $i$ $(1 \leqslant i \leqslant n+1)$ is contained in the pair $(q_{i-1}, q_i)$, so that its weight is computed in the transition from $i-1$ to $i$ in $f(\rho)$. From the definition of $\mathsf{weight}'$ and by commutativity of $\mathbb{S}$, we obtain $\mathsf{weight}(\rho) = \mathsf{weight}'(f(\rho))$.

We finally prove that $f$ is a bijection by constructing its inverse function. Let $(q_0, 0)(q_1, 1) \cdots (q_{n+1}, n{+}1)$ be an accepting run of $\mathcal{B}$ over $u$. Write $q_i = u_i\tau_i$ for $0 \leqslant i \leqslant n{+}1$.

We recover a simple accepting run $\rho$ of $\mathcal{A}$ over $u$ as the output of $\mathrm{Run}(0, \tau_0, \tau_1, \ldots, \tau_{n+1})$ where the recursive function Run is defined by:

**Function** $\mathrm{Run}(i, \tau_0, \tau_1, \ldots, \tau_{n+1})$

**match** $\tau_i$ **with**
- $r \rightarrow \tau_i'$ :
  - **output** $(r, i)$;
  - `Run` $(i+1, \tau_0, \ldots, \tau_i', \ldots, \tau_{n+1})$;
- $r \leftarrow \tau_i'$ :
  - **output** $(r, i)$;
  - `Run` $(i-1, \tau_0, \ldots, \tau_i', \ldots, \tau_{n+1})$;
- $r$ : `(* necessarily` $i = n+1$ `*)`
  - **output** $(r, n+1)$;

We can show by induction that $f(\mathtt{Run}(0, \tau_0, \tau_1, \ldots, \tau_{n+1})) = (q_0, 0)(q_1, 1) \cdots (q_{n+1}, n+1)$, and that for every simple accepting run $\rho$ of $\mathcal{A}$ with $f(\rho) = (u_0\tau_0, 0) \cdots (u_{n+1}\tau_{n+1}, n+1)$, we have $\mathtt{Run}(0, \tau_0, \ldots, \tau_{n+1}) = \rho$. Therefore, $f$ is a bijection. □

The proof of Proposition 5.7 may be easily adapted to various extensions of 2WA. For instance, instead of restricting to *simple* runs when computing the semantics of a 2WA $\mathcal{A}$, we may allow $k$-simple runs (for some fixed $k$) in which no configuration is visited more than $k$ times. Even when $k = 2$, we do not know an easy way to construct a 2WA $\mathcal{A}'$ whose semantics over 1-simple runs coincide with the semantics of $\mathcal{A}$ over 2-simple runs. But the proof of Proposition 5.7 allows to cope with this extension simply by allowing $k$-simple words when defining the set $Q'$ of states of $\mathcal{B}$. Hence, we can construct a 1WA $\mathcal{B}$ whose semantics coincides with the semantics of $\mathcal{A}$ over $k$-simple runs, showing that this extension does not add expressive power.

Instead of allowing more runs, one may also wish to restrict the semantics of $\mathcal{A}$ to fewer runs, e.g., those visiting the initial position $\triangleright$ no more than 3 times. To do so, the natural approach is to add some information to the states of $\mathcal{A}$, e.g., a counter recording how many times (limited to 3) the initial position has been visited. But simple runs of the resulting 2WA $\mathcal{A}'$ do not necessarily correspond to simple runs of $\mathcal{A}$. Indeed the configurations $((p, 1), 8)$ and $((p, 2), 8)$ of $\mathcal{A}'$ are distinct but if we project away the counter we get the same configuration $(p, 8)$ of $\mathcal{A}$. To obtained the desired semantics, one may introduce an equivalence relation $\sim$ on the states of $\mathcal{A}'$ (having the same state from $\mathcal{A}$) and restrict the semantics of $\mathcal{A}'$ to $\sim$-simple runs, i.e., runs which do not visit two $\sim$-equivalent configurations. Again, the proof of Proposition 5.7 allows to cope with the $\sim$-simple semantics of a 2WA, by restricting the set $Q'$ of states of $\mathcal{B}$ to $\sim$-simple words.

Finally, we introduce another extension which will be useful in the proof of Proposition 5.9 below. We generalize the acceptance condition of a 2WA $\mathcal{A} = (R, A, I, F, \Delta, \mathsf{weight})$ by letting $I, F$ be finite subsets of $R^+$. A run is declared accepting if the sequence of states visiting the initial position $\triangleright$ is in $I$ and similarly the sequence of states visiting the final position $\triangleleft$ is in $F$. Notice that this is indeed a generalization as every 2WA $\mathcal{A} = (R, A, I, F, \Delta, \mathsf{weight})$ can be encoded into this new formalism by replacing $I$ (respectively, $F$) with the (finite) set of all simple sequences of states starting with a state of $I$ (respectively, ending with a state of $F$). Again, the proof of Proposition 5.7 allows to cope with this extension easily by changing $I'$ to the sequences in $Q'_\triangleright$ whose projections on $R$ are in $I \subseteq R^+$, and similarly for $F'$.

We extend now the translation of Proposition 5.7 to weighted automata with pebbles. The general idea is an induction on the number of pebbles, but it has to be done with great care so that all and only the *simple* runs of the P2WA are taken into account.

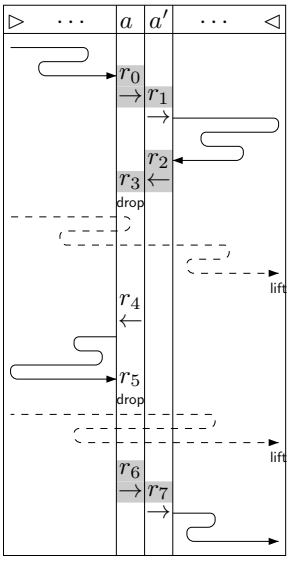


Fig. 7. Run of a 2WA and some crossing-sequences

PROPOSITION 5.9. *Let* $\mathbb{S}$ *be a* commutative *semiring,* $f \in \mathbb{S}\langle\langle A^+ \rangle\rangle$ *and* $p \geqslant 0$. *If* $f$ *is recognizable by a* P2WA *with* $p$ *pebbles, then* $f$ *is recognizable by a* P1WA *with* $p$ *pebbles.*

PROOF. We start with a P2WA $\mathcal{A} = (R, A, I, F, \Delta, \mathsf{weight})$ with $p$ pebbles, and a generalized acceptance condition: $I$ and $F$ are finite subsets of $R^+$. A run $\rho$ of $\mathcal{A}$ is accepting if the sequence of states visiting the initial position $\triangleright$ *while no pebbles are dropped* is in $I$, and the sequence of states visiting the final position $\triangleleft$ *while no pebbles are dropped* is in $F$. This generalization will be useful in the inductive step of the construction. We assume without loss of generality that $\mathsf{lift}$ transitions of $\mathcal{A}$ only occur at the end marker $\triangleleft$.

If $p = 0$, Proposition 5.7 permits to conclude. We now explain the construction for $p > 0$. A crossing-sequence may now also include $\mathsf{drop}$ moves, see Figure 7. The parts of a run having at least one pebble dropped (dashed in Figure 7) will be deferred to the inductive step. Hence, a crossing-sequence consists only of the states visiting some position when no pebbles are dropped. If the run is simple, then this sequence of states is also simple.

As in the proof of Proposition 5.7, we also include the current letter and the sequence of moves in a crossing-sequence. Hence, we let $Q'_{\triangleright}$ be the (finite) set of $R$-*simple* words in $\triangleright(R\rightarrow)^+$, $Q'_{\triangleleft}$ be the (finite) set of $R$-*simple* words in $\triangleleft(R\leftarrow)^*R$, and $Q'_A$ be the (finite) set of $R$-*simple* words in $A(R\{\leftarrow, \rightarrow, \mathsf{drop}\})^*R\rightarrow$. For instance, Figure 7 exhibits the crossing-sequence $q = ar_0{\rightarrow}r_3\mathsf{drop}\, r_4{\leftarrow}r_5\mathsf{drop}\, r_6{\rightarrow} \in Q'_A$ in which the states $r_0, r_3, r_4, r_5, r_6$ must be pairwise distinct.

For each state $q = a\tau \in Q'_A$, we construct a P2WA $\mathcal{A}_q = (R_q, \Gamma, I_q, F_q, \Delta_q, \mathsf{weight}_q)$ with $(p-1)$ pebbles which computes the sum of the weights of the tuples of runs filling the $\mathsf{drop}$-$\mathsf{lift}$ gaps in the crossing-sequence $q$, i.e., the dashed lines in Figure 7.

The input alphabet of $\mathcal{A}_q$ is $\Gamma = A \times \{0, 1\}$ where the second component encodes the position of the dropped pebble. Here, we are only interested in words with a unique position carrying a 1 in this second component of $A \times \{0, 1\}$: if $i$ is this unique position, we denote as $(u, i)$ such a word. Recall that $\widetilde{\Gamma} = \Gamma \cup \{\triangleright, \triangleleft\}$.

From a crossing-sequence $q = a\tau \in Q'_A$, we first extract the sequence $\sigma(\tau)$ of pairs of states surrounding $\mathsf{drop}$ moves. For instance, $\sigma(r_0{\rightarrow}r_3\mathsf{drop}\, r_4{\leftarrow}r_5\mathsf{drop}\, r_6{\rightarrow}) =$

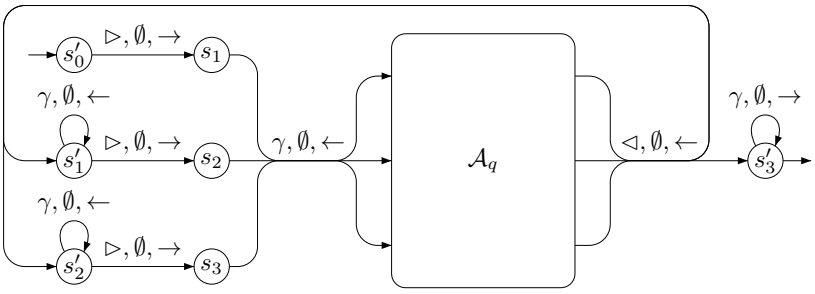


Fig. 8. The P2WA $\mathcal{A}_q$ filling the drop-lift gaps of the crossing-sequence $q$

$(r_3, r_4)(r_5, r_6)$ and $\sigma(r_1 \mathsf{drop}\, r_2 \mathsf{drop}\, r_3 \rightarrow) = (r_1, r_2)(r_2, r_3)$. We assume below that $\sigma(\tau) = (r_1, r'_1)(r_2, r'_2) \cdots (r_N, r'_N)$ with $N > 0$. Note that $N \leqslant |R|$ since $r_1, r_2, \ldots, r_N$ must be pairwise distinct.

We let $R_q = R \uplus \{s'_0, s_1, s'_1, \ldots, s_N, s'_N\}$ be the set of states of $\mathcal{A}_q$. The automaton $\mathcal{A}_q$ should fill the drop-lift gaps between $(r_j, r'_j)$ for all $1 \leqslant j \leqslant N$. To do so, it simulates $\mathcal{A}$ interpreting pebble $p$ as the $\{0,1\}$ component of $\Gamma$ (recall that pebble $p$ is the first which is dropped by $\mathcal{A}$). Hence, for $(r, \gamma, P, d, r') \in \Delta_q$ with $r, r' \in R$, $\gamma \in \widetilde{\Gamma}$, $P \subseteq \{1, \ldots, p-1\}$ and $d \in D$ we let

$$\begin{aligned}
\mathsf{weight}_q(r, (c,1), P, d, r') &= \mathsf{weight}(r, c, P \cup \{p\}, d, r') && \text{if } \gamma = (c,1)\\
\mathsf{weight}_q(r, (c,0), P, d, r') &= \mathsf{weight}(r, c, P, d, r') && \text{if } \gamma = (c,0)\\
\mathsf{weight}_q(r, \gamma, P, d, r') &= \mathsf{weight}(r, \gamma, P, d, r') && \text{if } \gamma \in \{\triangleright, \triangleleft\} \text{ (and } P = \emptyset)\,.
\end{aligned}$$

The additional states in $\{s'_0, s_1, s'_1, \ldots, s_N, s'_N\}$ and transitions will make sure that all gaps $(r_j, r'_j)$ are filled, see Figure 8. States $s'_{j-1}$ are used to reset the head to the initial position $\triangleright$ between two simulations, and we move from $s'_{j-1}$ to $s_j$ at the beginning of the word:

$$\begin{aligned}
\mathsf{weight}_q(s'_{j-1}, \gamma, \emptyset, \leftarrow, s'_{j-1}) &= 1 && \text{for } 1 < j \leqslant N \text{ and } \gamma \in \Gamma\\
\mathsf{weight}_q(s'_{j-1}, \triangleright, \emptyset, \rightarrow, s_j) &= 1 && \text{for } 1 \leqslant j \leqslant N\,.
\end{aligned}$$

A drop transition of $\mathcal{A}$ from state $r_j$ reading letter $a$ (recall $q = a\tau$) is simulated in $\mathcal{A}_q$ by a $\leftarrow$ transition from $s_j$ reading the first letter $\gamma$ of the word (note that when $\mathcal{A}_q$ is in state $s_j$ it must be on the first position of the word due to the $\rightarrow$ transition from $s'_{j-1}$ to $s_j$). A lift transition of $\mathcal{A}$ reaching $r'_j$ is simulated in $\mathcal{A}_q$ by a $\leftarrow$ transition to $s'_j$:

$$\begin{aligned}
\mathsf{weight}_q(s_j, \gamma, \emptyset, \leftarrow, r) &= \mathsf{weight}(r_j, a, \emptyset, \mathsf{drop}, r) && \text{for } 1 \leqslant j \leqslant N,\ r \in R \text{ and } \gamma \in \Gamma\\
\mathsf{weight}_q(r', \triangleleft, \emptyset, \leftarrow, s'_j) &= \mathsf{weight}(r', \triangleleft, \emptyset, \mathsf{lift}, r'_j) && \text{for } 1 \leqslant j \leqslant N \text{ and } r' \in R\,.
\end{aligned}$$

Finally, when $\mathcal{A}_q$ reaches $s'_N$, it terminates with a $\rightarrow$ move:

$$\mathsf{weight}_q(s'_N, \gamma, \emptyset, \rightarrow, s'_N) = 1 \qquad \text{for } \gamma \in \Gamma\,.$$

To define the acceptance condition of $\mathcal{A}_q$, we let $I_q$ be the (finite) set of *simple* sequences in $s'_0 R^* s'_1 R^* \cdots s'_{N-1} R^*$ and $F_q$ be the (finite) set of *simple* sequences in $R^* s'_N$. Considering this acceptance condition, an accepting run $\hat{\rho}$ of $\mathcal{A}_q$ — over a word of $\Gamma^+$ with a unique position carrying a 1 in its second component — can be split into a sequence of runs $\hat{\rho} = \rho'_0 \hat{\rho}_1 \rho'_1 \cdots \hat{\rho}_N \rho'_N$ where for each $1 \leqslant j \leqslant N$

— the run $\hat{\rho}_j$ fills the drop-lift gap $(r_j, r'_j)$. It goes from $s_j$ to $s'_j$ and simulates a run $\rho_j$ of $\mathcal{A}$ which starts with a drop transition $(r_j, a, \emptyset, \mathsf{drop}, r)$ and ends with a lift transition

$(r', \lhd, \emptyset, \mathsf{lift}, r'_j)$ without lifting pebble $p$ in-between. Moreover, $\mathsf{weight}_q(\hat{\rho}_j) = \mathsf{weight}(\rho_j)$ for all $1 \leqslant j \leqslant N$.

— the run $\rho'_0$ consists of a single transition from $s'_0$ to $s_1$, the run $\rho'_j$ for $0 < j < N$ loops on state $s'_j$ to reset the head to the initial position $\rhd$ and then moves to state $s_j$, and the run $\rho'_N$ consists of a single transition looping on $s'_N$ to reach the final position $\lhd$. Moreover, $\mathsf{weight}_q(\rho'_j) = 1$ for all $0 \leqslant j \leqslant N$.

Note that, the run $\hat{\rho}$ is simple if and only if the sequence of runs $\rho_1, \dots, \rho_N$ is *globally* simple, i.e., a configuration may not occur twice in some $\rho_j$ and it may not occur both in $\rho_i$ and $\rho_j$ with $i \neq j$. This is important since the runs $\rho_1, \dots, \rho_N$ fill the $\mathsf{drop\text{-}lift}$ gaps of the crossing-sequence $a\tau$ and are therefore part of a single run of $\mathcal{A}$. This explains why we have a single copy of $\mathcal{A}$ in $\mathcal{A}_q$. If instead we used $N$ copies of $\mathcal{A}$ inside $\mathcal{A}_q$, one for each $\mathsf{drop\text{-}lift}$ gap $(r_j, r'_j)$ then the simplicity of $\hat{\rho}$ would not imply the *global* simplicity[5] of $\rho_1, \dots, \rho_N$.

Clearly, all accepting runs of $\mathcal{A}_q$ start from state $s'_0$ and end in state $s'_N$. But this basic initial and final condition is not sufficient to ensure that the $\mathsf{drop\text{-}lift}$ gaps are filled correctly. For instance, $\mathcal{A}_q$ could go directly from $s'_0$ to $s'_N$ without ever visiting $s'_1, \dots, s'_{N-1}$. It could also visit all the new states but not in the intended order, which would not correspond to a sequence of runs of $\mathcal{A}$ filling the gaps $(r_1, r'_1) \cdots (r_N, r'_N)$. This is why we use the generalized acceptance condition for $I_q$.

Towards a uniform construction below, we also define an automaton $\mathcal{A}_q$ when the crossing-sequence $q$ has no drop ($N = 0$). In this case, $\mathcal{A}_q$ has a single state $s'_0$ with a loop moving right with weight 1 in order to simply scan the word from begin to end.

By induction, for each $q \in Q'_A$ we can construct a P1WA $\mathcal{B}_q$ with $(p-1)$ pebbles which is equivalent to $\mathcal{A}_q$, i.e., such that for each word $u \in \Gamma^+$ there is a weight preserving bijection between the simple accepting runs of $\mathcal{A}_q$ over $u$ and the accepting runs of $\mathcal{B}_q$ over $u$. We let $\mathcal{B}_q = (Q'_q, \Gamma, I'_q, F'_q, \Delta'_q, \mathsf{weight}'_q)$ and we assume without loss of generality that $I'_q = \{\mathsf{init}_q\}$ and $F'_q = \{\mathsf{final}_q\}$ are singletons. Note that, if the crossing-sequence $q$ has no drop then we may choose $\mathcal{B}_q = \mathcal{A}_q$.

We define now the P1WA $\mathcal{B} = (Q', A, I', F', \Delta', \mathsf{weight}')$ with $p$ pebbles associated with $\mathcal{A}$. Its set of states is

$$Q' = Q'_\rhd \uplus Q'_\lhd \uplus Q'_A \uplus \overline{Q'_A} \uplus \biguplus_{q \in Q'_A} Q'_q \,.$$

The set $I'$ of initial states of $\mathcal{B}$ is the set of crossing-sequences in $Q'_\rhd$ whose projections on $R$ are in $I \subseteq R^+$, and similarly, the set $F'$ of final states of $\mathcal{B}$ is the set of crossing-sequences in $Q'_\lhd$ whose projections on $R$ are in $F \subseteq R^+$.

The automaton $\mathcal{B}$ contains a copy of each $\mathcal{B}_q$ in order to simulate the parts of a run when the first pebble is dropped (dashed lines in Figure 7). In this copy, the second component of the alphabet $\Gamma = A \times \{0,1\}$ of $\mathcal{B}_q$ is re-interpreted as the encoding of the first pebble $p$: for $s, s' \in Q'_q$, $c \in A$ and $P \subseteq \{1, \dots, p-1\}$ we set

$$\begin{aligned}
\mathsf{weight}'(\mathsf{init}_q, \rhd, \emptyset, \to, s') &= \mathsf{weight}'_q(\mathsf{init}_q, \rhd, \emptyset, \to, s') \\
\mathsf{weight}'(s, c, P, \to, s') &= \mathsf{weight}'_q(s, (c,0), P, \to, s') \\
\mathsf{weight}'(s, c, P \cup \{p\}, \to, s') &= \mathsf{weight}'_q(s, (c,1), P, \to, s') \,.
\end{aligned}$$

From each state $q = a\tau \in Q'_A$, we first call (the copy of) the automaton $\mathcal{B}_q$ dropping the first pebble and when the computation of $\mathcal{B}_q$ is done we lift the pebble returning to the copy

[5] Alternatively, we could define an equivalence relation $\sim$ on states of $\mathcal{A}_q$ and use a $\sim$-simple semantics of runs, as originally done in [Bollig et al. 2010].

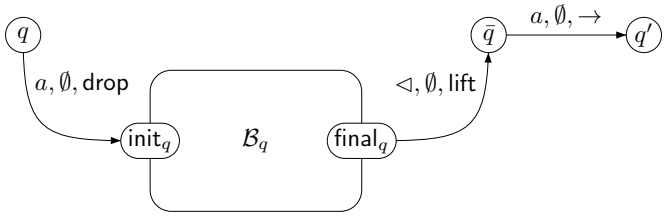


Fig. 9. Call to the P1WA $\mathcal{B}_q$ filling the drop-lift gaps of the crossing-sequence $q$

$\bar{q} \in \overline{Q'_A}$ of $q \in Q'_A$, see Figure 9. Then, we perform a $\to$ move to the next crossing-sequence $q' \in Q'_A \cup Q'_\lhd$. The drop and lift moves are with weight 1:

$$\mathsf{weight}'(q, a, \emptyset, \mathsf{drop}, \mathsf{init}_q) = \mathsf{weight}'(\mathsf{final}_q, \lhd, \emptyset, \mathsf{lift}, \bar{q}) = 1\,.$$

As in the proof of Proposition 5.7, two crossing-sequences $q = a\tau \in Q'_\rhd \cup Q'_A$ and $q' = a'\tau' \in Q'_A \cup Q'_\lhd$ match if $|\tau|_\to = 1 + |\tau'|_\leftarrow$. We assume $\mathsf{Trans}(q, q')$ to be defined as in Proposition 5.7, so that we may define the weight of $\to$ moves of $\mathcal{B}$ induced by the matching crossing-sequences $(q, q')$ as the product of the weights of the (left or right) transitions of $\mathcal{A}$ contained in $(q, q')$:

$$\mathsf{weight}'(q, \rhd, \emptyset, \to, q') = \prod_{\delta \in \mathsf{Trans}(q,q')} \mathsf{weight}(\delta) \qquad \text{if } q \in Q'_\rhd$$

$$\mathsf{weight}'(\bar{q}, a, \emptyset, \to, q') = \prod_{\delta \in \mathsf{Trans}(q,q')} \mathsf{weight}(\delta) \qquad \text{if } q = a\tau \in Q'_A\,.$$

Lemma 5.10 below proves that there is a weight preserving bijection between the simple accepting runs of $\mathcal{A}$ and the accepting runs of $\mathcal{B}$. We conclude that $[\![\mathcal{B}]\!] = [\![\mathcal{A}]\!]$ since for all $u \in A^+$, $[\![\mathcal{A}]\!](u)$ is the sum of the weights of the simple accepting runs of $\mathcal{A}$ over $u$, whereas $[\![\mathcal{B}]\!](u)$ is the sum of the weights of the accepting runs of $\mathcal{B}$ over $u$. □

Lemma 5.10. *For every word $u \in A^+$, there is a weight preserving bijection between the simple accepting runs of $\mathcal{A}$ over $u$ and the accepting runs of $\mathcal{B}$ over $u$.*

Proof. Fix a word $u = u_1 \cdots u_n \in A^+$ of length $n \geqslant 1$. We will decompose our bijection into several ones, all preserving the weights in an adequate manner.

To extend the proof of Lemma 5.8, we first associate with a simple accepting run $\rho$ of $\mathcal{A}$ over $u$ a tuple $f(\rho) = (q_0, q_1, g_1, \ldots, q_i, g_i, \ldots, q_n, g_n, q_{n+1})$ where $q_i = u_i\tau_i$ is the crossing-sequence at position $0 \leqslant i \leqslant n+1$ and for each position $1 \leqslant i \leqslant n$, the tuple $g_i = (\rho_1^i, \ldots, \rho_{N_i}^i)$ consists of the sequence of subruns of $\rho$ with pebble $p$ dropped on position $i$, together with the surrounding drop/lift transitions.

Formally, for $0 \leqslant i \leqslant n+1$, to define the crossing sequence $q_i = u_i\tau_i$, we let $\tau_i$ be the "projection" of the configurations visiting position $i$ with an empty stack of pebbles, where configuration $(r, \varepsilon, i)$ is "projected" to $r\to$ (respectively, $r\leftarrow$, $r$ drop or $r$) if it is followed by some $(r', \varepsilon, i+1)$ (respectively, $(r', \varepsilon, i-1)$, $(r', i, 0)$ or is the last configuration).

Moreover, for $1 \leqslant i \leqslant n$ with $N_i = |q_i|_{\mathsf{drop}}$, we extract from $\rho$ the tuple $g_i = (\rho_1^i, \ldots, \rho_{N_i}^i)$ of subruns of the form $(r_j, \varepsilon, i)(r_{j+1}, \pi_{j+1}, 0) \cdots (r_{k-1}, \pi_{k-1}, n+1)(r_k, \varepsilon, i)$ where $\pi_\ell = i \cdot \pi'_\ell$ for all $j < \ell < k$, i.e., where the first pebble is dropped on position $i$.

We first show that $f$ is a bijection from simple accepting runs of $\mathcal{A}$ over $u$ to the set, denoted $\mathrm{Seq}_u$, of sequences $(q_0, q_1, g_1, \ldots, q_i, g_i, \ldots, q_n, g_n, q_{n+1})$ verifying:

— $q_0 \in I'$, $q_{n+1} \in F'$ and $q_i \in Q'_A$ for all $1 \leqslant i \leqslant n$;
— for all $0 \leqslant i \leqslant n$, the pair $(q_i, q_{i+1})$ matches, i.e., $|q_i|_\to = 1 + |q_{i+1}|_\leftarrow$;

— for all $1 \leqslant i \leqslant n$, denoting $N_i = |q_i|_{\mathsf{drop}}$, $g_i$ is a tuple $(\rho^i_1, \ldots, \rho^i_{N_i})$ of subruns of $\mathcal{A}$, globally simple, each starting with the drop of pebble $p$ on position $i$, ending with the first time this pebble is lifted, and that *matches* with $\sigma(q_i)$ (i.e., the $j$th pair in $\sigma(q_i)$ contains exactly the pair of first and last states of the run $\rho^i_j$).

It is not difficult to check that $f(\rho) \in \mathrm{Seq}_u$ whenever $\rho$ is a simple accepting run of $\mathcal{A}$ over $u$. Defining the weight of a sequence $(q_0, q_1, g_1, \ldots, q_n, g_n, q_{n+1}) \in \mathrm{Seq}_u$ as

$$\mathsf{weight}'(q_0, \triangleright, \emptyset, \rightarrow, q_1) \times \prod_{i=1}^{n} \big(\mathsf{weight}(g_i) \times \mathsf{weight}'(\overline{q}_i, u_i, \emptyset, \rightarrow, q_{i+1})\big)$$

where $\mathsf{weight}(g_i) = \prod_{j=1}^{N_i} \mathsf{weight}(\rho^i_j)$ with $g_i = (\rho^i_1, \ldots, \rho^i_{N_i})$, we can easily show that $f$ preserves the weights.

We finally prove that $f$ is a bijection by constructing its inverse function. Let $(q_0, q_1, g_1, \ldots, q_n, g_n, q_{n+1}) \in \mathrm{Seq}_u$. Write $q_i = u_i \tau_i$ for $0 \leqslant i \leqslant n+1$. We recover a simple accepting run $\rho$ of $\mathcal{A}$ over $u$ as the output of Run'$(0, \tau_0, \tau_1, g_1, \ldots, \tau_n, g_n, \tau_{n+1})$ where the recursive function Run' is defined by:

**Function** Run'$(i, \tau_0, \tau_1, g_1, \ldots, \tau_n, g_n, \tau_{n+1})$

**Data**: A position $i$ of $u$ and a tuple $(\tau_0, \tau_1, g_1, \ldots, \tau_n, g_n, \tau_{n+1})$
**Result**: A run $\rho$ of $\mathcal{A}$

```
match τ_i with
    r→ τ'_i :
        output (r, ε, i);
        Run'(i + 1, τ_0, ..., τ'_i, g_i, ..., τ_{n+1});
    r← τ'_i :
        output (r, ε, i);
        Run'(i − 1, τ_0, ..., τ'_i, g_i, ..., τ_{n+1});
    r drop τ'_i :
        match g_i with
            (ρ', g'_i) :
                output ρ';
                Run'(i, τ_0, ..., τ'_i, g'_i, ..., τ_{n+1});
    r : (* necessarily i = n + 1 *)
        output (r, ε, n + 1);
```

We then consider each tuple $g_i$ of runs of $\mathcal{A}$ with pebble $p$ dropped. It has already been explained in the previous proof how to construct a weight preserving bijection between a tuple $g_i = (\rho^i_1, \ldots, \rho^i_N)$, which matches with a sequence of pairs $\sigma(q_i)$, and an accepting run $\hat{\rho}_i = \rho'_0 \hat{\rho}^i_1 \rho'_1 \cdots \hat{\rho}^i_N \rho'_N$ of $\mathcal{A}_{q_i}$ over $(u, i)$ so that $g_i$ is globally simple if and only if $\hat{\rho}_i$ is simple: it consists of filling some gaps (with the runs $\rho'_j$), and replacing alphabet $A$ by $\Gamma = A \times \{0, 1\}$ (going from runs $\rho^i_j$ to $\hat{\rho}^i_j$).

Using the induction hypothesis, we know that there is a weight preserving bijection between simple accepting runs $\hat{\rho}_i$ of $\mathcal{A}_{q_i}$ over $(u, i)$ and accepting runs $\overline{\rho}_i$ of $\mathcal{B}_{q_i}$ over $(u, i)$.

It remains to change back the alphabet to $A$, to build a weight preserving bijection between accepting runs $\overline{\rho}_i$ of $\mathcal{B}_{q_i}$ over $(u, i)$ to runs $\overline{g}_i$ of $\mathcal{B}$ over $u$ starting in state $\mathsf{init}_{q_i}$ in position 0 and finishing in state $\mathsf{final}_{q_i}$ in position $n+1$. Finally, coming back to the sequences of $\mathrm{Seq}_u$, these intermediary bijections permit to build a weight preserving bijection from sequences $(q_0, q_1, g_1, \ldots, q_i, g_i, \ldots, q_n, g_n, q_{n+1}) \in \mathrm{Seq}_u$ to accepting runs

$(q_0,\varepsilon,0)(q_1,\varepsilon,1)\overline{g}_1(\overline{q}_1,\varepsilon,1)\cdots(q_n,\varepsilon,n)\overline{g}_n(\overline{q}_n,\varepsilon,n)(q_{n+1},\varepsilon,n+1)$ of $\mathcal{B}$ over $u$, which concludes the proof of correctness. □

### 5.4. Pebble weighted automata and weighted logic

The following theorem summarizes the main results of this section.

THEOREM 5.11. *Let $\mathbb{S}$ be a commutative semiring and $f \in \mathbb{S}\langle\langle A^+\rangle\rangle$. The following assertions are equivalent over $\mathbb{S}$ and $A$:*

*(1) $f$ is* wFOTC$^{\mathrm{b}}$(FO)*-definable;*
*(2) $f$ is* wFOTC$^{\mathrm{b},<}$(FO)*-definable;*
*(3) $f$ is recognizable by a* P2WA*;*
*(4) $f$ is recognizable by a* P1WA*.*

Inclusions 2 ⇒ 1 and 4 ⇒ 3 follow from the definitions of weighted logics and pebble weighted automata. Proposition 5.13 will reveal that 1 ⇒ 3. Proposition 5.9 will prove that 3 ⇒ 4: notice that this is the only implication using the commutativity of the semiring. The theorem follows from Proposition 5.12, which finally proves implication 4 ⇒ 2.

PROPOSITION 5.12. *Every series recognizable by a* P1WA *is* wFOTC$^{\mathrm{b},<}$(FO)*-definable.*

PROOF. Starting from a P1WA $\mathcal{A}$ with $p$ pebbles, we build a wFOTC$^{\mathrm{b},<}$(FO)-formula by induction on $p$. Without loss of generality, we may assume that lift transitions only take place at the end marker $\lhd$. The case $p = 0$ follows from Theorem 4.1 since a P1WA with 0 pebble can be translated easily into a weighted automaton by removing the first transitions reading $\rhd$, and *modulo* is definable in wFOTC$^{\mathrm{b},<}$(FO) (see Example 3.13). Note that the tape contains $\rhd u \lhd$ and $\widetilde{\mathrm{pos}}(u) = \{0,\dots,|u|+1\}$ but we still have $\mathrm{pos}(u) = \{1,\dots,|u|\}$, $\mathsf{first} = 1$ and $\mathsf{last} = |u|$.

The induction step is shown like in the proof of the implication 1 ⇒ 2 of Theorem 4.1. Notice that we no longer suppose that there exists only one initial state $q_0 = 1$. The main difference is however in the definition of the formulae $\Xi^v_{q,q'}(x)$, which now uses the induction hypothesis. When dropping the *first* pebble on some position $z$, the head is reset to the initial position labeled $\rhd$, then the automaton behaves as a P1WA with $(p-1)$ pebbles, starting in some state $q$, reading the word with position $z$ "marked", until it reaches the final position labeled $\lhd$ in some states $q'$, where it will have to lift the first pebble. By induction, there is a wFOTC$^{\mathrm{b},<}$(FO)-formula $\Theta_{q,q'}(z)$ computing the sum of the weights of all these runs of $\mathcal{A}$.

For $\ell \geqslant 1$, $v = v_1 \cdots v_\ell \in A^+$ and $q_0, q_\ell \in Q$, we define

$$\Xi^v_{q_0,q_\ell}(x) \stackrel{\text{def}}{=} \bigoplus_{q_1,\dots,q_{\ell-1}\in Q} \bigotimes_{1\leqslant k\leqslant \ell} P_{v_k}(x-q_0+k) \otimes \nu_{q_k,v_{k+1},q_{k+1}}(x-q_0+k)$$

with

$$\nu_{q,a,q'}(z) \stackrel{\text{def}}{=} \mathsf{weight}(q,a,\emptyset,\to,q') \oplus \bigoplus_{r,r',r''\in Q} \big(\mathsf{weight}(q,a,\emptyset,\mathsf{drop},r) \otimes \Theta_{r,r'}(z) \otimes \mathsf{weight}(r',\lhd,\emptyset,\mathsf{lift},r'') \otimes \mathsf{weight}(r'',a,\emptyset,\to,q')\big)\,.$$

We have to guess non-deterministically the states $q_1,\dots,q_{\ell-1}$ that the automaton will visit between positions $x$ and $y$ while pebble $p$ is not dropped, because each of them is useful to compute the runs with pebble $p$ dropped. We obtain for $u \in A^+$ and positions $i, i+\ell-1 \in \mathrm{pos}(u)$

$$[\![\Xi^{u[i..i+\ell-1]}_{q_0,q_\ell}]\!](u,i+q_0-1) = \sum_{q_1,\dots,q_{\ell-1}\in Q} \prod_{0\leqslant k\leqslant \ell-1} [\![\nu_{q_{k-1},u_{i+k},q_k}(z)]\!](u,i+k)\,.$$

Then, we use almost the same weighted formulae as in Theorem 4.1. For small words $u \in A^+$ with $|u| \leqslant 2m$, we use

$$\Phi_{\text{small}} \stackrel{\text{def}}{=} (x = \mathsf{first} \wedge y = \mathsf{last}) \otimes \bigoplus_{1 \leqslant d \leqslant 2m,\, v \in A^d} \Big[ (\mathsf{last} = d) \wedge \bigwedge_{1 \leqslant k \leqslant d} P_{v_k}(k) \Big] \otimes [\![\mathcal{A}]\!](v)$$

and we obtain $[\![\Phi_{\text{small}}]\!](u) = [\![\mathcal{A}]\!](u)$. Finally, we define

$$\Phi_1(x,y) = \bigoplus_{\substack{q_0 \in I,\, q, q' \in Q \\ v \in A^m}} \big(\mathsf{first} = x \wedge 2m < \mathsf{last} \wedge y = m + q'\big) \otimes \mathsf{weight}(q_0, \triangleright, \emptyset, \rightarrow, q) \otimes \Xi^v_{q,q'}(q)$$

$$\Phi_2(x,y) = \bigoplus_{q,q' \in Q,\, v \in A^m} \big(\mathsf{first} < x \wedge x - q + 2m < \mathsf{last} \wedge x \equiv_m q \wedge y - q' = x - q + m\big) \otimes \Xi^v_{q,q'}(x)$$

$$\Phi_3(x,y) = \bigoplus_{\substack{q \in Q,\, q' \in F \\ m < d \leqslant 2m,\, v \in A^d}} \big(\mathsf{first} < x \wedge x - q + d = \mathsf{last} \wedge x \equiv_m q \wedge y = \mathsf{last}\big) \otimes \Xi^v_{q,q'}(x)$$

and we can show, as in the proof of Theorem 4.1, that $[\mathrm{TC}^{2m,<}_{x,y}(\Phi_{\text{small}} \oplus \Phi_1 \oplus \Phi_2 \oplus \Phi_3)](\mathsf{first}, \mathsf{last})$ computes $[\![\mathcal{A}]\!]$, as required.

We remark that this construction yields a $\mathrm{wFOTC}^{\mathrm{b},<}(\mathrm{FO})$-expression using $p+2$ nested $\mathrm{TC}^{N,<}$-operators: $p+1$ nested $\mathrm{TC}^{N,<}$-operators come from the induction in the proof of Theorem 4.1, and the last one comes from the definition of the modulo operation. □

Proposition 5.13. *Let $\mathbb{S}$ be a (non necessarily commutative) semiring and $f \in \mathbb{S}\langle\langle A^+ \rangle\rangle$. If $f$ is $\mathrm{wFOTC}^{\mathrm{b}}(\mathrm{FO})$-definable, then $f$ is recognizable by a* P2WA.

Proof. By Lemma 3.3, we know that every series which is $\mathrm{wFOTC}^{\mathrm{b}}(\mathrm{FO})$-definable is in fact $\mathrm{wFOTC}^{\mathrm{b}}(\mathrm{AP})$-definable. Hence, for the unweighted part of $\mathrm{wFOTC}^{\mathrm{b}}(\mathrm{FO})$, we just have to prove that atomic formulae $P_a(x), \neg P_a(x), x \leqslant y, \neg(x \leqslant y)$ are recognized by some P2WA with 0 pebble. For $P_a(x)$ (respectively, $\neg P_a(x)$), the automaton simply scans the input word, searching for letter $(a,1)$ (respectively, $(b,1)$ with $b \neq a$), as the free variable $x$ is encoded in the alphabet $A \times \{0,1\}$. For the atomic formula $\neg(x \leqslant y)$, the automaton scans the input word, looking for the encoding of $y$ before that of $x$. One verifies the formula $x \leqslant y$ in the same way.

It remains to prove that the class P2WA is closed under the weighted constructs of $\mathrm{wFOTC}^{\mathrm{b}}(\mathrm{FO})$. First, notice that the constant series $s \in \mathbb{S}$ is easily recognized with a P2WA with 0 pebble. Note that if a series is recognizable by a P2WA with $p$ pebbles, it is also recognizable by a P2WA with $p+1$ pebbles. Therefore, given two P2WAs, one can assume that they use the same number of pebbles. Let $\mathcal{A}_1, \mathcal{A}_2$ be two P2WAs with $p$ pebbles over $A$. Closure under $\oplus$ is obtained using the disjoint union of the automata $\mathcal{A}_1$ and $\mathcal{A}_2$.

The P2WA with $p$ pebbles that recognizes the Hadamard product which maps every word $u \in A^+$ to $[\![\mathcal{A}_1]\!](u) \times [\![\mathcal{A}_2]\!](u)$ consists of three phases: first, it simulates the automaton $\mathcal{A}_1$ until it reaches the last position in a final state of $\mathcal{A}_1$; then, it comes back to the beginning of the word with transitions of weight 1; finally, it simulates the automaton $\mathcal{A}_2$ and exit in a final state of $\mathcal{A}_2$.

For first-order quantifiers, we use an extra pebble. For a P2WA $\mathcal{A}$ with $p$ pebbles over alphabet $A \times \{0,1\}$ computing $[\![\Phi(x)]\!]$, we construct two P2WA with $(p+1)$ pebbles over alphabet $A$ computing respectively $[\![\bigoplus_x \Phi]\!]$ and $[\![\bigotimes_x \Phi]\!]$.

The automaton for $\bigoplus_x \Phi$ nondeterministically chooses with transitions of weight 1 a position in the word, drops the extra pebble $p+1$, and simulates automaton $\mathcal{A}$, interpreting pebble $p+1$ as the extra $\{0,1\}$ component encoding variable $x$. After this simulation, it lifts the pebble and terminates the computation, going to the endmarker $\lhd$ with weights 1.

For $\bigotimes_x \Phi$, with transitions of weight 1, the new automaton drops successively the extra pebble $p+1$ on positions $1, \dots, |u|$ of the input word $u \in A^+$. Whenever it drops pebble $p+1$, it simulates $\mathcal{A}$ interpreting this pebble as the encoding of variable $x$ until it reaches the end marker $\lhd$ where it lifts pebble $p+1$.

Finally, assume that we have a P2WA $\mathcal{A}$ with $p$ pebbles over the alphabet $A \times \{0,1\}^2$ which evaluates some wFOTC$^{\mathrm{b}}$(FO) formula $\Phi(x,y)$. We construct a P2WA $\mathcal{A}'$ with $(p+2)$ pebbles over $A \times \{0,1\}^2$ in order to evaluate $[\mathrm{TC}^N_{x,y}\Phi](x',y')$ ($x'$ and $y'$ are now the two free variables). With transitions of weight 1, $\mathcal{A}'$ moves to the position $i'$ to which maps variable $x'$. If $y'$ also maps to $i'$, it simply simulates $\mathcal{A}$ interpreting $x' = y'$ as $x = y$, resulting in the evaluation of $\Phi(i',i')$. Otherwise, it drops its new pebble $p+2$. Whenever $\mathcal{A}'$ drops pebble $p+2$ on some position $i$, it enters some special state $q_{\mathsf{drop}}$ and with weights 1, it moves to some guessed position $j \neq i$ such that $|j-i| \leqslant N$ and drops pebble $p+1$. Whenever $\mathcal{A}'$ drops pebble $p+1$, it then simulates $\mathcal{A}$ interpreting positions $i$ and $j$ on which pebbles $p+2$ and $p+1$ are dropped as $x$ and $y$, resulting in the evaluation of $\Phi(i,j)$. When the simulation of $\mathcal{A}$ is completed, $\mathcal{A}'$ lifts pebble $p+1$ with weight 1 in order to return to position $j$. Still with weights 1, $\mathcal{A}'$ moves to position $i$ where pebble $p+2$ was dropped remembering $j-i$ and lifts pebble $p+2$. It then returns to position $j$. If $y' = j$ is the current position, the transitive closure is completed. With weights 1, we move to the end marker $\lhd$ and accept. Otherwise, $\mathcal{A}'$ drops pebble $p+2$ again in order to continue the evaluation of the transitive closure.

Note that, the simplicity requirement on runs implies that pebble $p+2$ cannot be dropped twice on the same position. Indeed, after dropping pebble $p+2$, $\mathcal{A}'$ always enters the same state $q_{\mathsf{drop}}$. Moreover, pebble $p+2$ is never dropped on position $y'$. On a successful run $\rho$ of $\mathcal{A}'$, either pebble $p+2$ is never dropped, and then we have $x' = i' = y'$ and we compute $\Phi(i',i')$; otherwise, pebble $p+2$ is successively dropped on some positions $x' = i_0, i_1, \dots, i_{\ell-1}$ such that $\ell \geqslant 1$ and $x' = i_0, i_1, \dots, i_{\ell-1}, i_\ell = y'$ are pairwise distinct and $|i_j - i_{j-1}| \leqslant N$ for $0 \leqslant j < \ell$. The sum of the weights of the successful runs of $\mathcal{A}'$ visiting such a pairwise distinct sequence $x' = i_0, \dots, i_\ell = y'$ is precisely $[\![\Phi]\!](u, i_0, i_1) \times \cdots \times [\![\Phi]\!](u, i_{\ell-1}, i_\ell)$. We deduce that $\mathcal{A}'$ computes the $N$-bounded transitive closure $[\mathrm{TC}^N_{x,y}\Phi](x',y')$. □

## 6. ALGORITHMIC ISSUES

### 6.1. Evaluation

An important problem is the evaluation of an automaton wrt. a given word. More precisely, given a word $u = u_1 \cdots u_n$ and a P1WA $\mathcal{A} = (Q, A, I, F, \Delta, \mathsf{weight})$, we are interested in computing $[\![\mathcal{A}]\!](u)$. In the sequel, we assume as before that $\mathsf{lift}$ transitions in $\mathcal{A}$ only occur at the end marker $\lhd$.

We fix the word $u = u_1 \cdots u_n \in A^+$ and inductively compute (using dynamic programming) a matrix $M(\pi,i) \in \mathbb{S}^{Q \times Q}$ for $0 \leqslant i \leqslant n+1$, and $\pi \in \mathrm{pos}(u)^{\leqslant p}$: coefficient $M(\pi,i)_{q,q'}$ is the sum of weights of runs that go from position 0 in state $q$ to position $i$ in state $q'$ with pebbles $p, \dots, p-|\pi|+1$ permanently dropped on positions described by $\pi$. Let $\mu(a,P,d) \in \mathbb{S}^{Q \times Q}$ be the matrix defined by $\mu(a,P,d)_{q,q'} = \mathsf{weight}(q,a,P,d,q')$ if $(q,a,P,d,q') \in \Delta$, and $\mu(a,P,d)_{q,q'} = 0$ otherwise. We denote by $Id \in \{0,1\}^{Q \times Q}$ the identity matrix and we initially set $M(\pi,0)_{q,q'} = Id$. For $1 \leqslant i \leqslant n+1$, we set

$$M(\pi,i) = \begin{cases} M(\pi,i-1) \times \mu(u_{i-1}, \mathrm{peb}(\pi,i-1), \to) \times \\ \quad \big(Id + \mu(u_i, \mathrm{peb}(\pi,i), \mathsf{drop}) \times M(\pi i, n+1) \times \mu(\lhd, \emptyset, \mathsf{lift})\big) & \text{if } i \leqslant n \wedge |\pi| < p \\ M(\pi,i-1) \times \mu(u_{i-1}, \mathrm{peb}(\pi,i-1), \to) & \text{otherwise.} \end{cases}$$

Indeed, going from position 0 to position $i$ can be decomposed as first going from position 0 to position $i-1$, then performing a right move and finally optionally (we use the identity matrix to simplify the matrix computation in this formula) dropping a pebble. Then, we

obtain

$$[\![\mathcal{A}]\!](u) = \lambda \times M(\varepsilon, n+1) \times \gamma$$

where $\lambda \in \{0,1\}^{1\times Q}$ is the line vector coding the initial states ($\lambda_q = 1$ if and only if $q \in I$) and $\gamma \in \{0,1\}^{Q\times 1}$ is the column vector coding the final states ($\gamma_q = 1$ if and only if $q \in F$). Using dynamic programming, this can be computed in time $\mathcal{O}(|Q|^3 \cdot |u|^{p+1})$. Here, $\mathcal{O}(|Q|^3)$ is due to matrix multiplication, and $\mathcal{O}(|u|^{p+1})$ is the number of matrices $M(\pi, i)$ that have to be determined. We obtain:

THEOREM 6.1. *Evaluation of a* P1WA *with $p$ pebbles and set of states $Q$ wrt. a word $u \in A^+$ is $\mathcal{O}(|Q|^3 \cdot |u|^{p+1})$.*

This evaluation procedure can be improved with some efforts as presented in [Gastin and Monmege 2012]. In particular, lifted to our framework, this permits to evaluate a P1WA with $\mathcal{O}(|Q|^3 \cdot |u|^{\max(p,1)})$ scalar operations (sum and product of elements of the semiring).

### 6.2. Satisfiability/emptiness

In the context of formal power series, one may call a closed expression $\Phi \in \mathrm{wFOTC}^{\mathrm{b},<}(\mathrm{FO})$ *satisfiable* if there is a word $w \in A^+$ such that $[\![\Phi]\!](w) \neq 0$. Over commutative positive semirings (i.e., zero-sum free and without divisors of zero), satisfiability is decidable due to Theorem 5.11, which reduces the problem to non-emptiness of the support of a formal power series recognized by a P2WA. The latter problem, in turn, can be reduced to the decidable emptiness problem for classical pebble automata over the Boolean semiring, as stated, e.g., in [Bojańczyk 2008].

However, this problem turns undecidable over general (not necessarily positive) semiring. In fact, we prove undecidability of emptiness for P1WA with 2 pebbles.

THEOREM 6.2. *Emptiness is undecidable over the semiring $\mathbb{Z}$ for* P1WA *with 2 pebbles, and formulae of* wFO(AP).

PROOF. A complete *2-counter machine* $\mathcal{M}$ is a tuple $(Loc, \Delta, \mathrm{init}, F)$ with $Loc$ a finite set of locations, $\Delta = Loc \times Instr \times Loc$ the set of transitions where $Instr = \{Inc^p, Dec^p, ZTest^p \mid p \in \{1,2\}\}$, $\mathrm{init} \in Loc$ the initial location and $F \subseteq Loc$ the set of final locations. If $w = (\ell_0, D_1, \ell_1)\cdots(\ell_{n-1}, D_n, \ell_n) \in \Delta^*$ and $D \in Instr$, we set $|w|_D = |\{k \in [1,n] \mid D_k = D\}|$, and $w_j = (\ell_0, D_1, \ell_1)\cdots(\ell_{j-1}, D_j, \ell_j)$ for $1 \leqslant j \leqslant n$. We define the value of counter $p \in \{1,2\}$ after step $j \in \{1,\dots,n\}$ as $c^p_j(w) = |w_j|_{Inc^p} - |w_j|_{Dec^p}$, and we also set $c^p_0 = 0$.

An *accepting run* of $\mathcal{M}$ is a word $w = (\ell_0, D_1, \ell_1)\cdots(\ell_{n-1}, D_n, \ell_n) \in \Delta^+$ such that:

(1) $\ell_0 = \mathrm{init}$ and $\ell_n \in F$,
(2) $c^p_j(w) \geqslant 0$ for all $j \in \{1,\dots,n\}$ and $p \in \{1,2\}$,
(3) if $D_j = ZTest^p$, then $c^p_{j-1}(w) = 0$ for all $j \in \{1,\dots,n\}$ and $p \in \{1,2\}$.

The emptiness problem for 2-counter machines consists in deciding whether a given 2-counter machine has an accepting run. It is well known to be undecidable [Minsky 1967]. This problem reduces to emptiness of 1-way pebble word automata with 2 pebbles over $(\mathbb{Z}, +, \times, 0, 1)$. From a 2-counter machine $\mathcal{M}$, we build such an automaton $\mathcal{A}$ assigning a nonzero weight to accepting runs of $\mathcal{M}$, and weight 0 to all other words. Hence, $\mathcal{A}$ has a nonzero semantics if and only if $\mathcal{M}$ has an accepting run.

A word $w \in \Delta^+$ is not an accepting run of $\mathcal{M}$ if and only if it does not consists in consecutive transitions, or it violates either (1), (2) or (3). Let $\mathcal{D}^p = \{j \leqslant |w| \mid D_j = Dec^p\}$ and $\mathcal{Z}^p = \{j \leqslant |w| \mid D_j = ZTest^p\}$. Then $w$ violates (2) if and only if for some $p \in \{1,2\}$,

there exists $j \in \mathcal{D}^p$ such that $c^p_{j-1}(w) = 0$, i.e.,

$$\prod_{j \in \mathcal{D}^p} c^p_{j-1}(w) = 0\,. \tag{8}$$

We can compute this product with a P1WA with 1 pebble over $\mathbb{Z}$ as explained in Example 5.5. Next, if (2) holds, then $w$ violates (3) if and only if for some $p \in \{1, 2\}$, there exists $j \in \mathcal{Z}^p$ such that $c^p_{j-1}(w) > 0$, that is, if and only if

$$\prod_{j \in \mathcal{Z}^p} \prod_{1 \leqslant k < j} (c^p_{j-1}(w) - k) = 0\,. \tag{9}$$

We have seen in Example 5.5 how to compute this value with a 2-pebble P1WA over $\mathbb{Z}$.

Henceforth, on every position $j \in \mathcal{D}^1 \cup \mathcal{D}^2 \cup \mathcal{Z}^1 \cup \mathcal{Z}^2$, automaton $\mathcal{A}$ drops a pebble in order to compute the products (8) or (9) as explained in Example 5.5. More precisely,

— on positions $j \in \mathcal{D}^1 \cup \mathcal{D}^2$, a drop transition leads to a copy of $\mathcal{A}_{\text{non-neg}}$, highlighted in Figure 4.
— on positions $j \in \mathcal{Z}^1 \cup \mathcal{Z}^2$, a drop transition leads to a copy of the automaton $\mathcal{A}_{\text{zero-test}}$ highlighted in Figure 5.
— a loop is used to do nothing on every position $j \notin \mathcal{D}^1 \cup \mathcal{D}^2 \cup \mathcal{Z}^1 \cup \mathcal{Z}^2$.
— finally, transitions with weight 0 are used either on the loop- or the drop-transitions to check if (1) is violated or if two consecutive letters of $\Delta$ are not consecutive transitions in $\mathcal{M}$.

Using Example 3.6, it is easy to construct a formula of wFO(AP) equivalent to automaton $\mathcal{A}$, showing that emptiness is also undecidable for wFO(AP). □

The problem is still open for automata with a single pebble.

## 7. CONCLUSION

In this paper, we introduced weighted automata with pebbles. We showed that these devices capture weighted first order logic extended with a transitive closure operator. Contrary to previous work, we do not restrict the first order logic but allow for arbitrary nesting of universal quantification. Moreover, we established several new characterizations of the classical recognizable formal power series in terms of logics with transitive closure.

These results are not only of theoretical interest. They also lay the basis for quantitative extensions of database query languages, such as XPath, and provide tracks to evaluate quantitative aspects of XML documents. The framework of weighted automata is natural for answering questions such as "How many nodes are selected by a request?", or "How difficult is it to answer a query?". The navigational mechanism of pebble automata is also well-suited in this context. So, our work is a first step before tackling similar questions on trees.

A second, and maybe related, line of future research concerns quantitative model checking, aiming at a general framework of quantitative specification languages that cover existing ones like [Laroussinie et al. 2012; Laroussinie et al. 2010; Fischer et al. 2010]. Note that this would also involve branching structures rather than words, as they naturally represent unfoldings of Kripke structures.

## ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments that permitted to simplify some proofs, and globally improve the quality of this paper.

## REFERENCES


Jean Berstel and Christophe Reutenauer. 2010. *Noncommutative Rational Series With Applications*. Cambridge University Press.

Manuel Blum and Carl Hewitt. 1967. Automata on a 2-Dimensional Tape. In *Proceedings of the 8th Annual Symposium on Switching and Automata Theory (SWAT'67)*.

Mikołaj Bojańczyk. 2008. Tree-Walking Automata. In *Language and Automata Theory and Applications (LATA'08) (Lecture Notes in Computer Science)*, Vol. 5196. Springer.

Mikołaj Bojańczyk, Mathias Samuelides, Thomas Schwentick, and Luc Segoufin. 2006. Expressive Power of Pebble Automata. In *Proceedings of the 33rd international conference on Automata, Languages and Programming - Volume Part I (ICALP'06) (Lecture Notes in Computer Science)*, Vol. 4051. Springer, 157–168.

Benedikt Bollig and Paul Gastin. 2009. Weighted versus Probabilistic Logics. In *Proceedings of the 13th International Conference on Developments in Language Theory (DLT'09) (Lecture Notes in Computer Science)*, Vol. 5583. Springer, 18–38.

Benedikt Bollig, Paul Gastin, Benjamin Monmege, and Marc Zeitoun. 2010. Pebble Weighted Automata and Transitive Closure Logics. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP'10) – Part II (Lecture Notes in Computer Science)*, Vol. 6199. Springer, 587–598.

J. Richard Büchi. 1959. *Weak Second-Order Arithmetic and Finite Automata*. Technical Report. University of Michigan.

Hubert Comon-Lundh, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. 2008. *Tree Automata Techniques and Applications*. http://tata.gforge.inria.fr/

Manfred Droste and Paul Gastin. 2009. Weighted Automata and Weighted Logics. In *Handbook of Weighted Automata*, Werner Kuich, Heiko Vogler, and Manfred Droste (Eds.). Springer, Chapter 5, 175–211.

Manfred Droste, Werner Kuich, and Heiko Vogler. 2009. *Handbook of Weighted Automata*. Springer.

Manfred Droste and Heiko Vogler. 2006. Weighted Tree Automata and Weighted Logics. *Theoretical Computer Science* 366, 3 (2006), 228–247.

Manfred Droste and Heiko Vogler. 2009. Weighted Logics for Unranked Tree Automata. *Theory of Computing Systems* (2009).

Calvin C. Elgot. 1961. Decision Problems of Finite Automata Design and Related Arithmetics. *Trans. Amer. Math. Soc.* 98 (1961), 21–52.

Joost Engelfriet and Hendrik Jan Hoogeboom. 1999. Tree-Walking Pebble Automata. *Jewels are forever, contributions to Theoretical* (1999), 72–83.

Joost Engelfriet and Hendrik Jan Hoogeboom. 2007. Automata with Nested Pebbles Capture First-Order Logic with Transitive Closure. *Logical Methods in Computer Science* 3 (2007), 1–27.

Kousha Etessami. 1997. Counting Quantifiers, Successor Relations, and Logarithmic Space. *J. Comput. System Sci.* 53, 3 (1997), 400–411.

Ina Fichtner. 2011. Weighted Picture Automata and Weighted Logics. *Theory of Computing Systems* 48, 1 (2011), 48–78.

Diana Fischer, Erich Grädel, and Lukasz Kaiser. 2010. Model Checking Games for the Quantitative $\mu$-Calculus. *Theory of Computing Systems* 47, 3 (2010), 696–719.

Paul Gastin and Benjamin Monmege. 2012. Adding Pebbles to Weighted Automata. In *Proceedings of the 17th International Conference on Implementation and Application of Automata (CIAA'12) (Lecture Notes in Computer Science)*, Vol. 7381. Springer, 28–51.

Hans Hansson and Bengt Jonsson. 1994. A Logic for Reasoning about Time and Reliability. *Formal Aspects of Computing* 6, 5 (1994), 512–535.

Lauri Hella, Leonid Libkin, Juha Nurmonen, and Wong Limsoon. 2001. Logics with Aggregate Operators. *J. ACM* 48, 4 (2001), 880–907.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *An Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Neil Immerman and Eric Lander. 1990. Describing Graphs: a First-Order Approach to Graph Canonization. In *Complexity Theory Retrospective*. Springer-Verlag, 59–81.

François Laroussinie, Antoine Meyer, and Eudes Petonnet. 2010. Counting LTL. In *Proceedings of the 17th International Symposium on Temporal Representation and Reasoning (TIME'10)*. IEEE Computer Society Press, 51–58.

François Laroussinie, Antoine Meyer, and Eudes Petonnet. 2012. Counting CTL. *Logical Methods in Computer Science* 9, 1 (2012).

Leonid Libkin. 2000. Logics with Counting and Local Properties. *ACM Transactions on Computational Logic* 1 (2000), 33–59.

Christian Mathissen. 2010. Weighted Logics for Nested Words and Algebraic Formal Power Series. *Logical Methods in Computer Science* 6, 1 (2010).

Ingmar Meinecke. 2006. Weighted Logics for Traces. In *Proceedings of the First International Computer Science Conference on Theory and Applications (CSR'06) (Lecture Notes in Computer Science)*, Vol. 3967. Springer-Verlag, 235–246.

Marvin L. Minsky. 1967. *Computation: Finite and Infinite Machines.* Prentice-Hall, Inc.

Frank Neven and Thomas Schwentick. 2003. On the Power of Tree-Walking Automata. *Information and Computation* 183, 1 (2003), 86–103.

Frank Neven, Thomas Schwentick, and Victor Vianu. 2004. Finite State Machines for Strings over Infinite Alphabets. *ACM Transactions on Computational Logic* 5 (2004), 403–435.

Michael O. Rabin. 1969. Decidability of Second-Order Theories and Automata on Infinite Trees. *Trans. Amer. Math. Soc.* 141 (1969), 1–35.

Michael O. Rabin and Dana Scott. 1959. Finite Automata and Their Decision Problems. *IBM Journal of Research and Development* 3, 2 (1959), 114–125.

Mathias Samuelides and Luc Segoufin. 2007. Complexity of Pebble Tree-Walking Automata. In *Proceedings of the 16th International Conference on Fundamentals of Computation Theory (FCT'07) (Lecture Notes in Computer Science)*, Vol. 4639. Springer, 458–469.

Marcel-Paul Schützenberger. 1961. On the Definition of a Family of Automata. *Information and Control* 4 (1961), 245–270.

John C. Shepherdson. 1959. The Reduction of Two-Way Automata to One-Way Automata. *IBM Journal of Research and Development* 3, 2 (1959), 198–200.

Balder ten Cate and Luc Segoufin. 2010. Transitive Closure Logic, and Nested Tree Walking Automata, and Xpath. *J. ACM* 57, 3 (2010), 1–41.

James W. Thatcher and Jesse B. Wright. 1968. Generalized Finite Automata Theory with an Application to a Decision Problem of Second-Order Logic. *Mathematical Systems Theory* 2, 1 (1968), 57–81.

Wolfgang Thomas. 1982. Classifying Regular Events in Symbolic Logic. *J. Comput. System Sci.* 25 (1982), 360–376.

Boris A. Trakhtenbrot. 1961. Finite Automata and Logic of Monadic Predicates. *Doklady Akademii Nauk SSSR* 149 (1961), 326–329.