Pebble weighted automata and transitive closure logics *

Benedikt Bollig[1], Paul Gastin[1], Benjamin Monmege[1], and Marc Zeitoun[1,2]

[1] LSV, ENS Cachan, CNRS & INRIA, France
firstname.lastname@lsv.ens-cachan.fr

[2] LaBRI, Univ. Bordeaux & CNRS, France

Abstract. We introduce new classes of weighted automata on words. Equipped with pebbles and a two-way mechanism, they go beyond the class of recognizable formal power series, but capture a weighted version of first-order logic with bounded transitive closure. In contrast to previous work, this logic allows for unrestricted use of universal quantification. Our main result states that pebble weighted automata, nested weighted automata, and this weighted logic are expressively equivalent. We also give new logical characterizations of the recognizable series.

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 [5,8,18].

Some 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 [7], first introduced by Schützenberger [15]. The output of a weighted automaton running on a 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 all possible run weights, each calculated as the product of its transition outcomes. Indeed, probabilistic automata and word transducers appear as instances of that framework (see [7, Part IV]).

A logical characterization of weighted automata, however, was established only recently [6], 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 quantification as sum and product from a semiring. To make this definition work, however, one has to restrict the universal first-order quantification, which, otherwise, appears to be too powerful and goes beyond the class of recognizable series. In this paper, we

* Supported by FP7 Quasimodo, ANR-06-SETI-003 DOTS, ARCUS Île de France-Inde.

follow a different approach. Instead of restricting the logic, we define an extended automata model that naturally *reflects* it. Indeed, it turns out that universal quantification 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 [12,9,2], 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 bounded transitive closure and unrestricted use of quantification, extending the classical Boolean case for words [10]. Our equivalence proof makes a detour via another natural concept, named nested weighted automata, which resembles the nested tree-walking automata of [16]. The transitive closure logic also yields alternative characterizations of the recognizable formal power series.

Proofs omitted due to lack of space are available in [4].

2 Notation and background

In this section we set up the notation and we recall some basic results on weighted automata and weighted logics. We refer the reader to [6,7] for details.

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

Formal power series. A *semiring* is a structure $\mathbb{K} = (K, +, \cdot, \mathbf{0}, \mathbf{1})$ where $(K, +, \mathbf{0})$ is a commutative monoid, $(K, \cdot, \mathbf{1})$ is a monoid, $\cdot$ distributes over $+$, and $\mathbf{0}$ is absorbing for $\cdot$. We say that $\mathbb{K}$ is *commutative* if so is $(K, \cdot, \mathbf{1})$. We shall refer in the examples to the usual Boolean semiring $\mathbb{B} = (\{\mathbf{0}, \mathbf{1}\}, \vee, \wedge, \mathbf{0}, \mathbf{1})$ and to the semiring $(\mathbb{N}, +, \cdot, 0, 1)$ of natural numbers, denoted $\mathbb{N}$. A *formal power series* (or *series*, for short) is a mapping $f : \Sigma^+ \to \mathbb{K}$. The set of series is denoted $\mathbb{K}\langle\langle \Sigma^+ \rangle\rangle$. We denote again by $+$ and $\cdot$ the pointwise addition and multiplication (called the *Hadamard product*) on $\mathbb{K}\langle\langle \Sigma^+ \rangle\rangle$, and by $\mathbf{0}$ and $\mathbf{1}$ the constant series with values $\mathbf{0}$ and $\mathbf{1}$, respectively. Then $(\mathbb{K}\langle\langle \Sigma^+ \rangle\rangle, +, \cdot, \mathbf{0}, \mathbf{1})$ is itself a semiring.

Weighted automata. All automata we consider are finite. A *weighted automaton* (wA) over $\mathbb{K} = (K, +, \cdot, \mathbf{0}, \mathbf{1})$ and Σ is a tuple $\mathcal{A} = (Q, \mu, \lambda, \gamma)$, where Q is the set of states, $\mu : \Sigma \to K^{Q \times Q}$ is the transition weight function and $\lambda, \gamma : Q \to K$ are weight functions for entering and leaving a state. The function μ gives, for each $a \in \Sigma$ and $p, q \in Q$, the weight $\mu(a)_{p,q}$ of the transition $p \xrightarrow{a} q$. It extends uniquely to a homomorphism $\mu : \Sigma^+ \to K^{Q \times Q}$. Viewing μ as a mapping $\mu : Q \times \Sigma^+ \times Q \to K$, we sometimes write $\mu(p, u, q)$ instead of $\mu(u)_{p,q}$. A *run* on a word $u = u_1 \cdots u_n$ is a sequence of transitions $\rho = p_0 \xrightarrow{u_1} p_1 \xrightarrow{u_2} \cdots \xrightarrow{u_n} p_n$. The *weight* of the run ρ is $\mathrm{weight}(\rho) \stackrel{\text{def}}{=} \lambda(p_0) \cdot [\prod_{i=1}^{n} \mu(p_{i-1}, u_i, p_i)] \cdot \gamma(p_n)$, and the *weight* $[\![\mathcal{A}]\!](u)$ of u is the sum of all weights of runs on u, which can also be computed as $[\![\mathcal{A}]\!](u) = \lambda \cdot \mu(u) \cdot \gamma$, viewing λ, $\mu(u)$, γ as matrices of dimension

$1 \times |Q|$, $|Q| \times |Q|$ and $|Q| \times 1$, respectively. We call $[\![\mathcal{A}]\!] \in \mathbb{K}\langle\langle \Sigma^+ \rangle\rangle$ the *behavior*, or *semantics* of $\mathcal{A}$. A series $f \in \mathbb{K}\langle\langle \Sigma^+ \rangle\rangle$ is *recognizable* if it is the behavior of some wA. We let $\mathbb{K}^{\mathrm{rec}}\langle\langle \Sigma^+ \rangle\rangle$ be the collection of all recognizable series.

Example 1. Consider $(\mathbb{N}, +, \cdot, 0, 1)$ and let $\mathcal{A}$ be the automaton with a single state, $\mu(a) = 2$ for all $a \in \Sigma$, and $\lambda = \gamma = 1$. Then, $[\![\mathcal{A}]\!](u) = 2^{|u|}$ for all $u \in \Sigma^+$.

It is well-known that $\mathbb{K}^{\mathrm{rec}}\langle\langle \Sigma^+ \rangle\rangle$ is stable under $+$ and, if $\mathbb{K}$ is commutative, also under $\cdot$, making $(\mathbb{K}^{\mathrm{rec}}\langle\langle \Sigma^+ \rangle\rangle, +, \cdot, \mathbf{0}, \mathbf{1})$ a subsemiring of $(\mathbb{K}\langle\langle \Sigma^+ \rangle\rangle, +, \cdot, \mathbf{0}, \mathbf{1})$.

Weighted logics. We fix infinite supplies $\mathrm{Var} = \{x, y, z, t, \ldots\}$ of first-order variables, and $\mathrm{VAR} = \{X, Y, \ldots\}$ of second-order variables. The class of *weighted monadic second-order* formulas over $\mathbb{K}$ and Σ, denoted $\mathrm{MSO}(\mathbb{K}, \Sigma)$ (shortly MSO), is given by the following grammar, with $k \in K$, $a \in \Sigma$, $x, y \in \mathrm{Var}$ and $X \in \mathrm{VAR}$:

$$\varphi ::= k \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x\varphi \mid \forall x\varphi \mid \exists X\varphi \mid \forall X\varphi\,.$$

For $\varphi \in \mathrm{MSO}(\mathbb{K}, \Sigma)$, let $\mathrm{Free}(\varphi)$ denote the set of free variables of φ. If $\mathrm{Free}(\varphi) = \emptyset$, then φ is called a *sentence*. For a finite set $\mathcal{V} \subseteq \mathrm{Var} \cup \mathrm{VAR}$ and a word $u \in \Sigma^+$, a $(\mathcal{V}, u)$-*assignment* is a function σ that maps a first-order variable in $\mathcal{V}$ to an element of $\mathrm{Pos}(u)$ and a second-order variable in $\mathcal{V}$ to a subset 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 σ. 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, σ), where σ is a $(\mathcal{V}, u)$-assignment, can be encoded as a word over the extended alphabet $\Sigma_{\mathcal{V}} \stackrel{\mathrm{def}}{=} \Sigma \times \{0,1\}^{\mathcal{V}}$. We write a word $(u_1, \sigma_1) \cdots (u_n, \sigma_n) \in \Sigma_{\mathcal{V}}^+$ as (u, σ) where $u = u_1 \cdots u_n$ and $\sigma = \sigma_1 \cdots \sigma_n$. We call (u, σ) *valid* if, for each first-order variable $x \in \mathcal{V}$, the x-row of σ contains exactly one 1. If (u, σ) is valid, then σ can be considered as 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.

Fix a finite set $\mathcal{V}$ of variables such that $\mathrm{Free}(\varphi) \subseteq \mathcal{V}$. The semantics $[\![\varphi]\!]_{\mathcal{V}} \in \mathbb{K}\langle\langle \Sigma_{\mathcal{V}}^+ \rangle\rangle$ of φ wrt. $\mathcal{V}$ is given as follows: if (u, σ) is not valid, we set $[\![\varphi]\!]_{\mathcal{V}}(u, \sigma) = \mathbf{0}$, otherwise $[\![\varphi]\!]_{\mathcal{V}}$ is given by Figure 1. Hereby, the product follows the natural order on $\mathrm{Pos}(u)$ and some fixed order on the power set of $\mathrm{Pos}(u)$. We simply write $[\![\varphi]\!]$ for $[\![\varphi]\!]_{\mathrm{Free}(\varphi)}$ and say that φ is recognizable if so is $[\![\varphi]\!]$. We note $\mathbb{K}^{\mathrm{MSO}}\langle\langle \Sigma^+ \rangle\rangle$ the class of all series definable by a sentence of $\mathrm{MSO}(\mathbb{K}, \Sigma)$.

Example 2. For $\mathbb{K} = \mathbb{B}$, recognizable and $\mathrm{MSO}(\mathbb{K}, \Sigma)$-definable languages coincide. In contrast, for $\mathbb{K} = (\mathbb{N}, +, \cdot, 0, 1)$, the very definition yields $[\![\forall x \forall y\, 2]\!](u) = 2^{|u|^2}$, which is not recognizable [6]. Indeed, the function computed by a wA $\mathcal{A}$ satisfies $[\![\mathcal{A}]\!](u) = 2^{\mathcal{O}(|u|)}$. Also observe that the behavior of the automaton of Example 1 is $[\![\forall y\, 2]\!]$. Therefore, recognizable series are not stable under universal first-order quantification.

Let $\mathrm{bMSO}(\mathbb{K}, \Sigma)$ be the syntactic Boolean fragment of $\mathrm{MSO}(\mathbb{K}, \Sigma)$ given by

$$\varphi ::= \mathbf{0} \mid \mathbf{1} \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg\varphi \mid \varphi \wedge \varphi \mid \forall x\varphi \mid \forall X\varphi\,,$$

$$[\![k]\!]_{\mathcal{V}}(u,\sigma) = k, \quad \text{for } k \in K$$

$$[\![P_a(x)]\!]_{\mathcal{V}}(u,\sigma) = \begin{cases} \mathbf{1} \text{ if } u_{\sigma(x)} = a \\ \mathbf{0} \text{ otherwise} \end{cases}$$

$$[\![x \in X]\!]_{\mathcal{V}}(u,\sigma) = \begin{cases} \mathbf{1} \text{ if } \sigma(x) \in \sigma(X) \\ \mathbf{0} \text{ otherwise} \end{cases}$$

$$[\![x \leq y]\!]_{\mathcal{V}}(u,\sigma) = \begin{cases} \mathbf{1} \text{ if } \sigma(x) \leq \sigma(y) \\ \mathbf{0} \text{ otherwise} \end{cases}$$

$$[\![\neg\varphi]\!]_{\mathcal{V}}(u,\sigma) = \begin{cases} \mathbf{1} \text{ if } [\![\varphi]\!]_{\mathcal{V}}(u,\sigma) = \mathbf{0} \\ \mathbf{0} \text{ otherwise} \end{cases}$$

$$[\![\varphi_1 \vee \varphi_2]\!]_{\mathcal{V}}(u,\sigma) = [\![\varphi_1]\!]_{\mathcal{V}}(u,\sigma) + [\![\varphi_2]\!]_{\mathcal{V}}(u,\sigma)$$

$$[\![\varphi_1 \wedge \varphi_2]\!]_{\mathcal{V}}(u,\sigma) = [\![\varphi_1]\!]_{\mathcal{V}}(u,\sigma) \cdot [\![\varphi_2]\!]_{\mathcal{V}}(u,\sigma)$$

$$[\![\exists x\varphi]\!]_{\mathcal{V}}(u,\sigma) = \sum_{i \in \mathrm{Pos}(u)} [\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(u, \sigma[x \mapsto i])$$

$$[\![\exists X\varphi]\!]_{\mathcal{V}}(u,\sigma) = \sum_{I \subseteq \mathrm{Pos}(u)} [\![\varphi]\!]_{\mathcal{V}\cup\{X\}}(u, \sigma[X \mapsto I])$$

$$[\![\forall x\varphi]\!]_{\mathcal{V}}(u,\sigma) = \prod_{i \in \mathrm{Pos}(u)} [\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(u, \sigma[x \mapsto i])$$

$$[\![\forall X\varphi]\!]_{\mathcal{V}}(u,\sigma) = \prod_{I \subseteq \mathrm{Pos}(u)} [\![\varphi]\!]_{\mathcal{V}\cup\{X\}}(u, \sigma[X \mapsto I])$$

Fig. 1. Semantics of weighted MSO

where $a \in \Sigma$, $x, y \in \mathrm{Var}$ and $X \in \mathrm{VAR}$. One can check, by induction, that the semantics of any bMSO formula over an arbitrary semiring $\mathbb{K}$ assumes values in $\{\mathbf{0}, \mathbf{1}\}$ and coincides with the classical semantics in $\mathbb{B}$.

We use macros for Boolean disjunction $\varphi \veebar \psi \stackrel{\text{def}}{=} \neg(\neg\varphi \wedge \neg\psi)$ and Boolean existential quantifications $\exists\!\!\!\cdot x\varphi \stackrel{\text{def}}{=} \neg\forall x\neg\varphi$, and $\exists\!\!\!\cdot X\varphi \stackrel{\text{def}}{=} \neg\forall X\neg\varphi$. The semantics of $\veebar$ and $\exists\!\!\!\cdot$ coincide with the classical semantics of disjunction and existential quantification in the Boolean semiring $\mathbb{B}$. Finally, we define $\varphi \xrightarrow{+} \psi \stackrel{\text{def}}{=} \neg\varphi \vee (\varphi \wedge \psi)$ so that, if φ is a Boolean formula (*i.e.*, $[\![\varphi]\!](\Sigma^+) \subseteq \{\mathbf{0}, \mathbf{1}\}$), $[\![\varphi \xrightarrow{+} \psi]\!](u,\sigma) = [\![\psi]\!](u,\sigma)$ if $[\![\varphi]\!](u,\sigma) = \mathbf{1}$, and $[\![\varphi \xrightarrow{+} \psi]\!](u,\sigma) = \mathbf{1}$ if $[\![\varphi]\!](u,\sigma) = \mathbf{0}$.

A common fragment of $\mathrm{MSO}(\mathbb{K}, \Sigma)$ is the weighted first-order logic $\mathrm{FO}(\mathbb{K}, \Sigma)$, where no second-order quantifier appears (note that second order variables may still appear free): $\varphi ::= k \mid P_a(x) \mid x \leq y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x\varphi \mid \forall x\varphi$.

We define similarly $\mathrm{bFO}(\mathbb{K}, \Sigma)$ as the fragment of $\mathrm{bMSO}(\mathbb{K}, \Sigma)$ with no second-order quantifiers. We also let bFO+mod be the fragment of bMSO consisting of bFO augmented with modulo constraints $x \equiv_\ell m$ for constants $1 \leq m \leq \ell$ (since the positions of words start with 1, it is more convenient to compute modulo as a value between 1 and ℓ). The semantics is given by $[\![x \equiv_\ell m]\!](u,\sigma) = \mathbf{1}$ if $\sigma(x) \equiv m \bmod \ell$ and $\mathbf{0}$ otherwise: it can be defined in bMSO by

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

For $\mathcal{L} \subseteq \mathrm{bMSO}$ closed under $\vee$, $\wedge$ and $\neg$, an *$\mathcal{L}$-step formula* is a formula obtained from the grammar $\varphi ::= k \mid \alpha \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$, with $k \in K$ and $\alpha \in \mathcal{L}$. In particular, quantifications are only allowed in formulas $\alpha \in \mathcal{L}$. 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.

Lemma 3. *For every $\mathcal{L}$-step formula φ, one can construct an equivalent formula $\psi = \bigvee_i (\varphi_i \wedge k_i)$ with $\varphi_i \in \mathcal{L}$ and $k_i \in K$, with the same set of free variables.*

From now on, we freely use Lemma 3, using the special form it provides for $\mathcal{L}$-step formulas. All bMSO-step formulas are clearly recognizable. By [6], $\forall x\, \varphi$ is recognizable for any bMSO-step formula φ. The fragment $\mathrm{RMSO}(\mathbb{K}, \Sigma)$ of MSO is defined by restricting universal second-order quantification to bMSO formulas and universal first-order quantification to bMSO-step formulas.

Theorem 4 ([6]). *A series is recognizable iff it is definable in* $\mathrm{RMSO}(\mathbb{K}, \Sigma)$.

3 Transitive closure logic and weighted automata

To ease notation, we write $[\![\varphi(x,y)]\!](u,i,j)$ instead of $[\![\varphi(x,y)]\!](u,[x \mapsto i, y \mapsto j])$. We allow constants, modulo constraints and comparisons, like *e.g.* $x \leq y+2$. 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 bFO-formulas, except "$x \equiv_\ell m$" with $1 \leq m \leq \ell$, which is bMSO-definable.

Bounded transitive closure. For a formula $\varphi(x,y)$ with at least two free variables x and y, and an integer $N > 0$, we let $\varphi^{1,N}(x,y) \stackrel{\text{def}}{=} (x \leq y \leq x+N) \wedge \varphi(x,y)$ and for $n \geq 2$, we define the formula $\varphi^{n,N}(x,y)$ as

$$\exists z_0 \cdots \exists z_n \big[x = z_0 \wedge y = z_n \wedge \bigwedge_{1 \leq \ell \leq n} (z_{\ell-1} < z_\ell \leq z_{\ell-1} + N) \wedge \varphi(z_{\ell-1}, z_\ell) \big] . \quad (1)$$

We define for each $N > 0$ the N-$\mathrm{TC}^<_{xy}$ operator by $N\text{-}\mathrm{TC}^<_{xy}\varphi = \bigvee_{n \geq 1} \varphi^{n,N}$. This infinite disjunction is well-defined: $[\![\varphi^{n,N}(x,y)]\!](u,\sigma) = 0$ if $n \geq \max(2, |u|)$, *i.e.*, on each pair (u,σ), only finitely many disjuncts assume a nonzero value. Intuitively, the N-$\mathrm{TC}^<_{xy}$ operator generalizes the forward transitive closure operator from the Boolean case, but limiting it to *intermediate* forward steps of length $\leq N$. The fragment $\mathrm{FO}{+}\mathrm{BTC}^<(\mathbb{K}, \Sigma)$ is then defined by the grammar

$$\varphi ::= k \mid P_a(x) \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x\varphi \mid \forall x\varphi \mid N\text{-}\mathrm{TC}^<_{xy}\varphi\,,$$

with $N \geq 1$ and the restriction that one can apply negation only over bFO-formulas. We denote by $\mathbb{K}^{\mathrm{FO}+\mathrm{BTC}^<}\langle\!\langle \Sigma^+ \rangle\!\rangle$ the class of all $\mathrm{FO}{+}\mathrm{BTC}^<(\mathbb{K}, \Sigma)$-definable series.

Example 5. Let $\varphi(x,y) \stackrel{\text{def}}{=} (y = x{+}1) \wedge \forall z\, 2 \wedge (x = 1 \stackrel{+}{\to} \forall z\, 2)$ over $\mathbb{K} = \mathbb{N}$. Let $u = u_1 \cdots u_n$. For any $N \geq 1$, we have $[\![N\text{-}\mathrm{TC}^<_{xy}\varphi]\!](u, \mathsf{first}, \mathsf{last}) = \prod_{i=1}^{n-1} [\![\varphi]\!](u, i, i{+}1)$ due to the constraint $y = x{+}1$ in φ. Now, $[\![\varphi]\!](u,1,2) = 2^{2|u|}$ and $[\![\varphi]\!](u,i,i{+}1) = 2^{|u|}$ if $i > 1$, so $[\![N\text{-}\mathrm{TC}^<_{xy}\varphi]\!](u, \mathsf{first}, \mathsf{last}) = 2^{|u|^2}$. This example shows in particular that the class of recognizable series is not closed under $\mathrm{BTC}^<$.

Example 6. It is well-known that *modulo* can be expressed in $\mathrm{bFO}{+}\mathrm{BTC}^<$ by

$$x \equiv_\ell m \stackrel{\text{def}}{=} (x = m) \vee [\ell\text{-}\mathrm{TC}^<_{yz}(z = y + \ell)](m, x)\,.$$

We now consider syntactical restrictions of FO+BTC$^<$, inspired by normal form formulas of [13] where only one "external" transitive closure is allowed.

For $\mathcal{L} \subseteq$ bMSO, $\mathrm{BTC}^<_{\mathrm{step}}(\mathcal{L})$ consists of formulas of the form $N\text{-}\mathrm{TC}^<_{xy}\varphi$, where $\varphi(x,y)$ is an $\mathcal{L}$-step formula with two free variables x, y. We say that $f \in \mathbb{K}\langle\langle \Sigma^+ \rangle\rangle$ is $\mathrm{BTC}^<_{\mathrm{step}}(\mathcal{L})$-*definable* if there exists an $\mathcal{L}$-step formula $\varphi(x,y)$ such that, for all $u \in \Sigma^+$, $f(u) = [\![N\text{-}\mathrm{TC}^<_{xy}\varphi]\!](u, \mathsf{first}, \mathsf{last})$.

For $\mathcal{L} \subseteq$ bMSO, let $\exists\forall_{\mathrm{step}}(\mathcal{L})$ consists of all MSO-formulas of the form $\exists X \forall x\, \varphi(x, X)$ with φ an $\mathcal{L}$-step formula: this defines a fragment of the logic $\mathrm{RMSO}(\mathbb{K}, \Sigma)$ introduced in [6]. The following result characterizes the expressive power of weighted automata.

Theorem 7. *Let $\mathbb{K}$ be a (possibly noncommutative) semiring and $f \in \mathbb{K}\langle\langle \Sigma^+ \rangle\rangle$. The following assertions are equivalent over $\mathbb{K}$ and Σ:*

(1) *f is recognizable.*
(2) *f is $\mathrm{BTC}^<_{\mathrm{step}}$(bFO+mod)-definable.*
(3) *f is $\mathrm{BTC}^<_{\mathrm{step}}$(bMSO)-definable.*
(4) *f is $\exists\forall_{\mathrm{step}}$(bFO)-definable.*
(5) *f is $\exists\forall_{\mathrm{step}}$(bMSO)-definable.*

Proof. Fix a weighted automaton $\mathcal{A} = (Q, \mu, \lambda, \gamma)$ with $Q = \{1, \dots, n\}$. For $d \geq 1$ and $p, q \in Q$, we use a formula $\psi^d_{p,q}(x,y)$ to compute the weight of the factor of length d located between positions x and y, when $\mathcal{A}$ goes from p to q:

$$\psi^d_{p,q}(x,y) \stackrel{\text{def}}{=} (y = x + d - 1) \wedge \bigvee_{v = v_1 \cdots v_d} \Big(\mu(v)_{p,q} \wedge \bigwedge_{1 \leq i \leq d} P_{v_i}(x + i - 1) \Big).$$

We construct a formula $\varphi(x,y)$ of bFO+mod allowing to define the semantics of $\mathcal{A}$ using a BTC$^<$: $[\![\mathcal{A}]\!](u) = [\![2n\text{-}\mathrm{TC}^<_{xy}\varphi]\!](u, \mathsf{first}, \mathsf{last})$. The idea, inspired by [17], consists in making the transitive closure pick positions $z_\ell = \ell n + q_\ell$, with $1 \leq q_\ell \leq n$, for successive values of ℓ, to encode runs of $\mathcal{A}$ going through state q_ℓ just before reading the letter at position $\ell n + 1$. To make this still work for $\ell = 0$, one can assume wlog. that $\lambda(1) = \mathbf{1}$ and $\lambda(q) = \mathbf{0}$ for $q \neq 1$, *i.e.*, the only initial state yielding a nonzero value is $q_0 = 1$. Consider slices $[\ell n + 1, (\ell + 1)n]$ of positions in the word where we evaluate the formula (the last slice might be incomplete). Each position y is located in exactly one such slice. We write $\langle y \rangle = \ell n + 1$ for the first position of that slice, as well as $[y] \stackrel{\text{def}}{=} y + 1 - \langle y \rangle \in Q$ for the corresponding "offset". Notice that, for $q \in Q$, $[y] = q$ can be expressed in bFO+mod simply by $y \equiv_n q$. Hence, we will freely use $[y]$ as well as $\langle y \rangle = y + 1 - [y]$ as macros in formulas. Our BTC$^<$-formula picks positions x and y marked $\bullet$ in Figure 2, and computes the weight of the factor of length n between the positions $\langle x \rangle$ and $\langle y \rangle - 1$, assuming states $[x]$ and $[y]$ just before and after these positions. The formula φ distinguishes the cases where x is far or near from the last position:

$$\begin{aligned} \varphi(x,y) = &\Big((\langle x \rangle + 2n \leq \mathsf{last}) \wedge (\langle y \rangle = \langle x \rangle + n) \wedge \psi^n_{[x],[y]}(\langle x \rangle, \langle y \rangle - 1) \Big) \\ &\vee \Big((\langle x \rangle + 2n > \mathsf{last}) \wedge (y = \mathsf{last}) \wedge \bigvee_{q \in Q} \psi^{y - \langle x \rangle + 1}_{[x],q}(\langle x \rangle, y) \wedge \gamma(q) \Big). \end{aligned}$$

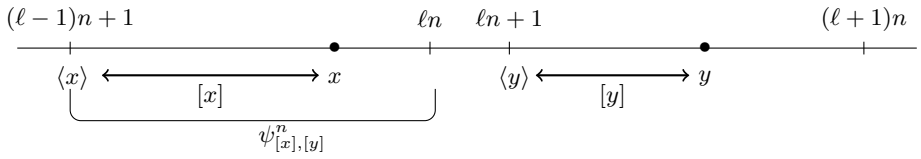

Fig. 2. Positions picked by the $\mathrm{BTC}^<$-formula

4 Weighted nested automata

Example 2 shows that weighted automata lack closure properties to capture FO+$\mathrm{BTC}^<$. We introduce a notion of nested automata making up for this gap.

For $r \geq 0$, the class r-nwA(Σ) (r-nwA if Σ is understood) of *r-nested weighted automata* over Σ (and $\mathbb{K}$) consists of all tuples $(Q, \mu, \lambda, \gamma)$ where Q is the set of states, $\lambda, \gamma : Q \to K$, and $\mu : Q \times \Sigma \times Q \to (r-1)$-nwA($\Sigma \times \{0,1\}$). Here, we agree that (-1)-nwA $= K$. In particular, a 0-nwA(Σ) is a weighted automaton over Σ. Intuitively, the weight of a transition is computed by an automaton of the preceding level running on the whole word, where the additional $\{0,1\}$ component marks the letter of the transition whose weight is to be computed.

Let us formally define the behavior $[\![\mathcal{A}]\!] \in \mathbb{K}\langle\langle \Sigma^+ \rangle\rangle$ of $\mathcal{A} = (Q, \mu, \lambda, \gamma) \in r$-nwA($\Sigma$). If $r = 0$, then $[\![\mathcal{A}]\!]$ is the behavior of $\mathcal{A}$ considered as wA over Σ. For $r \geq 1$, the weight of a *run* $\rho = q_0 \xrightarrow{u_1} q_1 \xrightarrow{u_2} \cdots \xrightarrow{u_n} q_n$ of $\mathcal{A}$ on $u = u_1 \cdots u_n$ is

$$\mathrm{weight}(\rho) \stackrel{\text{def}}{=} \lambda(q_0) \cdot \Big[\prod_{i=1}^{n} [\![\mu(q_{i-1}, u_i, q_i)]\!](u, i)\Big] \cdot \gamma(q_n)\,,$$

where $(u, i) \in (\Sigma \times \{0,1\})^+$ is the word $v = v_1 \cdots v_n$ with $v_i = (u_i, 1)$ and $v_j = (u_j, 0)$ if $j \neq i$. As usual, $[\![\mathcal{A}]\!](u)$ is the sum of the weights of all runs of $\mathcal{A}$ on u. Note that, unlike the nested automata of [16], the values given by lower automata do not explicitly influence the possible transitions.

A series $f \in \mathbb{K}\langle\langle \Sigma^+ \rangle\rangle$ is *r-nwA-recognizable* if $f = [\![\mathcal{A}]\!]$ for some r-nwA $\mathcal{A}$. It is nwA-*recognizable* if it is r-nwA-recognizable for some r. We let $\mathbb{K}^{r\text{-nwA}}\langle\langle \Sigma^+ \rangle\rangle$ (resp., $\mathbb{K}^{\mathrm{nwA}}\langle\langle \Sigma^+ \rangle\rangle$) be the class of r-nwA-recognizable (resp., nwA-recognizable) series over $\mathbb{K}$ and Σ.

Example 8. A 1-nwA recognizing the series $u \mapsto 2^{|u|^2}$ over $\mathbb{N}$ is $\mathcal{A} = (\{p\}, \mu, \mathbf{1}, \mathbf{1})$ where, for every $a \in \Sigma$, $\mu(p, a, p)$ is the weighted automaton of Example 1.

We can generalize the proof of (1) $\Rightarrow$ (2) in Theorem 7 in order to get the following result. The converse will be obtained in Section 5.

Proposition 9. *Every* nwA*-recognizable series is* FO+$\mathrm{BTC}^<$*-definable.*

5 Pebble weighted automata

We now consider *pebble weighted automata* (pwA). A pwA 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 either drop or lift a pebble at the current head position. Applicable transitions and weights depend on the current letter, current state, and the pebbles carried by the current position. Pebbles are handled using a stack policy: if the automaton has r pebbles and pebbles $\ell, \dots, r$ have already been dropped, it can either lift pebble ℓ (if $\ell \leq r$), drop pebble $\ell - 1$ (if $\ell \geq 2$), 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 $\tilde{\Sigma} = \Sigma \uplus \{\triangleright, \triangleleft\}$. To compute the value of $w = w_1 \cdots w_n \in \Sigma^+$, a pwA will work on a tape holding $\widetilde{w} = \triangleright w \triangleleft$. For convenience, we number the letters of $\widetilde{w}$ from 0, setting $\widetilde{w}_0 = \triangleright$, $\widetilde{w}_{n+1} = \triangleleft$, and $\widetilde{w}_i = w_i$ for $1 \leq i \leq n$.

Let $r \geq 0$. Formally, an *r-pebble weighted automaton (r-*pwA*)* over $\mathbb{K}$ and Σ is a pair $\mathcal{A} = (Q, \mu)$ where Q is a finite set of states and $\mu : Q \times \tilde{\Sigma} \times 2^r \times D \times Q \to K$ is the transition weight function, with $D = \{\leftarrow, \rightarrow, \mathrm{drop}, \mathrm{lift}\}$.

A *configuration* of a r-pwA $\mathcal{A}$ on a word $w \in \Sigma^+$ of length n is a triple $(p, i, \zeta) \in Q \times \{0, \dots, n+2\} \times \{1, \dots, n\}^{\leq r}$. The word w itself will be understood. Informally, p denotes the current state of $\mathcal{A}$ and i is the head position in $\widetilde{w}$, *i.e* positions 0 and $n+1$ point to $\triangleright$ and $\triangleleft$, respectively, position $1 \leq i \leq n$ points to $\widetilde{w}_i \in \Sigma$, and position $n+2$ is outside $\widetilde{w}$. Finally, $\zeta = \zeta_\ell \cdots \zeta_r$ with $1 \leq \ell \leq r+1$ encodes the locations of pebbles $\ell, \dots, r$ ($\zeta_m \in \{1, \dots, n\}$ is the position of pebble m) while pebbles $1, \dots, \ell-1$ are currently not on the tape. For $i \in \{0, \dots, n+1\}$, we set $\zeta^{-1}(i) = \{m \in \{\ell, \dots, r\} \mid \zeta_m = i\}$ (viewing ζ as a partial function from $\{1, \dots, r\}$ to $\{0, \dots, n+1\}$). Note that $\zeta^{-1}(0) = \zeta^{-1}(n+1) = \emptyset$.

There is a *step* of weight k from configuration (p, i, ζ) to configuration (q, j, η) if $i \leq n+1$, $k = \mu(p, \widetilde{w}_i, \zeta^{-1}(i), d, q)$, and

$$\begin{cases} j = i-1 & \text{if } d = \leftarrow \\ j = i+1 & \text{if } d = \rightarrow \\ j = i & \text{otherwise} \end{cases} \quad \text{and} \quad \begin{cases} \eta = i\zeta & \text{if } d = \mathrm{drop} \\ \zeta = i\eta & \text{if } d = \mathrm{lift} \\ \eta = \zeta & \text{otherwise.} \end{cases}$$

A *run* ρ of $\mathcal{A}$ is a sequence of steps from a configuration $(p, 0, \varepsilon)$ to a configuration $(q, n+2, \varepsilon)$ (at the end, no pebble is left on the tape). We denote by $\mathrm{weight}(\rho)$ the product of weights of the steps of run ρ (from left to right, but we will mainly work with a commutative semiring in this section). The run ρ is *simple* if whenever two configurations α and β appear in ρ, we have $\alpha \neq \beta$.

The series $[\![\mathcal{A}]\!] \in \mathbb{K}\langle\langle \Sigma^+ \rangle\rangle$ is defined by $[\![\mathcal{A}]\!](w) = \sum_{\rho \text{ simple run on } w} \mathrm{weight}(\rho)$. We denote by $\mathbb{K}^{r\text{-pwA}}\langle\langle \Sigma^+ \rangle\rangle$ the collection of formal power series definable by a r-pwA, and we let $\mathbb{K}^{\mathrm{pwA}}\langle\langle \Sigma^+ \rangle\rangle = \bigcup_{r \geq 0} \mathbb{K}^{r\text{-pwA}}\langle\langle \Sigma^+ \rangle\rangle$. Note that a 0-pwA is in fact a 2-way weighted automaton. It follows from Theorem 11 that 2-way wA have the same expressive power as classical (1-way) wA.

Example 10. Let us sketch a 1-pwA $\mathcal{A}$ 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 $\mathbf{1}$. 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 $[\![\mathcal{A}]\!](u) = 2^{|u|^2}$.

Theorem 11. *For every commutative semiring $\mathbb{K}$, and every $r \geq 0$, we have*
(1) $\mathbb{K}^{r\text{-pwA}}\langle\langle \Sigma^+ \rangle\rangle \subseteq \mathbb{K}^{r\text{-nwA}}\langle\langle \Sigma^+ \rangle\rangle$
(2) $\mathbb{K}^{\mathrm{FO+BTC}^<}\langle\langle \Sigma^+ \rangle\rangle = \mathbb{K}^{\mathrm{pwA}}\langle\langle \Sigma^+ \rangle\rangle = \mathbb{K}^{\mathrm{nwA}}\langle\langle \Sigma^+ \rangle\rangle$.
(3) $\mathbb{K}^{\mathrm{FO+BTC}^<}\langle\langle \Sigma^+ \rangle\rangle \subseteq \mathbb{K}^{\mathrm{pwA}}\langle\langle \Sigma^+ \rangle\rangle$ *holds even for noncommutative semirings.*

Proof. (1) We provide a translation of a generalized version of r-pwA to r-nwA. That generalized notion equips an r-pwA $\mathcal{A} = (P, \mu)$ with an equivalence relation $\sim \subseteq P \times P$, which is canonically extended to configurations of $\mathcal{A}$: we write $(p, i, u) \sim (p', i', u')$ if $p \sim p'$, $i = i'$, and $u = u'$.

The semantics $[\![\mathcal{A}]\!]_\sim$ is then defined by replacing equality of configurations with $\sim$ in the definition of simple run. To stress this fact, we henceforth say that a run is $\sim$-*simple*. So let $r \geq 0$, $\mathcal{A} = (P, \mu)$ be an r-pwA over $\mathbb{K}$ and Σ, and $\sim \subseteq P \times P$ be an equivalence relation. We assume wlog. that all runs in which a pebble is dropped and immediately lifted have weight $\mathbf{0}$. We build an r-nwA $\langle\mathcal{A}\rangle_\sim = (Q, \nu, \lambda, \gamma)$ over Σ such that $[\![\langle\mathcal{A}\rangle_\sim]\!] = [\![\mathcal{A}]\!]_\sim$.

The construction of $\langle\mathcal{A}\rangle_\sim$ proceeds inductively on the number of pebbles r. It involves two alternating transformations, as illustrated in Figure 3. The left-hand side depicts a $\sim$-simple run of $\mathcal{A}$ on some word w with factor ab. To simulate such a run, $\langle\mathcal{A}\rangle_\sim$ scans w from left to right and guesses, at each position i, the sequence of those states and directions that are encountered at i while pebble r has not, or just, been dropped. The state of $\langle\mathcal{A}\rangle_\sim$ taken before reading the a at position i is $q = p_0 \to p_3 \leftarrow p_4 \operatorname{drop} p_5\, \hat{p}_5 \operatorname{lift} p_6 \leftarrow p_7 \operatorname{drop} p_8\, \hat{p}_8 \operatorname{lift} p_9 \to$ (which is enriched by the input letter a, as will be explained below). As the micro-states $p_0, p_3, \ldots$ form a segment of a $\sim$-simple run, $p_0, p_3, p_4, p_6, p_7, p_9$ are pairwise distinct (wrt. $\sim$) and so are $p_5, \hat{p}_5, p_8, \hat{p}_8$. Segments when pebble r is dropped on position i are defered to a $(r-1)$-nwA $\mathcal{B}_q$, which is called at position i and computes the run segments from p_5 to $\hat{p}_5$ and from p_8 to $\hat{p}_8$ that both start in i. To this aim, $\mathcal{B}_q$ works on an extension of w where position i is marked, indicating that pebble r is considered to be at i.

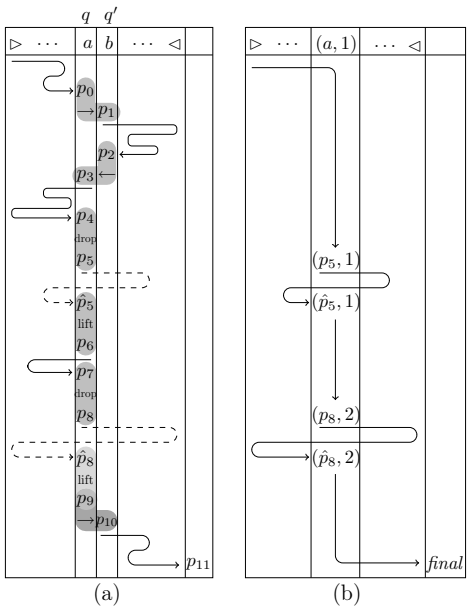

Fig. 3. (a) Runs of r-pwA $\mathcal{A}$ and r-nwA $\langle\mathcal{A}\rangle_\sim$; (b) run of $(r-1)$-pwA $\mathcal{A}_q$

We define the r-nwA $\langle\mathcal{A}\rangle_\sim$. Let $\Pi = (P\{\to, \leftarrow\} \cup P\{\text{drop}\}PP\{\text{lift}\})^* P\{\to\}$. Sequences from Π keep track of states and directions that are taken at one given position. As aforementioned, they must meet the requirements of $\sim$-simple runs so that only some of them can be considered as states. Formally, given

$\pi \in \mathit{\Pi}$, we define projections $\mathrm{pr}_1(\pi) \in P^+$ and $\mathrm{pr}_2(\pi) \in (PP)^*$ inductively by $\mathrm{pr}_1(\varepsilon) = \mathrm{pr}_2(\varepsilon) = \varepsilon$ and

$$\mathrm{pr}_1(p \to \pi) = \mathrm{pr}_1(p \leftarrow \pi) = p\,\mathrm{pr}_1(\pi) \qquad \mathrm{pr}_1(p\,\mathrm{drop}\,p_1\,\hat{p}_1\,\mathrm{lift}\,\pi) = p\,\mathrm{pr}_1(\pi)$$
$$\mathrm{pr}_2(p \to \pi) = \mathrm{pr}_2(p \leftarrow \pi) = \mathrm{pr}_2(\pi) \qquad \mathrm{pr}_2(p\,\mathrm{drop}\,p_1\,\hat{p}_1\,\mathrm{lift}\,\pi) = p_1\,\hat{p}_1\,\mathrm{pr}_2(\pi).$$

Let $\mathit{\Pi}_\sim$ denote the set of sequences $\pi \in \mathit{\Pi}$ such that $\mathrm{pr}_1(\pi)$ consists of pairwise distinct states wrt. $\sim$ and so does $\mathrm{pr}_2(\pi)$ (there might be states that occur in both $\mathrm{pr}_1(\pi)$ and $\mathrm{pr}_2(\pi)$). With this, we set $Q = (\tilde{\Sigma} \uplus \{\square\}) \times \mathit{\Pi}_\sim$. The letter $a \in \tilde{\Sigma} \uplus \{\square\}$ of a state $(a, \pi) \in Q$ will denote the symbol that is to be read next. Symbol $\square$ means that there is no letter left so that the automaton is beyond the scope of $\triangleright w \triangleleft$ when w is the input word.

Next, we explain how the weight of the run segments of $\mathcal{A}$ with lifted pebble r is computed in $\langle\mathcal{A}\rangle_\sim$. Two neighboring states of $\langle\mathcal{A}\rangle_\sim$ need to match each other, which can be checked locally by means of transitions. To determine a corresponding weight, we first count weights of those transitions that move from the current position i to $i+1$ or from $i+1$ to i. This is the reason why a state of $\langle\mathcal{A}\rangle_\sim$ also maintains the letter that is to be read next. In Figure 3(a), the 7 micro-transitions that we count in the step from q to q' are highlighted in gray. Assuming $q = (a, \pi_0)$ and $q' = (b, \pi_1)$, we obtain a value $\mathrm{weight}_{a,b}(\pi_0 \,|\, \pi_1)$ as the product $\mu(p_0, a, \emptyset, \to, p_1) \cdot \mu(p_2, b, \emptyset, \leftarrow, p_3) \cdot \cdots \cdot \mu(\hat{p}_8, a, \{r\}, \mathrm{lift}, p_9) \cdot \mu(p_9, a, \emptyset, \to, p_{10})$. The formal definition of $\mathrm{weight}_{a,b}(\pi_0 \,|\, \pi_1) \in K$ is omitted.

We are now prepared to define the components ν, λ, γ of $\langle\mathcal{A}\rangle_\sim$. For $q_0 = (a_0, \pi_0)$ and $q_1 = (a_1, \pi_1)$ states in Q, we set

$$\begin{aligned} \lambda(q_0) &= \textstyle\sum_{(\triangleright,\pi)\in Q} \mathrm{weight}_{\triangleright,a_0}(\pi \,|\, \pi_0) \\ \gamma(q_0) &= \textstyle\sum_{(\square,\pi)\in Q} \mathrm{weight}_{\triangleleft,\square}(\pi_0 \,|\, \pi) && \text{if } a_0 = \triangleleft \\ \nu(a_0)_{q_0,q_1} &= \mathrm{weight}_{a_0,a_1}(\pi_0 \,|\, \pi_1) \cdot \mathcal{B}_{q_0}\,. \end{aligned}$$

Here, $\mathcal{B}_q$ is the constant $\mathbf{1}$ if $r = 0$. Otherwise, $\mathcal{B}_q = \langle\mathcal{A}_q\rangle_{\sim_q}$ is an $(r-1)$-nwA over $\mathit{\Delta} = \mathit{\Sigma} \times \{0,1\}$ obtained inductively from the $(r-1)$-pwA $\mathcal{A}_q$ which is defined below together with its equivalence relation $\sim_q$. Notice that $k \cdot \mathcal{B}_q$ is obtained from $\mathcal{B}_q$ by multiplying its input weights by k.

Let us define the $(r-1)$-pwA $\mathcal{A}_q = (P', \mu')$ over $\mathit{\Delta}$ as well as $\sim_q \subseteq P' \times P'$. Suppose $q = (a, \pi)$ and $\mathrm{pr}_2(\pi) = p_1\,\hat{p}_1 \cdots p_N\,\hat{p}_N$. The behavior of $\mathcal{A}_q$ is split into $N+2$ phases. In phase 0, it scans the input word from left to right until it finds the (unique) letter of the form $(a, 1)$ with $a \in \mathit{\Sigma}$. At that position, call it i, $\mathcal{A}_q$ enters state $(p_1, 1)$ (the second component indicating the current phase). All these transitions are performed with weight $\mathbf{1}$. Then, $\mathcal{A}_q$ starts simulating $\mathcal{A}$, considering pebble r at position i. Back at position i in state $(\hat{p}_1, 1)$, weight-$\mathbf{1}$ transitions will allow $\mathcal{A}_q$ to enter the next phase, starting in $(p_2, 2)$ and again considering pebble r at position i. The simulation of $\mathcal{A}$ ends when $(\hat{p}_N, N)$ is reached in position i. In the final phase $(N+1)$ weight-$\mathbf{1}$ transitions guides $\mathcal{A}_q$ to the end of the tape where it stops. The relation $\sim_q$ will consider states (p, j) and (p', j') equivalent iff $p \sim p'$, *i.e.*, it ignores the phase number. This explains the purpose of the equivalence relation: in order for the automaton $\mathcal{A}_q$ to simulate

the dashed part of the run of $\mathcal{A}$, we use phase numbers, so that, during the simulation two different states (p, j) and (p, j') of $\mathcal{A}_q$ may correspond to the same original state p of $\mathcal{A}$. Now, only simple runs are considered to compute $[\![\mathcal{A}]\!]$. Therefore, for the simulation to be faithful, we want to rule out runs of $\mathcal{A}_q$ containing two configurations which only differ by the phase number, that is, containing two $\sim_q$-equivalent configurations, which is done by keeping only $\sim_q$-simple runs. A run of $\mathcal{A}_q$ is illustrated in Figure 3(b) (with $N = 2$). Note that, if $N = 0$, then $\mathcal{A}_q$ simply scans the word from left to right, outputting weight $\mathbf{1}$. Finally, we sketch the proof of (3).

We proceed by induction on the structure of the formula and suppose that a valuation of free variables is given in terms of pebbles that are already placed on the word and cannot be lifted. Disjunction, conjunction, and first-order quantifications are easy to simulate. To evaluate $[N\text{-TC}^{<}_{xy}\varphi](z, t)$ for some formula $\varphi(x, y)$, we either evaluate $\varphi(x, y)$, with pebbles being placed on z and t such that $z \leq t \leq z + N$, or choose non-deterministically (and with weight $\mathbf{1}$) positions $z = z_0 < z_1 < \cdots < z_{n-1} < z_n = t$ with $n \geq 2$, using two additional pebbles, 2 and 1. We drop pebble 2 on position z_0 and pebble 1 on some guessed position z_1 with $z_0 < z_1 \leq \min(t, z_0 + N)$. We then run the subroutine to evaluate $\varphi(z_0, z_1)$. Next we move to position z_1 and lift pebble 1. We move left to position z_0 remembering the distance $z_1 - z_0$. We lift pebble 2 and move right to z_1 using the stored distance $z_1 - z_0$. We drop pebble 2 on z_1 and iterate this procedure until t is reached. □

Conclusion and perspectives

We have introduced pebble weighted automata and characterized their expressive power in terms of first-order logic with a bounded transitive closure operator. It follows that satisfiability is decidable over commutative positive semirings. Here, a sentence $\varphi \in \text{FO+BTC}^{<}$ is said *satisfiable* if there is a word $w \in \Sigma^+$ such that $[\![\varphi]\!](w) \neq \mathbf{0}$. From Theorem 11, satisfiability reduces to non-emptiness of the support of a series recognized by a pwA. For positive semirings, the latter problem, in turn, can be reduced to the decidable emptiness problem for classical pebble automata over the Boolean semiring. We leave it as an open problem to determine for which semirings the satisfiability problem is decidable.

Unbounded transitive closure. We do not know if allowing unbounded steps in the transitive closure leads beyond the power of (weak) pebble automata. We already know that allowing unbounded steps is harmless for bMSO-step formulas. It is also easy to show that such an unbounded transitive closure can be captured with *strong* pebble automata, *i.e.*, that can lift the last dropped pebble even when not scanning its position. Therefore, we aim at studying the expressive power of strong pebble automata and unbounded transitive closure.

Tree-walking automata. Our results are not only of theoretical interest. They also lay the basis for quantitative extensions of database query languages such as XPath, and may provide tracks to evaluate quantitative aspects of the structure

of XML documents. The framework of weighted tree (walking) automata [1,11] 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 tree-walking automata [3,14,2] is also well-suited in this context. For these reasons, we would like to adapt our results to tree languages.

We thank the anonymous referees for their valuable comments.

References

1. J. Berstel and C. Reutenauer. Recognizable formal power series on trees. *TCS*, 18(2):115 – 148, 1982.
2. M. Bojańczyk. Tree-walking automata. In *LATA'08*, volume 5196 of *LNCS*, pages 1–17. Springer, 2008.
3. M. Bojańczyk, M. Samuelides, T. Schwentick, and L. Segoufin. Expressive power of pebble automata. In *ICALP'06*, volume 4051 of *LNCS*, pages 157–168. Springer, 2006.
4. B. Bollig, P. Gastin, B. Monmege, and M. Zeitoun. Pebble weighted automata and transitive closure logics. Available at: http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/rapports.
5. J. R. Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6:66–92, 1960.
6. M. Droste and P. Gastin. Weighted automata and weighted logics. *Theoretical Computer Science*, 380(1-2):69–86, 2007. Special issue of ICALP'05.
7. M. Droste, W. Kuich, and H. Vogler, editors. *Handbook of Weighted Automata*. EATCS Monographs in Theoret. Comput. Sci. Springer, 2009.
8. C. C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Amer. Math. Soc.*, 98:21–52, 1961.
9. J. Engelfriet and H. J. Hoogeboom. Tree-walking pebble automata. In *Jewels Are Forever, Contributions to Theoretical Computer Science in Honor of Arto Salomaa*, pages 72–83. Springer, 1999.
10. J. Engelfriet and H. J. Hoogeboom. Automata with nested pebbles capture first-order logic with transitive closure. *Log. Meth. in Comput. Sci.*, 3, 2007.
11. Z. Fülöp and L. Muzamel. Weighted Tree-Walking Automata. *Acta Cybernetica*, 19(2):275 – 293, 2009.
12. N. Globerman and D. Harel. Complexity results for two-way and multi-pebble automata and their logics. *Theoretical Computer Science*, 169(2):161–184, 1996.
13. F. Neven and T. Schwentick. On the power of tree-walking automata. In Springer, editor, *ICALP*, volume 1853 of *LNCS*, pages 547–560, 2000.
14. M. Samuelides and L. Segoufin. Complexity of pebble tree-walking automata. In *FCT*, pages 458–469, 2007.
15. M. P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4:245–270, 1961.
16. B. ten Cate and L. Segoufin. Transitive closure logic, nested tree walking automata, and XPath. *J. ACM*, 2010. To appear. Short version in PODS'08.
17. W. Thomas. Classifying regular events in symbolic logic. *Journal of Computer and System Sciences*, 25(3):360–376, 1982.
18. B. A. Trakhtenbrot. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR*, 149:326–329, 1961. In Russian.