

A Unifying Survey on Weighted Logics and Weighted Automata

Paul Gastin, Benjamin Monmege

▶ **To cite this version:**

Paul Gastin, Benjamin Monmege. A Unifying Survey on Weighted Logics and Weighted Automata: Core Weighted Logic: Minimal and Versatile Specification of Quantitative Properties. Soft Computing, 2018, 22 (4), pp.1047-1065. 10.1007/s00500-015-1952-6 . hal-01130216v2

HAL Id: hal-01130216
https://hal.science/hal-01130216v2

Submitted on 12 Feb 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright

Noname manuscript No.
(will be inserted by the editor)

A Unifying Survey on Weighted Logics and Weighted Automata

Core Weighted Logic: Minimal and Versatile Specification of Quantitative Properties

Paul Gastin · Benjamin Monmege

Received: date / Accepted: date

Abstract Logical formalisms equivalent to weighted automata have been the topic of numerous research papers in the recent years. It started with the seminal result by Droste and Gastin on weighted logics over semirings for words. It has been extended in two dimensions by many authors. First, the weight domain has been extended to valuation monoids, valuation structures, etc., to capture more quantitative properties. Along another dimension, different structures such as ranked or unranked trees, nested words, Mazurkiewicz traces, etc., have been considered. The long and involved proofs of equivalences in all these papers are implicitly based on the same core arguments.

This article provides a meta-theorem which unifies these different approaches. Towards this, we first revisit weighted automata by defining a new semantics for them in two phases—an abstract semantics based on multisets of weight structures (independent of particular weight domains) followed by a concrete semantics. Then, we introduce a core weighted logic with a minimal number of features and a simplified syntax, and lift the new semantics to this logic. We show at the level of the abstract semantics that weighted automata and core weighted logic have the same expressive power. Finally, we show how previous results can be recovered from our result by logical reasoning.

In this paper, we prove the meta-theorem for words, ranked and unranked trees, showing the robustness of our approach.

Part of the research leading to these results was achieved when the second author was at Université libre de Bruxelles (Belgium), and has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under Grant Agreement n601148 (CASSTING).

P. Gastin
LSV, ENS Cachan, CNRS, Université Paris-Saclay, 94235 Cachan, France
paul.gastin@lsv.ens-cachan.fr

B. Monmege
Aix-Marseille Université, LIF, CNRS, France
benjamin.monmege@lif.univ-mrs.fr

1 Introduction

Weighted automata are a well-studied formalism modelling quantitative behaviours. Introduced by Schützenberger in [20], they have been applied in many areas such as image compression [1], natural language processing [17], verification and synthesis of programs [4], etc. In the last years, high-level specification formalisms of quantitative properties have received increasing interest. Among other successes, the connection between monadic second-order logic (MSO) and finite automata established by Büchi, Elgot and Trakhtenbrot [3,15,22], has been extended to the weighted setting.

There have been many attempts to find a suitable extension of MSO to describe quantitative properties which captures the expressive power of weighted automata. The considered variants differ with respect to the structures (words, ranked or unranked trees, nested words, etc.) and the weight domains (semirings, valuation monoids, valuation structures, multi-operator monoids, etc.). This article aims at revisiting the link between weighted logics and weighted automata in a *uniform* manner with regards of these two dimensions.

Our main contribution is to consider a new fragment of weighted logics containing a minimal set of features. In order to simplify the uniformity with respect to the structures, we syntactically separate a Boolean fragment from the weighted part: only the syntax of Boolean formulæ depends on the structures considered.

Then, we clearly separate a small fragment able to define step functions—that we call step formulæ—from the more general weighted logic. Because of the minimal set of features that it displays, we call our logic *core* weighted monadic second-order logic. This separation into three distinct layers, more or less clear in previous works, is designed both to clarify the subsequent study of the expressive power, and to simplify the use of the weighted logic.

Towards defining the semantics of this new logic, we first revisit weighted automata by defining an alternative semantics, then lifting it to formulæ. This is done in two phases. First, an abstract semantics associates with a structure a multiset of *weight labelled structures*. E.g., in the case of words, a weighted automaton/formula will map every word to a multiset of weight words. In the setting of trees, every tree is associated with a multiset of weight trees (of the same shape as the original tree). This abstract semantics is fully uninterpreted and, hence, does not depend on any algebraic structure over the set of weights considered. This semantics is in the spirit of a transducer. It has already been used in similar contexts: in [16] with an operator $H(\omega)$ which relabels trees with operations taken from a multi-operator monoid, in [19] with a weight assignment logic over infinite words, in [12,19] with Nivat theorems for weighted automata over various structures. In a second phase, a concrete semantics is given, by means of an *aggregation* operator taking the abstract semantics and aggregating every multiset of weight structures to a single value (in a possibly different weight domain). For instance, the usual semantics of weighted automata over semirings can be recovered by mapping every weight word to the product of its weights, and merging the multiset with the addition of the semiring.

Separating the semantics in two successive phases, both for weighted automata and logics, allows us to revisit the original proof of expressive equivalence of [6] in the abstract semantics. This result has been extended to various weight domains and/or structures (see below). The proof of equivalence in all these works are based on the same core argument which relates runs of automata with the evaluation of formulæ. Inspired by the above similarities, our choice of the abstract multiset semantics manifests this core argument. Because the abstract semantics is fully uninterpreted, no additional hypotheses on the weight domain is required to prove the equivalence. We then apply the aggregation operator to obtain a concrete equivalence between weighted automata and our core weighted logic.

Our last contribution is to show, by means of purely logical reasoning, that our new fragment of core weighted logic is expressively equivalent to the logics proposed in the previous works. Over finite words, this allows us to recover the results over semirings [6], (product) valuation monoids [10] and (product) valuation structures [11]. Valuation monoids replace the product operation of the semiring by a lenient valuation operation, making possible to consider discounted sums, average or more evolved combination of sequences of weights. Valuation structures finally also replace the sum by a more general evaluation operator, for instance ratios of several weights computed simultaneously. As an example, it is then possible to compute the ratio between rewards and costs, or the efficiency of use of a primary resource under some upper bound constraint on a secondary resource. Our unifying proof gives new insights on the additional hypotheses (commutativity, distributivity, etc) over the weight domains used in these works.

After studying in full details the case of finite words, we illustrate the uniformity of the method with respect to structures, by considering ranked and unranked trees. Once again, our study revisits existing works over semirings [13,14], (product) valuation monoids [7], and also multi-operator monoids [16]. The syntax of the logic in the case of multi-operator monoids is different from the other logics. The proof techniques used to show equivalence of the two formalisms are nevertheless very close to the original ones for semirings.

2 Preliminaries

In this section, we recall the basic notions of semirings and multisets that we will need in the following.

A *semiring* is a tuple $(S, +, \times, 0, 1)$ where $+$ and $\times$ are two binary operations, and 0 and 1 are two elements of S, verifying the following:

- $(S, +, 0)$ is a commutative monoid;
- $(S, \times, 1)$ is a monoid;
- $\times$ distributes over $+$;
- 0 is a zero, i.e., $s \times 0 = 0 \times s = 0$ for all $s \in S$.

Classical examples of semirings are given by $(\mathbb{Z}, +, \times, 0, 1)$, $(\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$, $(\mathbb{R} \cup \{+\infty\}, \min, +, +\infty, 0)$, $(\mathfrak{P}(\Sigma^\star), \cup, \cdot, \emptyset, \{\varepsilon\})$ where Σ is a finite alphabet and $\cdot$ denotes the concatenation. More examples can be found in, e.g., [9,18].

Multisets generalise sets by allowing several occurrences of the same element in a multiset. For instance, $A = \{\{a, a, b, b, b, c\}\}$ is a finite multiset consisting of two occurrences of a, three occurrences of b and one occurrence of c. Formally, a multiset A over a set X is a (multiplicity) function $A \colon X \to \mathbb{N}$ which gives for each $x \in X$ the number $A(x)$ of occurrences of x in A.

The size of a multiset is denoted by $|A| = \sum_{x \in X} A(x)$, and we denote by $x \in A$ the fact that $A(x) \neq 0$.

The disjoint union of multisets, defined by the pointwise sum $(A \uplus B)(x) = A(x) + B(x)$ for all $x \in X$, is a commutative and associative operation on multisets with the empty multiset, denoted by $\emptyset$, as neutral element. A multiset A is *finite* if its *support* $\mathrm{supp}(A) = \{x \in X \mid A(x) \neq 0\}$ is finite. We denote by $\mathbb{N}\langle X \rangle$ the set of finite multisets over X. Clearly, $(\mathbb{N}\langle X \rangle, \uplus, \emptyset)$ is a commutative monoid.

Every operation defined on the set X can be lifted to multisets over X. For instance, if $\diamond$ is a binary operation over X then it is lifted as a binary operation over $\mathbb{N}\langle X \rangle$ by $A \diamond B = \{\{a \diamond b \mid a \in A, b \in B\}\}$: formally, for all $x \in X$, we let

$$(A \diamond B)(x) = \sum_{\substack{(x_1,x_2) \in X^2 \\ x = x_1 \diamond x_2}} A(x_1) \times B(x_2)\,.$$

For instance, if $X = \mathbb{N}$ and $\diamond = \times$ is the usual product, we get $\{\{1,2,2\}\} \times \{\{3,6\}\} = \{\{3,6,6,6,12,12\}\}$.

Similarly, a unary operation $f\colon X \to Y$ is lifted as a unary operation from $\mathbb{N}\langle X \rangle$ to $\mathbb{N}\langle Y \rangle$ by $f(A) = \{\{f(a) \mid a \in A\}\}$: formally, for all $y \in Y$, we let

$$\big(f(A)\big)(y) = \sum_{\substack{x \in X \\ f(x) = y}} A(x)\,.$$

For instance, if $X = \mathbb{Z}$, $Y = \mathbb{N}$, $f(x) = x^2$ and $A = \{\{-2,-1,-1,0,1,3\}\}$ then $f(A) = \{\{0,1,1,1,4,9\}\}$.

When $(Y, +, 0)$ is a *commutative* monoid and $B \in \mathbb{N}\langle Y \rangle$ then we let

$$\sum B = \sum_{y \in Y} \big(\underbrace{y + \cdots + y}_{B(y) \text{ times}}\big)$$

be the sum of the elements in B (with multiplicities), where $\sum$ denotes the addition in the monoid. For instance, with notations of the previous example, we have $\sum f(A) = 16$. Note that $\sum \emptyset = 0$.

Multisets over a monoid form a semiring. More precisely, let $(M, \times, \mathbf{1})$ be a monoid. Then, lifting $\times$ to $\mathbb{N}\langle M \rangle$ yields a monoid with $\{\{\mathbf{1}\}\}$ as neutral element. Moreover, $\times$ distributes over $\uplus$ and $\emptyset$ is a zero for $\times$. Therefore, $(\mathbb{N}\langle M \rangle, \uplus, \times, \emptyset, \{\{\mathbf{1}\}\})$ is a semiring.

In this paper, we will mainly use the free monoid $(R^\star, \cdot, \varepsilon)$ over a finite set R of weights, and the induced semiring $(\mathbb{N}\langle R^\star \rangle, \uplus, \cdot, \emptyset, \{\{\varepsilon\}\})$.

3 Weighted automata

Weighted automata are a well-studied model of computation of formal power series introduced by Schützenberger in [20]. Originally introduced over integers, they have afterwards been extended to more general weighted domains. We recall here the syntax of weighted automata, and then give the semantics in various weight structures, before revisiting this semantics with our new *abstract semantics*.

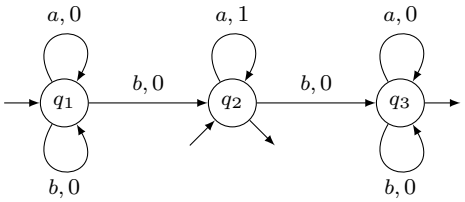


Fig. 1 A $\{0,1\}$-weighted automaton over $\{a, b\}$

3.1 Syntax of weighted automata

Formally, a *weighted automaton* is simply a classical finite state automaton over a finite alphabet Σ, where every transition is equipped with a weight taken from a set R. Precisely, an R-weighted automaton over Σ is a tuple $\mathcal{A} = (Q, \Delta, \mathsf{wgt}, I, F)$ with Q a non-empty finite set of states, $\Delta \subseteq Q \times \Sigma \times Q$ the set of transitions, $\mathsf{wgt}\colon \Delta \to R$ associating a weight to every transition and $I, F \subseteq Q$ are respectively initial and final states.

An example of a $\{0,1\}$-weighted automaton over alphabet $\Sigma = \{a, b\}$ is given in Fig. 1. It has three states q_1, q_2 and q_3, the first two being initial and the last two being final. The weights of transitions are depicted on the right of their labels: the only transition of weight different from 0, has weight 1 and is the loop on state q_2.

A *run* of $\mathcal{A}$ over a word $w = a_1 \cdots a_n \in \Sigma^+$ is a sequence of transitions $\rho = \delta_1 \cdots \delta_n \in \Delta^+$ with $\delta_i = (q_i, a_i, q'_i)$ for all $1 \leqslant i \leqslant n$ such that $q'_i = q_{i+1}$ for all $1 \leqslant i \leqslant n-1$. The run ρ is *accepting* if it starts in an initial state ($q_1 \in I$), and ends in a final state ($q'_n \in F$). For the weighted automaton of Fig. 1, the sequence $(q_1, a, q_1), (q_1, b, q_2), (q_2, a, q_2), (q_2, a, q_2), (q_2, b, q_3)$ is an accepting run over the word *abaab*.

3.2 Semantics over semirings

When the set R of weights is a subset of a semiring $(S, +, \times, 0, 1)$, weighted automata are called *weighted automata over semirings*. The usual quantitative semantics on a word $w \in \Sigma^+$ is obtained as follows. First, we compute the value of a run $\rho = \delta_1 \cdots \delta_n$ on w using the multiplication in the semiring: $\mathsf{Val}(\rho) = \mathsf{wgt}(\delta_1) \times \cdots \times \mathsf{wgt}(\delta_n)$. Then, the semantics of $\mathcal{A}$ on w is obtained by taking the sum in the semiring of the

values of accepting runs:

$$[\![\mathcal{A}]\!](w) = \sum_{\rho} \mathsf{Val}(\rho) \tag{1}$$

where ρ ranges over all accepting runs over w.

For instance, consider the $\{0,1\}$-weighted automaton of Fig. 1 over the semiring $(\mathbb{N}\cup\{-\infty\}, \max, +, -\infty, 0)$. Each run uses the loop over the state q_2 to *count* the length of a block of a's in the word. Hence, we can check that the semantics of the automaton maps every word w to the maximal length of a block of a's in w. For instance, $[\![\mathcal{A}]\!](abaaba) = 2$.

3.3 Semantics over valuation monoids

Several extensions of weighted automata over semirings have been proposed. We start here with valuation monoids [10] and we will discuss valuation structures [11] in the next subsection.

The motivation comes from the fact that, for certain applications, the value of a run cannot be computed with a product in a semiring. For instance, the value of a run may be a discounted sum of the weights of transitions, or the average of the weights. An idea is to replace the product of the semiring by a valuation function—without the conditions that the product of a semiring must verify—which computes the value of a sequence of weights.

A *valuation monoid* is a tuple $(S, +, 0, \mathsf{Val})$ where $(S, +, 0)$ is a commutative monoid and $\mathsf{Val}\colon S^\star \to S$ is a function, called valuation operator, that maps each finite sequence of elements of S to an element of S. In the original work of [10], the valuation operator is only defined on nonempty sequences from S^+ but we may extend it by setting $\mathsf{Val}(\varepsilon)$ to an arbitrary value. Also, in the definition of [10], Val must satisfy some additional properties, such as $\mathsf{Val}(s) = s$ for all $s \in S$, and for $s_1 \cdots s_n \in S^+$ we have $\mathsf{Val}(s_1 \cdots s_n) = 0$ whenever $s_i = 0$ for some i. However, these additional conditions are not necessary to define the semantics of weighted automata and to obtain the equivalence between weighted automata and our weighted logic that will be presented in Section 4.

For instance, we may use as valuation a discounted sum $\mathsf{Val} = \mathsf{Disc}_\lambda$ defined for $\lambda \in (0,1)$ and a sequence of real numbers by $\mathsf{Disc}_\lambda(s_1 \cdots s_n) = s_1 + \lambda s_2 + \cdots + \lambda^{n-1} s_n$. It is also possible to consider as valuation the average $\mathsf{Val} = \mathsf{Avg}$ of a sequence of real numbers, given by $\mathsf{Avg}(s_1 \cdots s_n) = \frac{1}{n}(s_1 + \cdots + s_n)$. In those two cases, $+$ denotes the usual addition over real numbers, and we may consider the monoid operation to be max or min for instance, allowing us to compute optimal discounted or average costs. We may also use the sum $+$ which is not allowed by [10] since it would not fulfil the additional conditions given above.

We now give the semantics of R-weighted automata with $R \subseteq S$, and $(S, +, 0, \mathsf{Val})$ a valuation monoid: in the following, such automata are called *weighted automata over valuation monoids*. Let $\mathcal{A}$ be an R-weighted automaton and $w = a_1 \cdots a_n \in \Sigma^+$. As expected, the value of a run $\rho = \delta_1 \cdots \delta_n$ on w is computed with the valuation operator $\mathsf{Val}(\rho) = \mathsf{Val}(\mathsf{wgt}(\delta_1) \cdots \mathsf{wgt}(\delta_n))$. Then, the semantics of $\mathcal{A}$ over w is defined by (1) as before.

As an example, consider again the $\{0,1\}$-weighted automaton of Fig. 1 and the valuation monoid $(\mathbb{R} \cup \{-\infty\}, \max, \mathsf{Disc}_\lambda, -\infty)$. Then, the semantics of the automaton on a word $w \in \Sigma^+$ is the maximal score of the blocks of a's. The score of a block is computed as $\lambda^{i-1}\frac{\lambda^\ell - 1}{\lambda - 1}$ where i is the position of the first letter of the block and ℓ is its length.

As another example, consider the valuation monoid $(\mathbb{N} \cup \{\infty\}, \min, \mathsf{Sq}, \infty)$ where the valuation[1] Sq maps a sequence $s_1 \cdots s_n$ to the square of the number of s_i's which are equal to 1. Then, the semantics of the weighted automaton of Fig. 1 is the minimum of the squares of the lengths of a-blocks.

3.4 Semantics over valuation structures

In order to get even more flexibility in the computation of the semantics, [11] proposed to replace the sum of the run values which is used in (1) by a more general evaluator function F. In addition, [11] allows the set of weights R used in the automaton and for the value of runs to be different from the final set of weights S computed by the evaluator function.

Formally, a *valuation structure*[2] is a tuple (U, Val, S, F) where U and S are two sets, $\mathsf{Val}\colon U^\star \to U$ is a valuation operator, and $F\colon \mathbb{N}\langle U\rangle \to S$ is an evaluator function mapping a finite multiset of weights in U to a single weight in S.

Given an R-weighted automaton $\mathcal{A}$ with $R \subseteq U$ (that we call *weighted automata over a valuation structure*), we compute the value of a run as in the case of valuation monoids: $\mathsf{Val}(\rho) = \mathsf{Val}(\mathsf{wgt}(\delta_1) \cdots \mathsf{wgt}(\delta_n))$ when $\rho = \delta_1 \cdots \delta_n$. Then, the semantics of $\mathcal{A}$ over a word $w \in \Sigma^+$ is defined in two steps:

$$[\![\mathcal{A}]\!](w) = F(\{\!\{\mathsf{Val}(\rho) \mid \rho \text{ accepting run over } w\}\!\})\,.$$

First, Val transforms the set of accepting runs over w in a multiset of weights, which is then transformed in the final semantics with the evaluator function.

[1] This valuation does not satisfy $\mathsf{Sq}(s) = s$.

[2] We do not require that $\mathsf{Val}(r) = r$ for all $r \in U$ as in [11].

For instance, we may choose $U = \mathbb{Z} \times \mathbb{N}$ with the valuation

$$\mathsf{Val}\big((x_1, y_1), \dots, (x_n, y_n)\big) = \left(\sum_{i=1}^{n} x_i, \sum_{i=1}^{n} y_i\right) .$$

Then, choosing $S = \mathbb{Q} \cup \{+\infty\}$ in the valuation structure, we may compute the average of the ratios between rewards and non-negative costs with the evaluator function defined by $F(\emptyset) = 0$ and

$$F(A) = \frac{1}{|A|} \sum_{(x,y)\in A} \frac{x}{y}$$

for $A \in \mathbb{N}\langle U \rangle \setminus \{\emptyset\}$, with $\frac{x}{0} = +\infty$ by convention. We may change the evaluator function to

$$F(A) = \max_{(x,y)\in A} \frac{x}{y}$$

to compute the maximal ratio between rewards and costs (with $-\infty$ added to S and $\max(\emptyset) = -\infty$). We may further change the evaluator function to

$$F_p(A) = \max\{x \mid (x, y) \in A \wedge y \leqslant p\}$$

with p any positive number, to compute the best reward under the given cost upper bound.

As a last example, consider the valuation structure $(\mathbb{N}, \mathsf{Val}, \mathbb{Q}, F)$ where the valuation Val sums the sequence of weights, and the evaluator function $F = \mathrm{Avg}$ computes the average of the multiset as $\mathrm{Avg}(\emptyset) = 0$ and

$$\mathrm{Avg}(A) = \frac{1}{|A|} \sum A$$

for $A \neq \emptyset$. Then, the $\{0, 1\}$-weighted automaton of Fig. 1 computes the average length of the a-blocks of the word.

3.5 Abstract semantics

After recalling some existing semantics of weighted automata, we now present a unifying framework, mostly based on weighted automata over semirings, that will still be able to recover as special cases all previous semantics.

The main idea is to split the semantics into two phases, as already done for valuation structures, but the separation takes place somewhere else in order to obtain an intermediary structure with even more information. Instead of multisets of values in R, we will use multisets of sequences in $R^\star$. Interestingly, since $(R^\star, \cdot, \varepsilon)$ is a monoid, we have seen in Section 2 that $(\mathbb{N}\langle R^\star \rangle, \uplus, \cdot, \emptyset, \{\{\varepsilon\}\})$ is a semiring. This remark will be useful in the following. As usual, we will not indicate the operation $\cdot$ in expressions over $\mathbb{N}\langle R^\star \rangle$.

The *abstract* semantics of an R-weighted automaton $\mathcal{A}$ is simply its semantics in the semiring $\mathbb{N}\langle R^\star \rangle$ when we identify the weights $r \in R$ occurring in $\mathcal{A}$ with the singleton multisets $\{\{r\}\}$. Computing in the semiring $\mathbb{N}\langle R^\star \rangle$, the value $\mathsf{Val}(\rho)$ of a run $\rho = \delta_1 \cdots \delta_n$ over some word $w \in \Sigma^+$ is

$$\{\{\mathsf{wgt}(\delta_1)\}\} \cdots \{\{\mathsf{wgt}(\delta_n)\}\} = \{\{\mathsf{wgt}(\delta_1) \cdots \mathsf{wgt}(\delta_n)\}\} .$$

The abstract semantics, denoted by $\{\!|\mathcal{A}|\!\}(w)$, is obtained by summing (i.e., taking the disjoint union) over all accepting runs over w:

$$\{\!|\mathcal{A}|\!\}(w) = \biguplus_{\rho = \delta_1 \cdots \delta_n} \{\{\mathsf{wgt}(\delta_1) \cdots \mathsf{wgt}(\delta_n)\}\} .$$

Notice that $\{\!|\mathcal{A}|\!\}(w)$ is indeed a finite multiset, since $\mathcal{A}$ admits only finitely many accepting runs over w.

Example 1 Consider the $\{0, 1\}$-weighted automaton of Fig. 1. Its abstract semantics over the word *abaab* is $\{\!|\mathcal{A}|\!\}(abaab) = \{\{00000, 00110, 10000\}\}$.

Remark 2 Our abstract semantics proposes another level of abstraction compared to the one obtained with Nivat theorems for weighted automata, e.g., in [12,19]. In these Nivat-like characterisations, the behaviour of a weighted automaton is described by its *set* of runs, whereas we use the less concrete description by its *multiset* of weight structures (here, sequences of weights). Our more abstract semantics is sufficient for our later discussion. Also, it allows us to keep an underlying semiring that permits to reuse initial results for weighted automata over semirings.

From this abstract semantics, we compute the concrete semantics by aggregating the multiset of weight sequences into a single weight, possibly lying in a different weight domain S. We do so with an aggregation operator $\mathsf{aggr} \colon \mathbb{N}\langle R^\star \rangle \to S$. We obtain the *concrete* semantics $[\![\mathcal{A}]\!](w) = \mathsf{aggr}(\{\!|\mathcal{A}|\!\}(w))$. The remaining of this section is devoted to some examples, covering as special cases the various semantics seen in Section 2.

Example 3 If R is a subset of a semiring $(S, +, \times, 0, 1)$, we recover the usual semantics by considering the aggregation operator

$$\mathsf{aggr}_{\mathtt{sr}}(A) = \sum \prod(A)$$

where $\prod \colon S^\star \to S$ maps every sequence $r_1 \cdots r_n \in S^\star$ to the product $r_1 \times \cdots \times r_n$, and is lifted over finite multisets as explained in the preliminaries.

Example 4 The previous example can be generalised to the case where R is a subset of a valuation monoid $(S, +, 0, \mathsf{Val})$. Using the lift of $\mathsf{Val} \colon S^\star \to S$ to finite multisets, the aggregation operation is now:

$$\mathsf{aggr}_{\mathtt{vm}}(A) = \sum \mathsf{Val}(A) .$$

Example 5 We can also recover the semantics in a valuation structure (U, Val, S, F) by considering the aggregation defined by

$$\mathsf{aggr}_{\mathsf{vs}}(A) = F(\mathsf{Val}(A))\,.$$

Notice that our framework is a priori more powerful than valuation structures. Indeed, in our context, we do not decouple the valuation and the evaluator operations. Instead, we keep a richer information, namely the multiset of weight sequences, allowing more powerful aggregations.

For instance, imagine that, on the weighted automaton of Fig. 1, we want to compute the discounted sum of the length of blocks of a's. E.g., if $\lambda \in (0,1)$ is the discount factor, we would like to associate to the word $abaababaaa$ the weight $1+\lambda\cdot 2+\lambda^2\cdot 1+\lambda^3\cdot 3$. For that purpose, consider R to be $\{0,1\}$, S to be $\mathbb{R}$, and the aggregation aggr to be defined on a multiset $A \in \mathbb{N}\langle R^\star\rangle$ by

$$\mathsf{aggr}(A) = \sum_{j=1}^{k} \lambda^{j-1} n_j$$

if $A = \{\{0^{m_1}1^{n_1}0^{p_1}, 0^{m_2}1^{n_2}0^{p_2}, \dots, 0^{m_k}1^{n_k}0^{p_k}\}\}$ with $m_1 < m_2 < \cdots < m_k$, and $\mathsf{aggr}(A) = 0$ otherwise. This aggregation cannot be obtained *directly* in the setting of valuation structures, since the value of a run should now take into account the discount factor, based on the number of blocks of a's previously seen: however, this number cannot be obtained by directly looking at the unique current run.

4 Core weighted monadic second-order logic

We now turn to the description of a new weighted logic, that will be equivalent to weighted automata. Most existing works start with the definition of a very general logic, and then introduce restrictions to match the expressive power of weighted automata. We take the opposite approach by defining a very basic weighted logic, yet powerful enough to be expressively equivalent to weighted automata. Our logic has three layers: the Boolean fragment which is the classical MSO logic over words, a *step* weighted fragment (step-wMSO) defining step functions (i.e., piecewise constant functions with a finite number of pieces), and the *core* weighted logic (core-wMSO) which has the full expressive power of weighted automata. We will show in Section 5 that core-wMSO is a fragment of the (full) weighted MSO logic (wMSO) defined in [6]. Considering a Boolean fragment inside a weighted logic was originally done in [2] and followed in many articles, see, e.g., [10,16].

$$\begin{aligned}
\varphi &::= \top \mid P_a(x) \mid x \leqslant y \mid x \in X \mid \neg\varphi \mid \varphi \wedge \varphi \mid \forall x\varphi \mid \forall X\varphi && \text{(MSO)}\\
\Psi &::= r \mid \varphi\,?\,\Psi : \Psi && \text{(step-wMSO)}\\
\Phi &::= \mathbf{0} \mid \varphi\,?\,\Phi : \Phi \mid \Phi + \Phi \mid \textstyle\sum_x \Phi \mid \sum_X \Phi \mid \prod_x \Psi && \text{(core-wMSO)}
\end{aligned}$$

with $a \in \Sigma$, $r \in R$, x, y first-order variables and X a second-order variable.

Table 1 Syntax of the core weighted logic core-wMSO(Σ, R).

$$\begin{aligned}
\{\!|r|\!\}_V(w,\sigma) &= \{\{r\}\}\\
\{\!|\varphi\,?\,\Psi_1 : \Psi_2|\!\}_V(w,\sigma) &= \begin{cases} \{\!|\Psi_1|\!\}_V(w,\sigma) & \text{if } w,\sigma \models \varphi \\ \{\!|\Psi_2|\!\}_V(w,\sigma) & \text{otherwise} \end{cases}
\end{aligned}$$

Table 2 Semantics of step-wMSO

4.1 Syntax and semantics

We fix a finite alphabet Σ and we consider a set $\mathcal{V}$ of first-order $(x, y, \dots)$ and second-order $(X, Y, \dots)$ variables. The syntax of MSO(Σ) is given in Table 1 (MSO). We may freely use classical macros such as $\bot = \neg\top$, $\varphi \vee \varphi' = \neg(\neg\varphi \wedge \neg\varphi')$, or $\exists x\,\varphi = \neg\forall x\,\neg\varphi$.

We do not recall the semantics of an MSO formula φ which can be defined inductively as usual. We denote by $\mathsf{free}(\varphi)$ the set of *free* variables in φ. Given a word $w \in \Sigma^+$ and a valuation σ from a set of variables $V \supseteq \mathsf{free}(\varphi)$ to the (sets of) positions of w, we write $w, \sigma \models \varphi$ when (w, σ) satisfies φ.

We will use the classical encoding of a pair (w, σ) consisting of a word $w \in \Sigma^+$ and a valuation σ on V as a word $\overline{w}$ over the extended alphabet $\Sigma_V = \Sigma \times \{0,1\}^V$. A word $\overline{w}$ in Σ_V^+ is said to be valid if for every first-order variable $x \in V$, the projection of $\overline{w}$ on the component indexed by x belongs to $0^\star 1 0^\star$. In the following, we identify a valid word $\overline{w}$ with its encoded pair (w, σ).

The syntax for step-wMSO(Σ, R) formulæ is given in Table 1 (step-wMSO). It uses weights from a set R and is based on the if-then-else operator $\varphi\,?\,\Psi_1 : \Psi_2$ where the condition is an MSO formula φ.

The semantics of such formulæ is given in terms of *finite* multisets of weights. More precisely, for a set V of variables that contains the set $\mathsf{free}(\Psi)$ of free variables of Ψ, we let $\{\!|\cdot|\!\}_V : \Sigma_V^+ \to \mathbb{N}\langle R\rangle$ be defined by $\{\!|\Psi|\!\}(\overline{w}) = \emptyset$ if $\overline{w}$ is not valid, and if $\overline{w}$ is the valid encoding of (w, σ) then the semantics is given in Table 2. Notice that $\{\!|\Psi|\!\}(\overline{w})$ is either the empty multiset (if $\overline{w}$ is not a valid encoding) or is a singleton multiset. We simply write $\{\!|\Psi|\!\}$ for $\{\!|\Psi|\!\}_{\mathsf{free}(\Psi)}$.

Finally, the syntax of the core-wMSO(Σ, R) logic is given in Table 1 (core-wMSO). It uses a special constant $\mathbf{0}$. In addition to the if-then-else operator, there are three sum operators whose semantics will be defined using the disjoint union (sum) of multisets, and a

$$\{\!|\mathbf{0}|\!\}_V(w,\sigma) = \emptyset$$

$$\{\!|\varphi\,?\,\Phi_1 : \Phi_2|\!\}_V(w,\sigma) = \begin{cases} \{\!|\Phi_1|\!\}_V(w,\sigma) & \text{if } w,\sigma \models \varphi \\ \{\!|\Phi_2|\!\}_V(w,\sigma) & \text{otherwise} \end{cases}$$

$$\{\!|\Phi_1 + \Phi_2|\!\}_V(w,\sigma) = \{\!|\Phi_1|\!\}_V(w,\sigma) \uplus \{\!|\Phi_2|\!\}_V(w,\sigma)$$

$$\{\!|\textstyle\sum_x \Phi|\!\}_V(w,\sigma) = \biguplus_{i \in \mathsf{pos}(w)} \{\!|\Phi|\!\}_{V\cup\{x\}}(w, \sigma[x \mapsto i])$$

$$\{\!|\textstyle\sum_X \Phi|\!\}_V(w,\sigma) = \biguplus_{I \subseteq \mathsf{pos}(w)} \{\!|\Phi|\!\}_{V\cup\{X\}}(w, \sigma[X \mapsto I])$$

$$\{\!|\textstyle\prod_x \Psi|\!\}_V(w,\sigma) = \prod_{i=1}^{|w|} \{\!|\Psi|\!\}_{V\cup\{x\}}(w, \sigma[x \mapsto i])$$

where $\sigma[x \mapsto i]$ denotes the valuation obtained from σ by mapping x to i and keeping the other assignments unchanged (and similarly for second-order variables). Note that $\prod_{i=1}^{|w|}$ is the product in the semiring $\mathbb{N}\langle R^\star\rangle$.

Table 3 Semantics of core-wMSO.

product which lifts the weights defined by step-wMSO formulæ to sequences of weights.

Remark 6 The idea of splitting the syntax into three layers has also been used by Perevoshchikov to define the weighted assignment logic over infinite words [19, Chapter 6]. There, the fragment uWAL is related to our step-wMSO and the full logic WAL allows one to apply sums, as in our core-wMSO logic.

Formally, the semantics of core-wMSO is again defined by induction on the formula. For a set V of variables that contains the set $\mathsf{free}(\Phi)$ of free variables of Φ, we let $\{\!|\cdot|\!\}_V : \Sigma_V^+ \to \mathbb{N}\langle R^\star\rangle$ be defined by $\{\!|\Phi|\!\}(\overline{w}) = \emptyset$ if $\overline{w}$ is not valid, and if $\overline{w}$ is the valid encoding of (w,σ) then the semantics is given in Table 3. We simply write $\{\!|\Phi|\!\}$ for $\{\!|\Phi|\!\}_{\mathsf{free}(\Phi)}$.

Example 7 We give a core-wMSO$(\{a,b\},\{0,1\})$ formula equivalent with the $\{0,1\}$-weighted automaton of Fig. 1:

$$\Phi = \textstyle\sum_X \sum_y \mathsf{block}(X,y)\,?\,\prod_x (x \in X\,?\,1 : 0) : \mathbf{0}$$

where the MSO formula $\mathsf{block}(X,y)$ defined by

$$\begin{aligned} &\Big[X = \emptyset \wedge (\mathsf{first}(y) \vee \mathsf{last}(y)) \wedge P_b(y)\Big] \\ \vee\ &\Big[(\mathsf{last}(y) \vee P_b(y+1)) \wedge \\ &\quad \exists x \Big(x \leqslant y \wedge (\mathsf{first}(x) \vee P_b(x-1)) \wedge \\ &\qquad \forall z \big((x \leqslant z \leqslant y \Leftrightarrow z \in X) \wedge (z \in X \Rightarrow P_a(z)) \big) \Big) \Big] \end{aligned}$$

states that X is a (possibly empty) maximal block of consecutive a's *ending* in y. Variable y enables us to distinguish two reasons for X to be empty: either because the word starts with b, or ends with b. Notice the use of the constant $0 \in R$ in the step-wMSO formula $x \in X\,?\,1 : 0$, not to be confused with the atomic formula $\mathbf{0}$ of core-wMSO. Semantically, we have

$$\{\!|\Phi|\!\}(baabab) = \{\{000000, 011000, 000010, 000000\}\}\,.$$

We say that two core-wMSO formulæ Φ and Φ' are equivalent, written $\Phi \equiv \Phi'$, if they have the same semantics: $\mathsf{free}(\Phi) = \mathsf{free}(\Phi')$ and $\{\!|\Phi|\!\}(w,\sigma) = \{\!|\Phi'|\!\}(w,\sigma)$ for all $(w,\sigma) \in \Sigma^+_{\mathsf{free}(\Phi)}$. The equivalence of step-wMSO formulæ is defined similarly.

As in Section 3.5, we apply an aggregation function $\mathsf{aggr} : \mathbb{N}\langle R^\star\rangle \to S$ transforming a finite multiset of weight sequences from R into a weight in a (possibly new) set S. We obtain the *concrete* semantics $[\![\Phi]\!](\overline{w}) = \mathsf{aggr}(\{\!|\Phi|\!\}(\overline{w}))$.

Remark 8 Notice that some operators of step-wMSO and core-wMSO can be removed without affecting the expressive power of these logics. For instance, we will show in Lemma 18, that, in step-wMSO, Boolean formulæ used in if-then-else operators can be restricted to be of the form $x \in X$. Moreover, using $\sum_X$ operator, we can emulate operator $+$ and $\sum_x$ in core-wMSO. We chose to keep these operators nevertheless since they allow for an easier model of quantitative properties, without making further developments more complex.

4.2 Equivalence with weighted automata

The seminal result of [6], linking weighted automata and wMSO formulæ, may then be rephrased as in the following theorem.

Theorem 9 *For each R-weighted automaton $\mathcal{A}$ over alphabet Σ, we can effectively construct a sentence $\Phi_\mathcal{A}$ in* core-wMSO(Σ,R)*, such that $\{\!|\mathcal{A}|\!\}(w) = \{\!|\Phi_\mathcal{A}|\!\}(w)$ for all $w \in \Sigma^+$.*

For each sentence Φ in core-wMSO(Σ,R)*, we can effectively construct an R-weighted automaton $\mathcal{A}_\Phi$ over Σ such that $\{\!|\Phi|\!\}(w) = \{\!|\mathcal{A}_\Phi|\!\}(w)$ for all $w \in \Sigma^+$.*

Since $\mathbb{N}\langle R^\star\rangle$ is a semiring and the semantics $\{\!|\cdot|\!\}$ of weighted automata and core-wMSO is the natural semantics in this semiring, one may think that Theorem 9 is a formal corollary of [6]. This is almost true but not entirely. Here we insist that the set of weights used in the formula $\Phi_\mathcal{A}$ is the same as the set of weights used in the weighted automaton $\mathcal{A}$ (and vice versa for $\mathcal{A}_\Phi$ and Φ). This is not guaranteed by [6] though the proof can be adapted to obtain Theorem 9. We give in Section 4.5 a rather short proof of Theorem 9, which is based on some new ideas and which ensures that the set of weights is preserved.

The equivalence of Theorem 9 transfers to the *concrete* semantics without any conditions on the aggregation operator.

Corollary 10 *For each R-weighted automaton $\mathcal{A}$ over alphabet Σ, we can effectively construct a sentence $\Phi_\mathcal{A}$ in* core-wMSO(Σ, R)*, such that for all $w \in \Sigma^+$,*

$$[\![\mathcal{A}]\!](w) = \mathsf{aggr}(\{\!|\mathcal{A}|\!\}(w)) = \mathsf{aggr}(\{\!|\Phi_\mathcal{A}|\!\}(w)) = [\![\Phi_\mathcal{A}]\!](w)\,.$$

For each sentence Φ in core-wMSO(Σ, R)*, we can effectively construct an R-weighted automaton $\mathcal{A}_\Phi$ over Σ such that for all $w \in \Sigma^+$,*

$$[\![\Phi]\!](w) = \mathsf{aggr}(\{\!|\Phi|\!\}(w)) = \mathsf{aggr}(\{\!|\mathcal{A}_\Phi|\!\}(w)) = [\![\mathcal{A}_\Phi]\!](w)\,.$$

Instantiating the aggregation operator with $\mathsf{aggr}_{\mathtt{sr}}$ for semirings, $\mathsf{aggr}_{\mathtt{vm}}$ for valuation monoids, or $\mathsf{aggr}_{\mathtt{vs}}$ for valuation structures, we obtain

Corollary 11 *The expressive power of R-weighted automata and* core-wMSO(Σ, R) *is the same in each of the following cases*

1. *$R \subseteq S$ for a semiring $(S, +, \times, 0, 1)$,*
2. *$R \subseteq S$ for a valuation monoid $(S, +, 0, \mathsf{Val})$, or*
3. *$R \subseteq U$ for a valuation structure (U, Val, S, F).*

Before presenting the proof of Theorem 9, we show, in the next two subsections, the robustness of core-wMSO by adding some useful features in it, without changing its expressive power. Notice that the proof of equivalences are fully logical, and do not make use of translations into weighted automata: in particular, they do not require the use of Theorem 9.

4.3 Adding $\mathbf{0}$ and sum to step-wMSO

It may be convenient to use a sum operator $\Psi_1 + \Psi_2$ and constant $\mathbf{0}$ at the level of step-wMSO, as allowed for core-wMSO. We denote by $\mathbf{0}$-step-wMSO(Σ, R) the fragment allowing the constant $\mathbf{0}$

$$\Psi ::= \mathbf{0} \mid r \mid \varphi \,?\, \Psi : \Psi$$

and $+\mathbf{0}$-step-wMSO(Σ, R) the fragment allowing the constant $\mathbf{0}$ as well as the sum operator

$$\Psi ::= \mathbf{0} \mid r \mid \varphi \,?\, \Psi : \Psi \mid \Psi + \Psi\,.$$

The inductive semantics of Table 2 is enriched with

$$\{\!|\mathbf{0}|\!\}_V(w, \sigma) = \emptyset$$

and

$$\{\!|\Psi_1 + \Psi_2|\!\}_V(w, \sigma) = \{\!|\Psi_1|\!\}_V(w, \sigma) \uplus \{\!|\Psi_2|\!\}_V(w, \sigma)\,.$$

Formulæ Ψ in $+\mathbf{0}$-step-wMSO(Σ, R) allow for more concise specification of properties, and their semantics $\{\!|\Psi|\!\} \in \mathbb{N}\langle R\rangle$ is now a *finite* multiset of weights instead of simply a singleton or the empty multiset. Notice that the semantics of a formula $\prod_x \Psi$ is still well-defined.

Lemma 12 *The expressive power of* core-wMSO(Σ, R) *does not change if we replace* step-wMSO(Σ, R) *formulæ by* $\mathbf{0}$-step-wMSO(Σ, R) *formulæ.*

Proof Consider a formula Ψ of $\mathbf{0}$-step-wMSO(Σ, R). We have to show that the formula $\prod_x \Psi$ can be expressed in core-wMSO. First, we construct an MSO formula φ_Ψ such that for all valid $(w, \sigma) \in \Sigma_V^+$, $\{\!|\Psi|\!\}_V(w, \sigma) = \emptyset$ if and only if $w, \sigma \models \varphi_\Psi$. We build φ_Ψ by induction on the formula Ψ:

$$\mathbf{0}_\Psi = \top\,, \qquad r_\Psi = \bot$$
$$\text{and} \qquad \varphi \,?\, \Psi_1 : \Psi_2 = (\varphi \wedge \varphi_{\Psi_1}) \vee (\neg\varphi \wedge \varphi_{\Psi_2})\,.$$

It is immediate to verify the correctness of this construction.

Then, let $\tilde{\Psi}$ be the step-wMSO(Σ, R) formula obtained from Ψ by replacing every occurrence of $\mathbf{0}$ with an arbitrary constant r of R. We can prove that for all valid $(w, \sigma) \in \Sigma_V^+$ such that $\{\!|\Psi|\!\}_V(w, \sigma) \neq \emptyset$, we have $\{\!|\Psi|\!\}_V(w, \sigma) = \{\!|\tilde{\Psi}|\!\}_V(w, \sigma)$. This implies that $\prod_x \Psi$ is equivalent to the core-wMSO(Σ, R) formula

$$(\exists x\, \varphi_\Psi) \,?\, \mathbf{0} : \textstyle\prod_x \tilde{\Psi}\,. \qquad \square$$

Using the previous result, we finally show how to add sums in step formulæ.

Lemma 13 *The expressive power of* core-wMSO(Σ, R) *does not change if we replace* step-wMSO(Σ, R) *formulæ by* $+\mathbf{0}$-step-wMSO(Σ, R) *formulæ.*

Proof Removing sums is done in two steps. First, we can easily check that

$$\varphi \,?\, (\Psi_1 + \Psi_2) : \Psi_3 \equiv (\varphi \,?\, \Psi_1 : \Psi_3) + (\varphi \,?\, \Psi_2 : \mathbf{0})$$
$$\varphi \,?\, \Psi_1 : (\Psi_2 + \Psi_3) \equiv (\varphi \,?\, \Psi_1 : \Psi_2) + (\varphi \,?\, \mathbf{0} : \Psi_3)\,.$$

Hence, for each formula Ψ of $+\mathbf{0}$-step-wMSO(Σ, R) we can construct an equivalent formula $\Psi_1 + \cdots + \Psi_n$ where the Ψ_i are in $\mathbf{0}$-step-wMSO(Σ, R).

For the second step, we let $\overline{X} = (X_1, \ldots, X_n)$ be a tuple of second-order variables. We simply write $\sum_{\overline{X}}$ instead of $\sum_{X_1} \cdots \sum_{X_n}$. We also let partition$(\overline{X})$ be an MSO formula stating that the sets associated with variables $X_1, \ldots, X_n$ form a partition of the positions. Let $\Psi' = x \in X_1 \,?\, \Psi_1 : \cdots x \in X_n \,?\, \Psi_n : \mathbf{0}$. Notice that Ψ' is in $\mathbf{0}$-step-wMSO(Σ, R). We claim that the core-wMSO formulæ

$$\Phi = \textstyle\prod_x (\Psi_1 + \cdots + \Psi_n)$$
$$\Phi' = \textstyle\sum_{\overline{X}} \text{partition}(\overline{X}) \,?\, \left[\prod_x \Psi'\right] : \mathbf{0}$$

are equivalent. The proof that $\Phi \equiv \Phi'$ relies on the distributivity of the semiring $\mathbb{N}\langle R^\star\rangle$. Details are left to the reader. We finally obtain the result by applying Lemma 12 to the formula $\prod_x \Psi'$. □

4.4 Adding new binary operators

For certain modelling applications, it might be useful to enhance our core logic with extra operators, for instance binary operators. In particular, the classical wMSO logic (see Section 5) allows a quantitative extension of the conjunction operator, though in a restricted manner. Extra operators also allow to define quantitative properties compositionally.

Example 14 Let us denote by Φ_a the core-wMSO formula of Example 7 computing the maximal length of blocks of a's when concretely evaluated in the semiring $(\mathbb{N}\cup\{-\infty\}, \max, +, -\infty, 0)$. If we want to compute the sum of the maximal length of blocks of a's and of the maximal length of blocks of b's in a word, it seems natural to authorise the use of the binary sum to decompose this objective in two distinct ones. Then, we could use formula $\Phi_a \diamond \Phi_b$ (where Φ_b is obtained by inverting the roles of a and b in Φ_a), denoting by $\diamond$ the new binary operator allowing to compute binary sums.

We now show how to add binary operators to the logic without changing its expressive power. Indeed, we can proceed in the same way to add operators with other arities.

Let $\diamond\colon R^2 \to R$ be an arbitrary binary operation over R. It naturally extends to a *partial* binary operation over $R^\star$ by setting for all sequences $\alpha = r_1 \cdots r_n$ and $\alpha' = r'_1 \cdots r'_n$ of same length

$$\alpha \diamond \alpha' = (r_1 \diamond r'_1) \cdots (r_n \diamond r'_n)\,.$$

The operator is not defined on sequences of different lengths. Then, as explained in the preliminaries, this binary operation extends to finite multisets in $\mathbb{N}\langle R\rangle$ or in $\mathbb{N}\langle R^\star\rangle$.

We enrich the syntax defined in Table 1 with $\Psi_1 \diamond \Psi_2$ for step-wMSO and with $\Phi_1 \diamond \Phi_2$ for core-wMSO. The semantics defined in Tables 2 and 3 are also enriched with

$$\begin{aligned}\{\!|\Psi_1 \diamond \Psi_2|\!\}_V(w,\sigma) &= \{\!|\Psi_1|\!\}_V(w,\sigma) \diamond \{\!|\Psi_2|\!\}_V(w,\sigma)\\ \{\!|\Phi_1 \diamond \Phi_2|\!\}_V(w,\sigma) &= \{\!|\Phi_1|\!\}_V(w,\sigma) \diamond \{\!|\Phi_2|\!\}_V(w,\sigma)\,.\end{aligned}$$

Example 15 When $\diamond$ is interpreted as $+$, the formula $\Phi_a \diamond \Phi_b$ of Example 14 has the following abstract semantics over $w = baaba$:

$$\begin{aligned}\{\!|\Phi_a|\!\}(w) &= \{\{00000, 01100, 00001\}\}\\ \{\!|\Phi_b|\!\}(w) &= \{\{10000, 00010, 00000\}\}\\ \{\!|\Phi_a \diamond \Phi_b|\!\}(w) &= \{\{10000, 00010, 00000, 11100, 01110,\\ &\qquad 01100, 10001, 00011, 00001\}\}\,.\end{aligned}$$

Notice that its concrete semantics in the semiring $(\mathbb{N}\cup\{-\infty\}, \max, +, -\infty, 0)$ is 3 which is indeed the sum of the maximal length of blocks of a's and b's.

We now show that adding such a binary operator does not increase the expressive power of step-wMSO and core-wMSO respectively. To that extent, let us denote by $\diamond$-step-wMSO the logic step-wMSO enriched with the diamond operator. We also write $\diamond$-core-wMSO for core-wMSO enriched with the diamond operator.

Proposition 16 *The expressive power of* step-wMSO *(respectively,* core-wMSO*) does not change if we add formulæ $\Psi_1 \diamond \Psi_2$ in* step-wMSO *(respectively, $\Phi_1 \diamond \Phi_2$ in* core-wMSO*).*

Proof Let Ψ_1, Ψ_2, Ψ_3 be $\diamond$-step-wMSO formulæ. By case splitting it is clear that

$$\begin{aligned}(\varphi\,?\,\Psi_1 : \Psi_2) \diamond \Psi_3 &\equiv \varphi\,?\,(\Psi_1 \diamond \Psi_3) : (\Psi_2 \diamond \Psi_3)\\ \Psi_3 \diamond (\varphi\,?\,\Psi_1 : \Psi_2) &\equiv \varphi\,?\,(\Psi_3 \diamond \Psi_1) : (\Psi_3 \diamond \Psi_2)\,.\end{aligned}$$

By applying inductively the above equivalences, we can rewrite any $\diamond$-step-wMSO formula until the $\diamond$ operator is only applied to constants from R. Now, for $r_1, r_2 \in R$, we have $r_1 \diamond r_2 \in R$ which is a legal step-wMSO formula. Therefore, we have proved that any formula in $\diamond$-step-wMSO can be rewritten in a semantically equivalent step-wMSO formula.

Since $\diamond$ distributes over $\uplus$ in the semiring $\mathbb{N}\langle R^\star\rangle$ we deduce that $\diamond$ distributes (left and right) over the sum operators of core-wMSO (we only give equations for left distributivity below):

$$\begin{aligned}\Phi_1 \diamond (\Phi_2 + \Phi_3) &\equiv (\Phi_1 \diamond \Phi_2) + (\Phi_1 \diamond \Phi_3)\\ \Phi_1 \diamond \textstyle\sum_x \Phi_2 &\equiv \textstyle\sum_x (\Phi_1 \diamond \Phi_2)\\ \Phi_1 \diamond \textstyle\sum_X \Phi_2 &\equiv \textstyle\sum_X (\Phi_1 \diamond \Phi_2)\,.\end{aligned}$$

By case splitting, we also get as above that $\diamond$ distributes over the if-then-else operator. Note also that $\mathbf{0} \diamond \Phi \equiv \mathbf{0} \equiv \Phi \diamond \mathbf{0}$. Hence, applying inductively the above equivalences, we can rewrite any $\diamond$-core-wMSO formula until the $\diamond$ operator is only applied to products $(\prod_x \Psi) \diamond (\prod_x \Psi')$ (up to renaming, we may assume that the same variable x is used in both products). We claim that

$$(\textstyle\prod_x \Psi) \diamond (\prod_x \Psi') \equiv \prod_x (\Psi \diamond \Psi')\,. \tag{2}$$

Let $w \in \Sigma^+$ be a word of length n and σ be a valuation of the variables in $V \supseteq (\mathsf{free}(\Psi) \cup \mathsf{free}(\Psi')) \setminus \{x\}$ (the semantical equivalence is trivially verified on invalid encodings $\overline{w} \in \Sigma_V^+$). Since Ψ and Ψ' are step-wMSO formulæ, for all $1 \leqslant i \leqslant n$ we have,

$$\begin{aligned} A_i &= \{\!|\Psi|\!\}_{V\cup\{x\}}(w, \sigma[x \mapsto i]) \in \mathbb{N}\langle R\rangle \\ B_i &= \{\!|\Psi'|\!\}_{V\cup\{x\}}(w, \sigma[x \mapsto i]) \in \mathbb{N}\langle R\rangle \\ A_i \diamond B_i &= \{\!|\Psi \diamond \Psi'|\!\}_{V\cup\{x\}}(w, \sigma[x \mapsto i]) \in \mathbb{N}\langle R\rangle \end{aligned}$$

and these multisets are singletons. Using this fact, we can easily check that[3]

$$(A_1 \cdots A_n) \diamond (B_1 \cdots B_n) = (A_1 \diamond B_1) \cdots (A_n \diamond B_n)\,. \quad (3)$$

We obtain (2) since

$$\begin{aligned} \{\!|\textstyle\prod_x \Psi|\!\}_V(w,\sigma) &= A_1 \cdots A_n \\ \{\!|\textstyle\prod_x \Psi'|\!\}_V(w,\sigma) &= B_1 \cdots B_n \\ \{\!|\textstyle\prod_x \Psi \diamond \Psi'|\!\}_V(w,\sigma) &= (A_1 \diamond B_1) \cdots (A_n \diamond B_n)\,. \end{aligned}$$

This concludes the proof since $\Psi \diamond \Psi'$ can be translated into an equivalent step-wMSO formula. □

Consider the concrete semantics $[\![\cdot]\!] = \mathsf{aggr}(\{\!|\cdot|\!\})$ defined with an aggregation $\mathsf{aggr}\colon \mathbb{N}\langle R^\star\rangle \to S$. Assume also that $\diamond\colon S^2 \to S$ is defined on S. We say that $\diamond$ *distributes* over aggr if for all $n > 0$ and $A, B \in \mathbb{N}\langle R^n\rangle$, we have

$$\mathsf{aggr}(A \diamond B) = \mathsf{aggr}(A) \diamond \mathsf{aggr}(B)\,.$$

In this case, we can easily check that the concrete semantics is *compositional*: for all $\overline{w} \in \Sigma_V^+$, we have $[\![\Phi_1 \diamond \Phi_2]\!](\overline{w}) = [\![\Phi_1]\!](\overline{w}) \diamond [\![\Phi_2]\!](\overline{w})$.

Example 17 In the case of a semiring $(S, +, \times, 0, 1)$, the aggregation is $\sum \prod$ (as defined in Example 3). We may easily check that for commutative semirings, the concrete semantics is compositional with respect to the product $\times$.

4.5 Proof of Theorem 9

We show first that we can restrict step-wMSO to a smaller fragment without changing the expressive power of core-wMSO. The syntax of set-step-wMSO(R) is given by

$$\Psi ::= r \mid x \in X\,?\,\Psi : \Psi \qquad \text{(set-step-wMSO)}$$

where x is a *fixed* first-order variable, X ranges over monadic second-order variables and r ranges over R. This restriction has also been considered previously. For instance, in the weighted timed setting [12], Droste and Perevoshchikov translate weighted (timed) automata into weighted sentences where Boolean formulæ inside the universal quantification are of the form $x \in X$ only. In our context, it means only set-step-wMSO inside a product $\prod_x$. Similarly, in the context of trees [16], Fülöp, Stüber, and Vogler use an operation $H(\omega)$ which renames every node of the input tree with an *operator* from some family ω (coming from a multi-operator monoid). Again, the renaming is described by means of formulæ of the form $x \in X$ only, and not by more general MSO formulæ.

Lemma 18 *The expressive power of* core-wMSO(Σ, R) *does not change if we replace* step-wMSO(Σ, R) *formulæ by* set-step-wMSO(R) *formulæ.*

Proof We start with a core-wMSO formula $\Phi = \prod_x \Psi$ where Ψ is a step-wMSO(Σ, R) formula. Let $\varphi_1, \ldots, \varphi_n$ be the MSO formulæ occurring in Ψ as conditions of the if-then-else operator. We let $\overline{X} = (X_1, \ldots, X_n)$ be a tuple of fresh second-order variables. Let also Ψ' be the formula obtained from Ψ by replacing every occurrence of φ_i by $x \in X_i$, for all $1 \leqslant i \leqslant n$. Notice that Ψ' is a set-step-wMSO(R) formula. We claim that $\Phi = \prod_x \Psi$ is equivalent to the formula

$$\Phi' = \textstyle\sum_{\overline{X}} \left(\bigwedge_i \forall x\, (x \in X_i \leftrightarrow \varphi_i) \right) ? \left(\prod_x \Psi' \right) : \mathbf{0}\,.$$

Indeed, let $V = \mathsf{free}(\Phi) = \mathsf{free}(\prod_x \Psi)$ and $V' = V \cup \{X_1, \ldots, X_n\}$. For every valid $(w, \sigma) \in \Sigma_V^+$ there is a unique $(w, \sigma') \in \Sigma_{V'}^+$ such that $\sigma'|_V = \sigma$ and $w, \sigma' \models \bigwedge_i \forall x\, (x \in X_i \leftrightarrow \varphi_i)$. For all $1 \leqslant i \leqslant n$, we have $\sigma'(X_i) = \{j \in \mathsf{pos}(w) \mid w, \sigma[x \mapsto j] \models \varphi_i\}$. We obtain $\{\!|\Phi'|\!\}(w,\sigma) = \{\!|\prod_x \Psi'|\!\}(w,\sigma')$. Then, it is easy to check by induction on Ψ that for all $j \in \mathsf{pos}(w)$ we have $\{\!|\Psi|\!\}(w, \sigma[x \mapsto j]) = \{\!|\Psi'|\!\}(w, \sigma'[x \mapsto j])$. We deduce that $\{\!|\Phi'|\!\}(w,\sigma) = \{\!|\prod_x \Psi'|\!\}(w,\sigma') = \{\!|\prod_x \Psi|\!\}(w,\sigma) = \{\!|\Phi|\!\}(w,\sigma)$ for all valid $(w,\sigma) \in \Sigma_V^+$. □

Proof (of Theorem 9) Let $\mathcal{A} = (Q, \Delta, \mathsf{wgt}, I, F)$ be a weighted automaton. We use a set variable X_δ for each transition $\delta \in \Delta$ and we let $\overline{X} = (X_\delta)_{\delta\in\Delta}$. Intuitively, the tuple $\overline{X}$ encodes a run of $\mathcal{A}$ over a word w when each set variable X_δ is interpreted as the set of positions at which transition δ is used in that run.

We can easily write an MSO formula $\mathsf{run}(\overline{X})$ which evaluates to true on some word w if and only if $\overline{X}$ encodes a run of $\mathcal{A}$ on w starting from I and ending in F. First, we state that $\overline{X}$ is a partition on the positions of w. Then we request that if the first position of w is in X_δ then $\delta \in I \times \Sigma \times Q$ is initial. Similarly, the transition of the last position should be final. Finally, if $\delta = (p, a, q)$ and $\delta' = (p', a', q')$ are the transitions of

[3] Actually, (3) holds for arbitrary multisets $A_i, B_j \in \mathbb{N}\langle R\rangle$.

two consecutive positions of w then $q = p'$. It is routine to write all these requirements in MSO (even in FO_3).

Assuming that $\mathsf{run}(\overline{X})$ holds, we let $\mathsf{weight}(x, \overline{X})$ be the set-step-wMSO formula which evaluates to $\mathsf{wgt}(\delta)$ where $\delta \in \Delta$ is the unique transition such that $x \in X_\delta$. Formally, if $\Delta = \{\delta_1, \delta_2, \ldots, \delta_n\}$ then we define $\mathsf{weight}(x, \overline{X})$ as

$$x \in X_{\delta_1} \,?\, \mathsf{wgt}(\delta_1) : \cdots x \in X_{\delta_{n-1}} \,?\, \mathsf{wgt}(\delta_{n-1}) : \mathsf{wgt}(\delta_n)$$

and

$$\Phi_{\mathcal{A}} = \textstyle\sum_{\overline{X}} \mathsf{run}(\overline{X}) \,?\, \left(\prod_x \mathsf{weight}(x, \overline{X})\right) : \mathbf{0}\,.$$

We can easily check that for all words $w \in \Sigma^+$ we have $\{\!|\mathcal{A}|\!\}(w) = \{\!|\Phi|\!\}(w)$.

Conversely, we proceed by induction on Φ, hence we have to deal with free variables. So we construct for each formula Φ a weighted automaton $\mathcal{A}_\Phi$ over the alphabet $\Sigma_{\mathsf{free}(\Phi)} = \Sigma \times \{0,1\}^{\mathsf{free}(\Phi)}$ such that for all $\overline{w} \in \Sigma^+_{\mathsf{free}(\Phi)}$ we have $\{\!|\Phi|\!\}(\overline{w}) = \{\!|\mathcal{A}_\Phi|\!\}(\overline{w})$.

It is folklore that we may increase the set of variables encoded in the alphabet whenever needed, e.g., to deal with sum or if-then-else. Formally, if $V \subseteq V'$ then we can *lift* an automaton $\mathcal{A}_V$ defined on the alphabet Σ_V to an automaton $\mathcal{A}_{V'}$ defined on $\Sigma_{V'}$ such that for all valid $(w, \sigma) \in \Sigma^+_{V'}$ we have $\{\!|\mathcal{A}_{V'}|\!\}(w, \sigma) = \{\!|\mathcal{A}_V|\!\}(w, \sigma|_V)$.

The automaton $\mathcal{A}_{\mathbf{0}}$ has a single state which is initial but not final and has no transitions.

We recall the classical constructions for the additive operators of core-wMSO: $+$, $\sum_x$ and $\sum_X$.

If $\Phi = \Phi_1 + \Phi_2$ then $\mathcal{A}_\Phi$ is obtained as the disjoint union of $\mathcal{A}_{\Phi_1}$ and $\mathcal{A}_{\Phi_2}$, both lifted to Σ_Φ.

If $\Phi = \sum_X \Phi_1$ then $\mathcal{A}_\Phi$ is obtained via a variant[4] of the projection construction starting from $\mathcal{A}_{\Phi_1}$. Assume that $\mathcal{A}_{\Phi_1} = (Q, \Delta, \mathsf{wgt}, I, F)$. We define $\mathcal{A}_\Phi = (Q \times \{0,1\}, \Delta', \mathsf{wgt}', I \times \{0\}, F \times \{0,1\})$ over alphabet $\Sigma_{\mathsf{free}(\Phi)}$ by letting

$$((p,i), a, (q,j)) \in \Delta' \text{ iff } (p, (a,j), q) \in \Delta$$

where (a, j) denotes the letter in $\Sigma_{\mathsf{free}(\Phi) \cup \{X\}}$ where the value of the X-component is given by j and the remaining $\Sigma_{\mathsf{free}(\Phi)}$-components (different from X) are given by a. We also let

$$\mathsf{wgt}'((p,i), a, (q,j)) = \mathsf{wgt}(p, (a,j), q)\,.$$

This transfer of the alphabet component for X to the state of $\mathcal{A}_\Phi$ allows us to define a bijection between the accepting runs of $\mathcal{A}_{\Phi_1}$ and the accepting runs of $\mathcal{A}_\Phi$, preserving sequences of weights. Then, we deduce easily that $\{\!|\mathcal{A}_\Phi|\!\} = \{\!|\Phi|\!\}$ over alphabet $\Sigma_{\mathsf{free}(\Phi)}$.

If $\Phi = \sum_x \Phi_1$, the construction is almost the same. In the definition of $\mathcal{A}_\Phi$, the set of accepting states is $F \times \{1\}$ and the transitions are given by

$$((p,0), a, (q,j)) \in \Delta' \text{ iff } (p, (a,j), q) \in \Delta$$
$$((p,1), a, (q,1)) \in \Delta' \text{ iff } (p, (a,0), q) \in \Delta$$

with weights inherited as before

$$\mathsf{wgt}'((p,0), a, (q,j)) = \mathsf{wgt}(p, (a,j), q)$$
$$\mathsf{wgt}'((p,1), a, (q,1)) = \mathsf{wgt}(p, (a,0), q)\,.$$

We turn now to the more interesting cases: if-then-else and $\prod_x$. Noticed that $\varphi \,?\, \Phi_1 : \Phi_2$ is equivalent to $(\varphi \,?\, \Phi_1 : \mathbf{0}) + (\neg\varphi \,?\, \Phi_2 : \mathbf{0})$, hence we only need to construct an automaton for $\Phi = \varphi \,?\, \Phi_1 : \mathbf{0}$. Let $V = \mathsf{free}(\Phi) = \mathsf{free}(\varphi) \cup \mathsf{free}(\Phi_1)$. Since φ is a (Boolean) MSO formula, by [3,15,22], we can construct a *deterministic*[5] automaton $\mathcal{A}_\varphi$ over the alphabet Σ_V which accepts a word $\overline{w} \in \Sigma^+_V$ if and only if it is a valid encoding $\overline{w} = (w, \sigma)$ satisfying φ. Now, by induction, we have an automaton $\mathcal{A}_{\Phi_1}$ over Σ_V which is equivalent to Φ_1: for all $\overline{w} \in \Sigma^+_V$, $\{\!|\Phi_1|\!\}(\overline{w}) = \{\!|\mathcal{A}_{\Phi_1}|\!\}(\overline{w})$. The automaton $\mathcal{A}_\Phi$ is obtained as the "intersection" of $\mathcal{A}_\varphi$ and $\mathcal{A}_{\Phi_1}$ (see the formal construction below). Now, let $\overline{w} \in \Sigma^+_V$. If $\overline{w}$ is not valid or $\overline{w} = (w, \sigma)$ is valid and does not satisfy φ then $\mathcal{A}_\varphi$ (hence also $\mathcal{A}_\Phi$) has no accepting run on $\overline{w}$ and we obtain $\{\!|\mathcal{A}_\Phi|\!\}(\overline{w}) = \emptyset = \{\!|\Phi|\!\}(\overline{w})$. On the other hand, assume that $\overline{w} = (w, \sigma)$ is valid and satisfy φ. Since $\mathcal{A}_\varphi$ is deterministic, there is a bijection between the accepting runs of $\mathcal{A}_\Phi$ and the accepting runs of $\mathcal{A}_{\Phi_1}$. By construction of $\mathcal{A}_\Phi$, this bijection preserves the sequence of weights associated with a run. We deduce that $\{\!|\mathcal{A}_\Phi|\!\}(w, \sigma) = \{\!|\mathcal{A}_{\Phi_1}|\!\}(w, \sigma) = \{\!|\Phi|\!\}(w, \sigma)$.

We give now the formal definition of $\mathcal{A}_\Phi$. Let $\mathcal{A}_{\Phi_1} = (Q_1, \Delta_1, \mathsf{wgt}_1, I_1, F_1)$ be the weighted automaton over Σ_V for Φ_1. Let $\mathcal{A}_\varphi = (Q_2, \Delta_2, I_2, F_2)$ be the deterministic automaton over Σ_V for φ. Then, we define $\mathcal{A}_\Phi = (Q, \Delta, \mathsf{wgt}, I, F)$ with $Q = Q_1 \times Q_2$, $I = I_1 \times I_2$, $F = F_1 \times F_2$, Δ is the set of triples $\delta = ((p_1, p_2), a, (q_1, q_2))$ such that $\delta_1 = (p_1, a, q_1) \in \Delta_1$ and $(p_2, a, q_2) \in \Delta_2$, and $\mathsf{wgt}(\delta) = \mathsf{wgt}(\delta_1)$.

Finally, it remains to deal with the case $\Phi = \prod_x \Psi$. By Lemma 18, we may assume that Ψ is a formula in set-step-wMSO(R). So $\mathsf{free}(\Psi) = \{x, X_1, \ldots, X_n\}$ and the tests in Ψ are of the form $x \in X_i$ for some $i \in \{1, \ldots, n\}$. Also, $\mathsf{free}(\Phi) = \{X_1, \ldots, X_n\}$ consists of second-order variables only, so every word $\overline{w} \in \Sigma^+_{\mathsf{free}(\Phi)}$ is valid.

[4] As already noticed in [10], a simple projection does not work. Indeed, it would result in transition labels that are multisets of weights, which is not possible since our theorem requires the same set of weights for the automaton and the formula.

[5] We could also use an *unambiguous* automaton for $\mathcal{A}_\varphi$.

For every $\tau \in \{0,1\}^n$, we define the evaluation $\Psi(\tau)$ inductively as follows: $r(\tau) = r$ and

$$(x \in X_i \,?\, \Psi_1 : \Psi_2)(\tau) = \begin{cases} \Psi_1(\tau) & \text{if } \tau_i = 1 \\ \Psi_2(\tau) & \text{otherwise.} \end{cases}$$

Let $\overline{w} = (a_1, \tau_1) \cdots (a_k, \tau_k) \in \Sigma^+_{\mathsf{free}(\Phi)}$ with $a_j \in \Sigma$ and $\tau_j \in \{0,1\}^{\mathsf{free}(\Phi)}$ for all $1 \leqslant j \leqslant k$. We can easily check that $\{\!|\Phi|\!\}(\overline{w}) = \{\{\Psi(\tau_1) \cdots \Psi(\tau_k)\}\}$. Define $\mathcal{A}_\Phi = (Q, \Delta, \mathsf{wgt}, I, F)$ with a single state which is both initial and final ($Q = I = F = \{q\}$) and for every $a \in \Sigma$ and $\tau \in \{0,1\}^{\mathsf{free}(\Phi)}$, there is a transition $\delta = (q, (a, \tau), q) \in \Delta$ with $\mathsf{wgt}(\delta) = \Psi(\tau)$. It is clear that for every word $\overline{w} = (a_1, \tau_1) \cdots (a_k, \tau_k) \in \Sigma^+_{\mathsf{free}(\Phi)}$, the automaton $\mathcal{A}_\Phi$ has a single run on $\overline{w}$ whose sequence of weights is $\Psi(\tau_1) \cdots \Psi(\tau_k)$. Therefore, $\{\!|\mathcal{A}_\Phi|\!\}(\overline{w}) = \{\!|\Phi|\!\}(\overline{w})$, which concludes the proof. □

$$\begin{aligned}
[\![r]\!]_V(w,\sigma) &= r \\
[\![\beta]\!]_V(w,\sigma) &= \begin{cases} 1 & \text{if } w, \sigma \models \beta \\ 0 & \text{otherwise} \end{cases} \\
[\![\neg\varphi]\!]_V(w,\sigma) &= \begin{cases} 1 & \text{if } [\![\varphi]\!]_V(w,\sigma) = 0 \\ 0 & \text{otherwise} \end{cases} \\
[\![\xi \vee \xi']\!]_V(w,\sigma) &= [\![\xi]\!]_V(w,\sigma) + [\![\xi']\!]_V(w,\sigma) \\
[\![\xi \wedge \xi']\!]_V(w,\sigma) &= [\![\xi]\!]_V(w,\sigma) \diamond [\![\xi']\!]_V(w,\sigma) \\
[\![\exists x\, \xi]\!]_V(w,\sigma) &= \sum_{i \in \mathsf{pos}(w)} [\![\xi]\!]_{V \cup \{x\}}(w, \sigma[x \mapsto i]) \\
[\![\forall x\, \xi]\!]_V(w,\sigma) &= \mathsf{Val}(([\![\xi]\!]_{V \cup \{x\}}(w, \sigma[x \mapsto i]))_{i \in \mathsf{pos}(w)}) \\
[\![\exists X\, \xi]\!]_V(w,\sigma) &= \sum_{I \subseteq \mathsf{pos}(w)} [\![\xi]\!]_{V \cup \{X\}}(w, \sigma[X \mapsto I]) \\
[\![\forall X\, \varphi]\!]_V(w,\sigma) &= \mathsf{Val}(([\![\varphi]\!]_{V \cup \{X\}}(w, \sigma[X \mapsto I]))_{I \subseteq \mathsf{pos}(w)}) \,.
\end{aligned}$$

Table 4 Semantics of wMSO over product valuation monoids. β stands for an atomic formula among $P_a(x)$, $x \leqslant y$, $x \in X$.

5 Restricted weighted MSO logic

We now present the syntax and semantics of the full wMSO logic that has been studied over semirings [6], valuation monoids [10] and valuation structures [11]. The syntax used in these previous works is different. Also, there is no separate semantics for the Boolean fragment, instead, it is obtained as a special case of the quantitative semantics. As we will see, this choice requires some additional conditions on the weight domain, called hypothesis (**01**) below. In order to obtain the same expressive power as weighted automata, we also have to restrict the usage of conjunction and universal quantifications in wMSO.

We present effective translations in both directions relating restricted wMSO with core-wMSO, and the conditions that the weight domain has to fulfil in different settings. Using Corollary 11, we obtain a purely logical proof of the equivalence between restricted wMSO and weighted automata, using core-wMSO as an intermediary, simple and elegant, logical formalism.

5.1 wMSO over semirings and valuation monoids

For a set R of weights, the logic $\mathsf{wMSO}(\Sigma, R)$ studied in [6] and [10] is given by the following grammar

$$\Xi ::= r \mid \varphi \mid \Xi \vee \Xi \mid \Xi \wedge \Xi \mid \exists x\, \Xi \mid \forall x\, \Xi \mid \exists X\, \Xi$$

where x and y are first-order variables, X is a second-order variable, $\varphi \in \mathsf{MSO}(\Sigma)$ and $r \in R$.

Defining the semantics of the conjunction operator requires the introduction of an additional binary operator $\diamond$. Expressing the semantics of Boolean formulæ with the quantitative semantics of weighted formulæ requires special elements 0 and 1. A semiring is naturally equipped with such objects, the multiplication and its zero and unit elements. A valuation monoid equipped with such objects is called a product valuation monoid. More formally, a *product valuation monoid* is a tuple $(S, +, 0, \mathsf{Val}, \diamond, 1)$ with $(S, +, 0)$ a commutative monoid, $\mathsf{Val}\colon S^\star \to S$, $\diamond\colon S^2 \to S$, and $1 \in S$. In [10], additional conditions are given in the definition of product valuation monoids. We will highlight these conditions in the following. Semirings are a special case of product valuation monoids. In this section, we assume that $\{0,1\} \subseteq R \subseteq S$ is a subset of a product valuation monoid $(S, +, 0, \mathsf{Val}, \diamond, 1)$.

The semantics of a Boolean formula $\xi = \varphi$ or a weighted formula $\xi = \Xi$ is defined uniformly. Let $\overline{w} \in \Sigma^+_V$ with $\mathsf{free}(\xi) \subseteq V$. We define $[\![\xi]\!]_V(\overline{w}) = 0$ if $\overline{w}$ is not valid. If $\overline{w}$ is the valid encoding of (w, σ) then the semantics is given in Table 4.

To relate wMSO with core-wMSO, we introduce an intermediary logic wMSO′ which semantically separates the Boolean fragment from the quantitative one. The syntax of $\mathsf{wMSO}'(\Sigma, R)$ is given by the grammar

$$\Xi ::= r \mid \varphi \,?\, \Xi : \Xi \mid \Xi \vee \Xi \mid \Xi \wedge \Xi \mid \exists x\, \Xi \mid \forall x\, \Xi \mid \exists X\, \Xi$$

where x and y are first-order variables, X is a second-order variable, $\varphi \in \mathsf{MSO}(\Sigma)$ and $r \in R$. Notice that a Boolean formula is no more a weighted formula. Instead, the if-then-else construct is used in wMSO′.

The semantics of wMSO′ is defined as in Table 4, removing the cases β, $\neg\varphi$ and $\forall X\, \varphi$, restricting the other cases to weighted formulæ $\xi = \Xi$, and using the case-

splitting semantics for the if-then-else operator:

$$[\![\varphi\,?\,\Xi : \Xi']\!]_V(w,\sigma) = \begin{cases} [\![\Xi]\!]_V(w,\sigma) & \text{if } w,\sigma \models \varphi \\ [\![\Xi']\!]_V(w,\sigma) & \text{otherwise.} \end{cases}$$

Let us now give a list of properties that the product valuation monoid should fulfil in order for wMSO and wMSO′ to be expressively equivalent:

- $0 \diamond s = s \diamond 0 = 0$ for all $s \in S$,
- $1 \diamond s = s \diamond 1 = s$ for all $s \in S$,
- $\mathsf{Val}(s_1 \cdots s_n) = 0$ for all $s_1 \cdots s_n \in S^+$ such that $s_i = 0$ for some i,
- and $\mathsf{Val}(1 \cdots 1) = 1$.

In the following, we call this list of hypotheses (**01**).

Lemma 19 *Under hypothesis* (**01**)*,* $\mathsf{wMSO}(\Sigma, R)$ *and* $\mathsf{wMSO}'(\Sigma, R)$ *have the same expressive power.*

Proof We first show by induction that, if $\varphi \in \mathsf{MSO}$ is a Boolean formula then the semantics of wMSO gives for a valid word $\overline{w} = (w, \sigma)$

$$[\![\varphi]\!]_V(w,\sigma) = \begin{cases} 1 & \text{if } w,\sigma \models \varphi \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The property holds for atomic formulæ by definition of the semantics in Table 4. It is also trivial by induction for negation. For conjunction, the property follows from the first two items of hypothesis (**01**) (only used with $s \in \{0,1\}$), and for the universal quantifications, it follows from the last two items.

Then, the translation of $\Xi \in \mathsf{wMSO}$ to $\Xi' \in \mathsf{wMSO}'$ only replaces *maximal* Boolean subformulæ φ in Ξ with $\varphi\,?\,1 : 0$. An occurrence of a Boolean subformula is maximal in Ξ if it is not a strict subformula of another Boolean subformula in Ξ. The equality between both semantics is a direct consequence of (4).

In the other direction, we simply replace formulæ $\varphi\,?\,\Xi_1 : \Xi_2$ with $(\varphi \wedge \Xi_1) \vee (\neg\varphi \wedge \Xi_2)$. Then, the equality between both semantics is a consequence of (4) and the first two items of hypothesis (**01**), used here with all possible values of s. □

In [10], several fragments of wMSO are studied. For instance, *almost Boolean* formulæ of wMSO is the fragment containing all constants $r \in R$, all Boolean formulæ φ, and which is closed under disjunction and conjunction. In contrast, in wMSO′, we define *step formulæ* as the fragment containing all constants $r \in R$ and closed under the if-then-else operator. Notice that step formulæ of wMSO′ correspond to step-wMSO formulæ. We now show the relationship between the two fragments.

Lemma 20 *For every almost Boolean formula* Ξ *of* $\mathsf{wMSO}(\Sigma, R)$ *we can construct an equivalent step formula* Ξ' *of* $\mathsf{wMSO}'(\Sigma, S)$*. Conversely, under hypothesis* (**01**)*, every step formula* Ξ' *of* $\mathsf{wMSO}'(\Sigma, R)$ *can be translated into an equivalent almost Boolean formula* Ξ *of* $\mathsf{wMSO}(\Sigma, R)$*.*

Proof Let $\{\varphi_1, \ldots, \varphi_n\}$ be the set of maximal MSO formulæ occurring in Ξ. The proof is by induction on n.

When $n = 0$, Ξ is a positive Boolean combination of constants. Replacing $\wedge$ with $\diamond$ and $\vee$ with $+$, we obtain an expression which evaluates to a new constant $s \in S$ (not necessarily in R). Then, $\Xi' = s$ is a step formula which is equivalent to Ξ.

Assume now that $n > 0$. Consider the formulae $\Xi[\varphi_n/1]$ and $\Xi[\varphi_n/0]$ obtained by substituting maximal occurrences of φ_n with 1 and 0 respectively. These are almost Boolean formulæ with $n-1$ maximal Boolean formula. By induction, there are equivalent step formulæ Ξ'_1 and Ξ'_0 of wMSO′. Moreover, by (4), we can show by induction that for a valid word $\overline{w} = (w, \sigma)$

$$[\![\Xi]\!]_V(w,\sigma) = \begin{cases} [\![\Xi[\varphi_n/1]]\!]_V(w,\sigma) & \text{if } w,\sigma \models \varphi \\ [\![\Xi[\varphi_n/0]]\!]_V(w,\sigma) & \text{otherwise.} \end{cases}$$

Then, formula $\Xi' = \varphi_n\,?\,\Xi'_1 : \Xi'_0$ is a step formula equivalent to Ξ.

Conversely, the translation from wMSO′ to wMSO shown in Lemma 19, if applied to a step formula, produces an almost Boolean formula. □

Notice that the translation from almost Boolean formulæ in wMSO to step formulæ in wMSO′, not only modifies the set of constants used in the logic, but also expands exponentially the size of the formulæ.

Other fragments of wMSO are studied in [10]. A formula $\Xi \in \mathsf{wMSO}$ is $\forall$-*restricted* if every subformula $\forall x\, \xi$ is such that ξ is almost Boolean. The formula Ξ is $\wedge$-*restricted*[6] if every subformula $\xi \wedge \xi'$ is such that both ξ and ξ' are almost Boolean, or ξ or ξ' is Boolean.

We will relate these fragments with corresponding ones in wMSO′. A formula $\Xi \in \mathsf{wMSO}'$ is $\forall$-restricted if every subformula $\forall x\, \xi$ is such that ξ is a step formula. Formula Ξ is $\wedge$-restricted if $\wedge$ is used only in Boolean subformulæ.

Remark 21 Under hypothesis (**01**), thanks to the translation of Lemma 19, $\forall$-restricted (respectively, $\wedge$-restricted) formulæ of wMSO′ translate into equivalent $\forall$-restricted (respectively, $\wedge$-restricted) formulæ of wMSO. Conversely, using the translation of Lemma 20 for almost Boolean formulæ occurring just below a universal

[6] These formulæ are call strongly $\wedge$-restricted in [10], in contrast with a slightly less restrained fragment of $\wedge$-restricted formulæ.

quantification, we can translate $\forall$-restricted formulæ of wMSO into $\forall$-restricted formulæ of wMSO'. Moreover, starting from a $\wedge$-restricted formula Ξ of wMSO, we first apply the transformation of Lemma 20 to maximal almost Boolean formulæ which are not Boolean. We obtain a new formula Ξ' in which the remaining conjunctions which are not inside a Boolean subformula are of the form $\varphi \wedge \Xi_1$ or $\Xi_1 \wedge \varphi$ with φ a Boolean formula. We translate these formulæ into $\varphi \,?\, \Xi_1 : 0$. We may still have some maximal Boolean subformulæ φ which are not guards of an if-then-else operator. We replace these with $\varphi \,?\, 1 : 0$. We obtain an equivalent $\wedge$-restricted formula Ξ'' of wMSO'.

We now establish the relationship between $\mathsf{core\text{-}wMSO}$ and these fragments of wMSO'. The first direction does not require any additional hypotheses.

The semantic equivalence between a formula $\Phi \in \mathsf{core\text{-}wMSO}$ and a formula Ξ in wMSO' or wMSO, denoted $\Phi \equiv \Xi$, is defined by $\mathsf{free}(\Phi) = \mathsf{free}(\Xi) = V$ and $\mathsf{aggr}_{\mathsf{sv}}(\{\!|\Phi|\!\}_V(\overline{w})) = [\![\Xi]\!]_V(\overline{w})$ for all $\overline{w} \in \Sigma_V^+$.

Proposition 22 *Every formula Φ of $\mathsf{core\text{-}wMSO}(\Sigma, R)$ can be translated into an equivalent $\forall$- and $\wedge$-restricted formula $\widetilde{\Phi}$ of $\mathsf{wMSO}'(\Sigma, R)$.*

Proof Since $\mathsf{step\text{-}wMSO}$ formulæ are syntactically identical with step formulæ of wMSO', we simply set $\widetilde{\Psi} = \Psi$. For $\mathsf{core\text{-}wMSO}$ formulæ Φ, the translation is performed inductively by

$$\widetilde{\varphi \,?\, \Phi_1 : \Phi_2} = \varphi \,?\, \widetilde{\Phi_1} : \widetilde{\Phi_2} \qquad \widetilde{\mathbf{0}} = 0$$
$$\widetilde{\Phi_1 + \Phi_2} = \widetilde{\Phi_1} \vee \widetilde{\Phi_2} \qquad \widetilde{\textstyle\sum_x \Phi_1} = \exists x\, \widetilde{\Phi_1}$$
$$\widetilde{\textstyle\sum_X \Phi_1} = \exists X\, \widetilde{\Phi_1} \qquad \widetilde{\textstyle\prod_x \Psi} = \forall x\, \widetilde{\Psi}\,.$$

Notice that $0 \in R$ so that $\widetilde{\Phi}$ is always a $\forall$- and $\wedge$-restricted formula of $\mathsf{wMSO}'(\Sigma, R)$. Let Φ in $\mathsf{core\text{-}wMSO}$ and let $\overline{w} \in \Sigma_V^+$ with $\mathsf{free}(\Phi) \subseteq V$. If $\overline{w}$ is not valid then $\{\!|\Phi|\!\}_V(\overline{w}) = \emptyset$ and $[\![\widetilde{\Phi}]\!]_V(\overline{w}) = 0$. We conclude with $\mathsf{aggr}_{\mathtt{vm}}(\emptyset) = 0$. We assume now that $\overline{w} = (w, \sigma)$ is valid. We show by induction that $\mathsf{aggr}_{\mathsf{sv}}(\{\!|\Phi|\!\}_V(w, \sigma)) = [\![\widetilde{\Phi}]\!]_V(w, \sigma)$.

The case $\Phi = \varphi \,?\, \Phi_1 : \Phi_2$ is trivial. For $\mathbf{0}$, $+$, $\sum_x$ and $\sum_X$, the result follows from the fact that the aggregation operator $\mathsf{aggr}_{\mathtt{vm}} = \sum \mathsf{Val}$ is a monoid morphism from $(\mathbb{N}\langle R^\star\rangle, \uplus, \emptyset)$ to $(S, +, 0)$, i.e., $\mathsf{aggr}_{\mathtt{vm}}(\emptyset) = 0$ and for all multisets $A, B \in \mathbb{N}\langle R^\star\rangle$, we have

$$\mathsf{aggr}_{\mathtt{vm}}(A \uplus B) = \mathsf{aggr}_{\mathtt{vm}}(A) + \mathsf{aggr}_{\mathtt{vm}}(B)\,.$$

Finally, assume that $\Phi = \prod_x \Psi$ and $\mathsf{pos}(w) = \{1, \dots, n\}$. For every position $i \in \mathsf{pos}(w)$, there exists a constant $r_i = [\![\widetilde{\Psi}]\!]_{V \cup \{x\}}(w, \sigma[x \mapsto i]) \in S$ such that the abstract semantics is $\{\!|\Psi|\!\}_{V \cup \{x\}}(w, \sigma[x \mapsto i]) = \{\{r_i\}\}$. We obtain

$$\begin{aligned}\mathsf{aggr}_{\mathtt{vm}}\big(\{\!|\textstyle\prod_x \Psi|\!\}_V(w, \sigma)\big) &= \sum \mathsf{Val}\big(\{\{r_1 \cdots r_n\}\}\big)\\ &= \mathsf{Val}(r_1 \cdots r_n)\\ &= [\![\forall x\, \widetilde{\Psi}]\!]_V(w, \sigma)\,.\end{aligned}$$ □

We now consider the reverse translation from wMSO' to $\mathsf{core\text{-}wMSO}$. We first define the hypothesis on the weight domain allowing us to obtain the result.

A subset R of a product valuation monoid $(S, +, 0, \mathsf{Val}, \diamond, 1)$ is said to be *regular* if for every weight $r \in R$, there exists a sentence Φ_r of $\mathsf{core\text{-}wMSO}(\Sigma, S)$ such that

$$\mathsf{aggr}_{\mathtt{vm}}(\{\!|\Phi_r|\!\}(w)) = r$$

for every $w \in \Sigma^+$.[7] Notice that the formula Φ_r may use all possible constants in S, and not only the ones in R.

Example 23 In case of a valuation that is a λ-discounted sum, we may use the formula

$$\Phi_r = \textstyle\prod_x \mathsf{first}(x) \,?\, r : 0$$

with $\mathsf{first}(x)$ the MSO formula stating that x is the first position of the word. Indeed, Φ_r maps a word w of length n to $r + \lambda \cdot 0 + \lambda^2 \cdot 0 + \cdots + \lambda^{n-1} \cdot 0 = r$. Another example is when the valuation takes the average of the weights. Then, the formula

$$\Phi_r = \textstyle\prod_x r$$

is appropriate. Indeed, it maps w to $\frac{r + \cdots + r}{n} = r$. As last example, in every semiring S, S is a regular subset using the formula

$$\Phi_r = \textstyle\prod_x \mathsf{first}(x) \,?\, r : 1$$

and the fact that 1 is a unit for multiplication.

Proposition 24 *If the set R is regular in S, each $\forall$- and $\wedge$-restricted formula $\Xi \in \mathsf{wMSO}'(\Sigma, R)$ can be translated into an equivalent formula $\widetilde{\Xi} \in \mathsf{core\text{-}wMSO}(\Sigma, S)$.*

Proof Using the regularity of R, we give a translation of $\forall$- and $\wedge$-restricted formulæ of $\mathsf{wMSO}'(\Sigma, R)$ into $\mathsf{core\text{-}wMSO}(\Sigma, S)$:

$$\widetilde{r} = \Phi_r \qquad \widetilde{\varphi \,?\, \Xi_1 : \Xi_2} = \varphi \,?\, \widetilde{\Xi_1} : \widetilde{\Xi_2}$$
$$\widetilde{\Xi \vee \Xi'} = \widetilde{\Xi} + \widetilde{\Xi'} \qquad \widetilde{\forall x \Xi} = \textstyle\prod_x \Xi$$
$$\widetilde{\exists x \Xi} = \textstyle\sum_x \widetilde{\Xi} \qquad \widetilde{\exists X \Xi} = \textstyle\sum_X \widetilde{\Xi}$$

Notice again that a step formula in wMSO' is already a $\mathsf{step\text{-}wMSO}$ formula and does not require any translation. Using a proof by induction as in Proposition 22, it is immediate to check that Ξ and $\widetilde{\Xi}$ are equivalent. □

[7] In [10], the regularity is defined with respect to some weighted automata. Here, we prefer to use a purely logical definition. However, using Corollary 11-2., those two definitions are equivalent.

We will now translate the $\forall$-restricted fragment of wMSO' into $\mathsf{core\text{-}wMSO}$ (no restriction on $\wedge$). To this end, we first define on weight domains the hypothesis (**DC**) which states that $\diamond$ distributes over $+$, i.e., $r \diamond (s+t) = r \diamond s + r \diamond t$ and $(s+t) \diamond r = s \diamond r + t \diamond r$ for all $r, s, t \in S$, and $\diamond$ is Val-commutative, i.e., $\mathsf{Val}(r_1 \cdots r_n) \diamond \mathsf{Val}(s_1 \cdots s_n) = \mathsf{Val}((r_1 \diamond s_1) \cdots (r_n \diamond s_n))$ for all $r_1, \ldots, r_n, s_1, \ldots, s_n \in S$.

Proposition 25 *If the set R is regular in S and under hypothesis* (**DC**)*, every $\forall$-restricted formula Ξ of $\mathsf{wMSO}'(\Sigma, R)$ can be translated into an equivalent formula $\widetilde{\Xi}$ of $\mathsf{core\text{-}wMSO}(\Sigma, S)$.*

Proof We use the translation of Proposition 24, adding the following rule to translate conjunctions:

$$\widetilde{\Xi \wedge \Xi'} = \widetilde{\Xi} \diamond \widetilde{\Xi'} .$$

For all $\forall$-restricted formula Ξ of wMSO', $\widetilde{\Xi}$ is now a formula of $\diamond$-$\mathsf{core\text{-}wMSO}$, as introduced in Section 4.4. Checking that Ξ and $\widetilde{\Xi}$ are equivalent is performed as in Proposition 24 for all constructs but the conjunction. For the conjunction, we have to show that

$$\begin{aligned}&\mathsf{aggr}_{\mathtt{vm}}\{\!|\Phi_1 \diamond \Phi_2|\!\}(w, \sigma)\\ &\qquad = \mathsf{aggr}_{\mathtt{vm}}\{\!|\Phi_1|\!\}(w, \sigma) \diamond \mathsf{aggr}_{\mathtt{vm}}\{\!|\Phi_2|\!\}(w, \sigma)\end{aligned}$$

i.e., that the semantics of $\diamond$ is compositional. This holds since, for all $n > 0$ and $A, B \in \mathbb{N}\langle S^n \rangle$, we have

$$\begin{aligned}&\mathsf{aggr}_{\mathtt{vm}}(A \diamond B)\\ &\quad = \textstyle\sum\{\{\mathsf{Val}(\gamma) \mid \gamma \in A \diamond B\}\}\\ &\quad = \textstyle\sum\{\{\mathsf{Val}(\alpha \diamond \beta) \mid \alpha \in A, \beta \in B\}\}\\ &\quad = \textstyle\sum\{\{\mathsf{Val}(\alpha) \diamond \mathsf{Val}(\beta) \mid \alpha \in A, \beta \in B\}\}\\ &\quad = (\textstyle\sum\{\{\mathsf{Val}(\alpha) \mid \alpha \in A\}\}) \diamond (\sum\{\{\mathsf{Val}(\beta) \mid \beta \in B\}\})\\ &\quad = \mathsf{aggr}_{\mathtt{vm}}(A) \diamond \mathsf{aggr}_{\mathtt{vm}}(B) .\end{aligned}$$

By Proposition 16, we may finally translate the $\diamond$-$\mathsf{core\text{-}wMSO}$ formula $\widetilde{\Xi}$ into an equivalent $\mathsf{core\text{-}wMSO}$ formula. □

Finally using Corollary 10, the previous propositions and Remark 21 linking fragments of wMSO and wMSO', we obtain

Theorem 26 *Suppose that S is regular. The following formalisms are expressively equivalent:*

- *S-weighted automata over alphabet Σ;*
- $\mathsf{core\text{-}wMSO}(\Sigma, S)$*;*
- *$\forall$- and $\wedge$-restricted* $\mathsf{wMSO}'(\Sigma, S)$
- *$\forall$-restricted* $\mathsf{wMSO}'(\Sigma, S)$ *if* (**DC**)*;*
- *$\forall$- and $\wedge$-restricted* $\mathsf{wMSO}(\Sigma, S)$ *if* (**01**)*;*
- *$\forall$-restricted* $\mathsf{wMSO}(\Sigma, S)$ *if* (**DC**) *and* (**01**)*.*

Notice that the sets of constants in logics and automata is always taken to be S in order to simplify the statements. For the same reason, the regularity condition is on the full set of weights S. Previous propositions give more precise results on how the set of constants change along the translations.

5.2 wMSO over valuation structures

The syntax of the logic wMSO over valuation structures [11] is the same as for valuation monoids defined before. The difference comes only from the semantics. Consider a valuation structure (U, Val, S, F). As previously, a binary operator $\diamond\colon U^2 \to U$, as well as a constant 1 in R, are necessary to define the semantics of conjunction and of the Boolean formulæ. Such a tuple $(U, \mathsf{Val}, S, F, \diamond, 1)$ is called a *product valuation structure*. Notice that, contrary to product valuation monoids, the element 1 lies in U and not in S.

The easiest way to define the semantics in that case is to rely on the case of product valuation monoids. Indeed, notice that $(\mathbb{N}\langle U \rangle, \uplus, \emptyset, \mathsf{Val}, \diamond, \{\{1\}\})$ is a product valuation monoid. The diamond operator is lifted from the valuation structure to multisets pointwisely. The valuation of a sequence $(A_i)_{1 \leqslant i \leqslant n}$ of multisets is the multiset $\{\{\mathsf{Val}(a_1 \cdots a_n) \mid a_i \in A_i\}\}$. Hence, this allows us to define the semantics of formulæ Ξ of wMSO in two steps. First, a semantics $\langle \Xi \rangle_V(\overline{w})$ in the product valuation monoid $\mathbb{N}\langle U \rangle$ as defined in Table 4. Then, the final interpretation $F(\langle \Xi \rangle_V(\overline{w}))$ given by F. Notice the difference between the semi-interpreted semantics $\langle \cdot \rangle_V(\overline{w})$ and our abstract semantics $\{\!| \cdot |\!\}_V(\overline{w})$ in the case of universal quantifications.

Theorem 26 also holds for product valuation structures with the hypotheses simplified as follows:

- the regularity condition on $\mathbb{N}\langle U \rangle$ is fulfilled when for every constant $r \in U$, there exists a sentence Φ_r of $\mathsf{core\text{-}wMSO}(\Sigma, U)$ such that $\{\!|\Phi_r|\!\}(w) = \{\{r\}\}$ for all $w \in \Sigma^+$;
- hypothesis (**01**) is fulfilled when $1 \diamond r = r \diamond 1 = r$ for all $r \in U$, and $\mathsf{Val}(1 \cdots 1) = 1$. In particular, conditions on the zero element $\emptyset \in \mathbb{N}\langle U \rangle$ are trivially verified because of the pointwise definition of $\diamond$ and Val over multisets;
- hypothesis (**DC**) is fulfilled when $\mathsf{Val}(r_1 \cdots r_n) \diamond \mathsf{Val}(s_1 \cdots s_n) = \mathsf{Val}((r_1 \diamond s_1) \cdots (r_n \diamond s_n))$ for all $r_1, \ldots, r_n, s_1, \ldots, s_n \in U$. In particular, $\diamond$ always distributes over $\uplus$ in $\mathbb{N}\langle U \rangle$.

Indeed, Theorem 26 for product valuation structures is obtained by first applying Theorem 26 to the product valuation monoid $\mathbb{N}\langle U \rangle$ and by then applying the evaluator function F.

6 Extensions to ranked and unranked trees

In this section, we show how to extend the equivalence between weighted automata and core-wMSO to other structures, namely ranked and unranked trees. We will primarily use a semantics in multisets of weight trees (instead of weight sequences). Then, we may apply an aggregation operator to recover a more concrete semantics. This approach allows us to infer results for semirings [13,14] and also for tree valuation monoids [7].

There are two main ingredients allowing us to prove the equivalence between core-wMSO and weighted automata. First, in the Boolean case, we should have an equivalence between unambiguous (or deterministic) automata and MSO logic. This equivalence is known for many structures such as words [3,15,22], ranked trees [21], unranked trees [5], etc. Second, the computation of the weight of a run ρ of an automaton, and the evaluation of a product formula $\prod_x \Psi$ should be based on the same mechanism. For words and valuation monoids (or valuation structures), it is the valuation of a *sequence* of weights. This is why we used an abstract semantics in the semiring of multisets of weight *sequences*. For trees and tree valuation monoids, the valuation takes a tree of weights as input and returns a value in the monoid. Hence, we use multisets of weight trees as abstract semantics. Note that, multisets of weight trees form a monoid but not a semiring.[8]

6.1 Ranked and unranked trees

Let $\mathbb{P} = \{1, 2, \ldots\}$ be the set of positive natural numbers and $\mathbb{P}^\star$ be the set of finite words over $\mathbb{P}$. A tree domain $\mathcal{B}$ is a finite, nonempty subset of $\mathbb{P}^\star$ such that if $u \cdot i \in \mathcal{B}$ with $u \in \mathbb{P}^\star$ and $i \in \mathbb{P}$, then we have $u, u \cdot 1, \ldots, u \cdot (i-1) \in \mathcal{B}$. An unranked tree over a set Σ of labels (Σ-tree) is a partial mapping $t \colon \mathbb{P}^\star \to \Sigma$ with a $\mathrm{dom}(t) \subseteq \mathbb{P}^\star$ being a tree domain. The arity of a node $u \in \mathrm{dom}(t)$ is $\mathsf{ar}(u) = \max\{i \in \mathbb{P} \mid u \cdot i \in \mathrm{dom}(t)\}$, with $\max \emptyset = 0$. We denote by UT_Σ the set of unranked Σ-trees.

A ranked alphabet is a pair (Σ, rk) with Σ a finite alphabet and $\mathsf{rk} \colon \Sigma \to \mathbb{N}$ giving the rank of each symbol in Σ. For each rank m, we denote by $\Sigma^{(m)}$ the set of symbol $a \in \Sigma$ with $\mathsf{rk}(a) = m$. A ranked tree over (Σ, rk) is a Σ-tree t such that for each $u \in \mathrm{dom}(t)$ we have $\mathsf{ar}(u) = \mathsf{rk}(t(u))$.

An *(unranked) tree automaton* over Σ is a tuple $\mathcal{A} = (Q, \Delta, F)$ with Q a nonempty finite set of states, $F \subseteq Q$ the set of accepting states, and $\Delta \subseteq \mathsf{Reg}(Q^\star) \times \Sigma \times Q$ a *finite* set of transitions, where $\mathsf{Reg}(Q^\star)$ denotes the set of regular languages over alphabet Q. Here we may (finitely) represent regular languages by finite-state automata or regular expressions over the alphabet Q.

A run of $\mathcal{A}$ over a Σ-tree t is a Δ-tree ρ with domain $\mathrm{dom}(\rho) = \mathrm{dom}(t)$ and such that for all $u \in \mathrm{dom}(t)$ we have $\rho(u) = (L(u), t(u), q(u)) \in \Delta$ and $q(u \cdot 1) \cdots q(u \cdot \mathsf{ar}(u)) \in L(u)$. The run is accepting if $q(\varepsilon) \in F$.[9]

The language $L(\mathcal{A})$ accepted by the tree automaton $\mathcal{A}$ is the set of Σ-trees on which there exists at least one accepting run.

Remark 27 In case of a ranked alphabet, we recover the classical ranked tree automata with $\Delta \subseteq \bigcup_m Q^m \times \Sigma^{(m)} \times Q$.

A tree automaton $\mathcal{A}$ is said to be deterministic if for all distinct transitions (L, a, q) and (L', a, q') in Δ, we have $L \cap L' = \emptyset$. Equivalently, for every word $\pi \in Q^\star$ and $a \in \Sigma$, there is at most one transition $(L, a, q) \in \Delta$ such that $\pi \in L$. Hence, every tree t admits at most one run. It is well known that tree automata can be determinised (see, e.g., [5]).

6.2 Weighted automata over trees

An *R-weighted (unranked) tree automaton* over Σ is a tuple $\mathcal{A} = (Q, \Delta, \mathsf{wgt}, F)$ with (Q, Δ, F) a tree automaton and $\mathsf{wgt} \colon \Delta \to R$ associating a weight to every transition.

The weight tree arising from a run ρ of $\mathcal{A}$ over a Σ-tree t is the R-tree $\mathsf{wgt} \circ \rho$ mapping each $u \in \mathrm{dom}(t)$ to $\mathsf{wgt}(\rho(u)) \in R$. The abstract semantics of an R-weighted tree automaton $\mathcal{A}$ is a multiset of weight trees. For all trees $t \in \mathsf{UT}_\Sigma$, we define

$$\{\!|\mathcal{A}|\!\}(t) = \{\!\{\mathsf{wgt} \circ \rho \mid \rho \text{ is an accepting run of } \mathcal{A} \text{ on } t\}\!\}\,.$$

Hence, our abstract semantics lives in the commutative monoid $\mathbb{N}\langle\mathsf{UT}_R\rangle$ of multisets of R-trees. Then, we may use an aggregation operator $\mathsf{aggr} \colon \mathbb{N}\langle\mathsf{UT}_R\rangle \to S$ to obtain a concrete semantics in a possibly different weight structure S:

$$[\![\mathcal{A}]\!](t) = \mathsf{aggr}(\{\!|\mathcal{A}|\!\}(t))\,.$$

Example 28 (Weighted automata over semirings) In the classical setting, the set R of weights is a subset of a semiring $(S, +, \times, 0, 1)$. The *value* of a run ρ of $\mathcal{A}$

[8] The product in the semiring of weight sequences is based on the concatenation of sequences. There is no natural counterpart for trees.

[9] Equivalently, we may assume that transitions are disjoint: if $(L, a, q), (L', a, q) \in \Delta$ then $L = L'$ or $L \cap L' = \emptyset$. In this case, a run can be defined as a simpler Q-tree ρ with $\mathrm{dom}(\rho) = \mathrm{dom}(t)$ and such that for all $u \in \mathrm{dom}(t)$ there is a (unique) transition $(L, t(u), \rho(u)) \in \Delta$ with $\rho(u \cdot 1) \cdots \rho(u \cdot \mathsf{ar}(u)) \in L$.

over a Σ-tree t is the product of the weights in the R-tree $\mathsf{wgt} \circ \rho$. Since the semiring is not necessarily commutative, we have to specify the order in which this product is computed. Classically, we choose the postfix order. Formally, given an R-tree ν, the product $\prod(\nu) = \mathsf{Prod}(\nu, \varepsilon)$ is computed bottom-up: for all $u \in \mathrm{dom}(\nu)$ we set

$$\mathsf{Prod}(\nu, u) = \mathsf{Prod}(\nu, u{\cdot}1) \times \cdots \times \mathsf{Prod}(\nu, u{\cdot}\mathsf{ar}(u)) \times \nu(u)\,.$$

Note that, if u is a leaf then $\mathsf{Prod}(\nu, u) = \nu(u)$. As for words, the mapping $\prod\colon \mathsf{UT}_R \to S$ can be lifted to a mapping $\prod\colon \mathbb{N}\langle\mathsf{UT}_R\rangle \to \mathbb{N}\langle S\rangle$. Then, the semantics is defined as always by summing the values of the accepting runs: $[\![\mathcal{A}]\!](t) = \sum_\rho \prod(\mathsf{wgt}(\rho))$ where the sum ranges over accepting runs ρ of $\mathcal{A}$ over the Σ-tree t. Therefore, the classical case of semirings is obtained from the abstract semantics with the aggregation operator

$$\mathsf{aggr}_{\mathtt{sr}}(A) = \sum \prod(A) = \textstyle\sum_{\nu \in A} \prod(\nu)\,.$$

In the case of a ranked alphabet, we recover the definition of [13] of weighted tree automata. The comparison with the weighted unranked tree automata of [14] is not as easy, at least over non-commutative semirings. We believe that over commutative semirings, our model is equivalent to the weighted unranked tree automata of [14]. The situation is different over non-commutative semirings. Our definition is best motivated by considering words as special cases of trees. There are two ways to inject words in unranked trees, as shown in Fig. 2: either in a horizontal way (a root with children representing the word from left to right), or a vertical way (unary nodes followed by a leaf, the word being read from bottom to top). With some easy encodings, we may see that our model of weighted unranked tree automata is a conservative extension of weighted word automata, both for the horizontal and the vertical injections of words. Moreover, our approach allows us to obtain the equivalence between automata and logic for arbitrary semirings (even non-commutative ones), as stated in Theorem 30.

In contrast, the model of [14] is not a conservative extension of weighted word automata for the horizontal injection. This is witnessed by an example given in [14, Theorem 6.10], that we now recall. In the (non-commutative) semiring $(\mathfrak{P}(\{p,q\}^\star), \cup, \cdot, \emptyset, \{\varepsilon\})$, with two distinct letters p and q, we consider $f\colon \mathsf{UT}_\Sigma \to \mathfrak{P}(\{p,q\}^\star)$ the tree series mapping every tree t composed of a root directly followed by n children ($n \in \mathbb{N}$) to the language $\{p^n q^n\}$, and every other tree to $\emptyset$. The model of weighted unranked tree automata we have chosen can not recognise this tree series. However, the model of automata described in [14] is able to recognise this

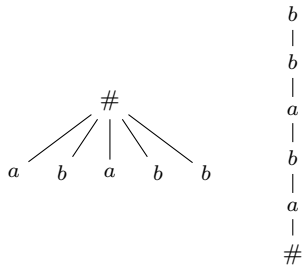


Fig. 2 Horizontal and vertical injection of the word *ababb* into trees

tree series. The main difference between the two models, that explains this discrepancy, is the way weights are assigned during the computation of the automaton. Whereas we have decided to assign weights to transitions of the unranked tree automaton, keeping a Boolean regular (hedge) language to determine whether a transition is enabled, Droste and Vogler decided instead to use a weighted (hedge) automaton when reading the sequence of states of the children. Then, to each position in the tree domain is associated the weight of the (hedge) automaton reading the sequence of states of the children. The semantics over a tree is given by a depth-first left-to-right product of those weights (first the weight of the children from left to right, and then the weight of the parent).

Example 29 (Tree valuation monoids) As for words, extensions of weighted automata to more general weight domains have been considered. Following [7], a tree valuation monoid is a tuple $(S, +, 0, \mathsf{Val})$ where $(S, +, 0)$ is a commutative monoid and $\mathsf{Val}\colon \mathsf{UT}_S \to S$ is a valuation function from S-trees to S. The value of a run ρ is now computed by applying this valuation function to the R-tree $\mathsf{wgt} \circ \rho$. The final semantics is obtained as above by summing the values of accepting runs. Therefore, the semantics in tree valuation monoids is obtained from the abstract semantics with the aggregation operator

$$\mathsf{aggr}_{\mathtt{vm}}(A) = \sum \mathsf{Val}(A) = \sum_{\nu \in A} \mathsf{Val}(\nu)\,.$$

For instance, when $(S, +, \times, 0, 1)$ is a semiring, we obtain a tree valuation monoid with the postfix product $\prod$ defined in Example 28. We refer to [7] for other examples of weighted ranked tree automata, including the interesting case of multi-operator monoids which is also studied in [16]. A further extension for unranked trees has recently been considered in [8].

Further extensions like *tree valuation structures*—where the sum in tree valuation monoids is replaced by a more general operator F as for words—are also

possible, though not considered in the literature so far. Our results will apply in this context as well.

6.3 core-wMSO over trees

There are only very few changes needed to lift the logic core-wMSO defined in Section 4 to ranked or unranked trees.

On one hand, we need to change the atomic predicates of the Boolean fragment MSO to the corresponding ones over trees. For unranked trees, we use $\mathsf{child}(x,y)$ which says that under the current valuation σ we have $\sigma(y) = \sigma(x)\cdot i$ for some $i \in \mathbb{P}$, and $\mathsf{nextsibling}(x,y)$ for $\sigma(x) = u\cdot i$ and $\sigma(y) = u\cdot(i+1)$ for some $u \in \mathbb{P}^\star$ and $i \in \mathbb{P}$. For ranked trees, we only use $\mathsf{child}_i(x,y)$ which means that y is the ith child of x ($\sigma(y) = \sigma(x)\cdot i$). Thanks to [21] for ranked trees and [5] for unranked trees, we know that MSO has the same expressive power as deterministic (bottom-up) tree automata.

On the other hand, we need to give the semantics of core-wMSO formulæ in the setting of trees. Once again, we will use the classical encoding of a pair (t,σ) consisting of a Σ-tree t and a valuation σ from a set V of variables to the domain of t, as a Σ_V-tree $\bar{t}$. The notion of validity of such a Σ_V-tree $\bar{t}$ is defined as for words. The semantics of formulæ of step-wMSO maps Σ_V-trees to $\mathbb{N}\langle R\rangle$, and is defined as in Table 2 using $t,\sigma \models \varphi$ for MSO formulæ φ. For core-wMSO, in case of a valid Σ_V-tree $\bar{t} = (t,\sigma)$, the semantics is defined as in Table 3 except for the operator $\prod_x$ for which we let

$$\{\!|\textstyle\prod_x \Psi|\!\}_V(t,\sigma) = \{\{t'\}\}$$

where t' is the R-tree with the same domain as t such that for all $u \in \mathrm{dom}(t)$ we have $\{\!|\Psi|\!\}_V(t,\sigma[x\mapsto u]) = \{\{t'(u)\}\}$.

Then, we obtain the following result:

Theorem 30 *For each R-weighted tree automaton $\mathcal{A}$ over alphabet Σ, we can effectively construct a sentence $\Phi_\mathcal{A}$ in* core-wMSO(Σ,R)*, such that $\{\!|\mathcal{A}|\!\}(t) = \{\!|\Phi_\mathcal{A}|\!\}(t)$ for all Σ-trees t.*

For each sentence Φ in core-wMSO(Σ,R)*, we can effectively construct an R-weighted tree automaton $\mathcal{A}_\Phi$ over Σ such that $\{\!|\Phi|\!\}(t) = \{\!|\mathcal{A}_\Phi|\!\}(t)$ for all Σ-trees t.*

Proof The proof of Theorem 9 given in Section 4.5 may easily be adapted to the case of trees. In particular, Lemma 18 can be proved *mutatis mutandis*.

Then, in the translation from automata to the logic, only the formula $\mathsf{run}(\overline{X})$ has to be modified to check that $\overline{X}$ encodes a run of $\mathcal{A}$ over a given tree t. For that formula, we still have to check that $\overline{X}$ is a partition of the positions of t, that the root is labelled with a transition $\delta \in \mathsf{Reg}(Q^\star)\times\Sigma\times F$, and that for each node u of t labelled with transition $\delta = (L,a,q)$, the sequence $q_1\cdots q_{\mathsf{ar}(u)} \in L$ where (L_i,a_i,q_i) is the transition associated with the child $u\cdot i$ of u. This last condition is checked with the help of an MSO formula over words equivalent to the regular language L, replacing the linear order $\leqslant$ over words with the reflexive and transitive closure of the sibling relation $\mathsf{nextsibling}(x,y)$.[10] Notice also that the subformula $\prod_x \mathsf{weight}(x,\overline{X})$ in $\Phi_\mathcal{A}$ correctly computes $\{\{\mathsf{wgt}\circ\rho\}\}$ where ρ is the run encoded by $\overline{X}$.

For the translation from logic to automata, there are also some minor adaptations to tree automata. For the projection, let us explain how to construct an automaton $\mathcal{A}_\Phi$ from $\mathcal{A}_{\Phi_1}$ where $\Phi = \sum_X \Phi_1$. As for the word case, we transfer the alphabet component for X of $\mathcal{A}_{\Phi_1}$ in the state of $\mathcal{A}_\Phi$. We denote by π_1 the projection from $Q\times\{0,1\}$ to Q. Then, if $\mathcal{A}_{\Phi_1} = (Q,\Delta,\mathsf{wgt},F)$ then $\mathcal{A}_\Phi = (Q\times\{0,1\},\Delta',\mathsf{wgt}',F\times\{0,1\})$ with

$$(\pi_1^{-1}(L),a,(q,j)) \in \Delta' \text{ iff } (L,(a,j),q) \in \Delta$$

with weights inherited by

$$\mathsf{wgt}'(\pi_1^{-1}(L),a,(q,j)) = \mathsf{wgt}(L,(a,j),q)\,.$$

A similar modification is made for $\sum_x$ in order to check during the computation that exactly one 1 appears in the x component. We use the second projection π_2 from $Q\times\{0,1\}$ to $\{0,1\}$. Then, transitions are given by

$$(\pi_1^{-1}(L)\cap\pi_2^{-1}(0^\star),a,(q,j)) \in \Delta' \text{ if } (L,(a,j),q)\in\Delta$$
$$(\pi_1^{-1}(L)\cap\pi_2^{-1}(0^\star10^\star),a,(q,1)) \in \Delta' \text{ if } (L,(a,0),q)\in\Delta$$

and weights are inherited as before.

Next, for the if-then-else operator, we use the translation of a Boolean MSO formula into a deterministic tree automaton [5]. The cartesian product of a deterministic tree automaton and a weighted tree automaton can be adapted easily.

Finally, the construction for $\Phi = \prod_x \Psi$ can be easily adapted. Again, we assume that Ψ is a formula in set-step-wMSO. The tree automaton $\mathcal{A}_\Phi$ has a single state q which is accepting. For every letter $(a,\tau) \in \Sigma_{\mathsf{free}(\Phi)}$ there is a transition $(\{q\}^\star,(a,\tau),q)$ with weight $\Psi(\tau)$. The automaton $\mathcal{A}_\Phi$ is deterministic and complete. It has a single run ρ on each $\Sigma_{\mathsf{free}(\Phi)}$-tree $\bar{t} = (t,\sigma)$ and for all nodes $u \in \mathrm{dom}(t)$ we have $\{\!|\Psi|\!\}(t,\sigma[x\mapsto u]) = \{\{\mathsf{wgt}(\rho(u))\}\}$. We deduce that $\{\!|\mathcal{A}_\Phi|\!\}(\bar{t}) = \{\{\mathsf{wgt}\circ\rho\}\} = \{\!|\Phi|\!\}(\bar{t})$. □

[10] An even simpler formula can be obtained in the simpler setting of ranked trees.

7 Conclusion

We proved the meta-theorem relating weighted automata and core-wMSO at the level of multisets of weight structures for words and trees. However, the definitions and techniques developed in this article can easily be adapted for other structures like nested words, Mazurkiewicz traces, etc. The logical equivalence between restricted wMSO and core-wMSO at the concrete level is established for words in Section 5. An analogous result can be obtained for trees with a similar logical reasoning. In particular, this allows for an extension to trees of the valuation structures of [11].

In this article, our meta-theorem is only stated and proved for finite structures. At the level of the concrete semantics, equivalences between weighted automata and weighted logics have been extended to infinite structures, such as words or trees over semirings [6], valuation monoids [10] or valuation structures [11]. An extension of our meta-theorem to infinite structures capturing these results is a natural open problem. Finite multisets of weight structures are not adequate anymore since automata may exhibit infinitely many runs on a given input structure. The abstract semantics should ideally distinguish between countably many runs or uncountably many runs.

References

1. Albert, J., Kari, J.: Digital image compression. In: W. Kuich, H. Vogler, M. Droste (eds.) Handbook of Weighted Automata, EATCS Monographs in Theoretical Computer Science, chap. 5, pp. 445–472. Springer (2009)
2. Bollig, B., Gastin, P.: 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, pp. 18–38. Springer (2009)
3. Büchi, J.R.: Weak second-order arithmetic and finite automata. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik **6**, 66–92 (1960)
4. Černý, P., Chatterjee, K., Henzinger, T.A., Radhakrishna, A., Singh, R.: Quantitative synthesis for concurrent programs. In: Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11), *Lecture Notes in Computer Science*, vol. 6806, pp. 243–259. Springer (2011)
5. Comon-Lundh, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications (2008). URL http://tata.gforge.inria.fr/
6. Droste, M., Gastin, P.: Weighted automata and weighted logics. Theoretical Computer Science **380**(1-2), 69–86 (2007)
7. Droste, M., Götze, D., Märcker, S., Meinecke, I.: Weighted tree automata over valuation monoids and their characterization by weighted logics. In: Algebraic Foundations in Computer Science, *Lecture Notes in Computer Science*, vol. 7020, pp. 30–55. Springer (2011)
8. Droste, M., Heusel, D., Vogler, H.: Weighted unranked tree automata over tree valuation monoids and their characterization by weighted logics. In: Proceedings of the 6th International Conference on Algebraic Informatics (CAI'15), *Lecture Notes in Computer Science*, vol. 9270, pp. 90–102. Springer (2015)
9. Droste, M., Kuich, W.: Semirings and formal power series. In: M. Droste, W. Kuich, H. Vogler (eds.) Handbook of Weighted Automata, EATCS Monographs in Theoretical Computer Science, chap. 1, pp. 3–27. Springer (2009)
10. Droste, M., Meinecke, I.: Weighted automata and weighted MSO logics for average and long-time behaviors. Information and Computation **220-221**, 44–59 (2012)
11. Droste, M., Perevoshchikov, V.: Multi-weighted automata and MSO logic. In: Proceedings of the 8th International Computer Science Symposium in Russia (CSR'13), *Lecture Notes in Computer Science*, vol. 7913, pp. 418–430. Springer (2013)
12. Droste, M., Perevoshchikov, V.: A Nivat theorem for weighted timed automata and weighted relative distance logic. In: Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP'14), *Lecture Notes in Computer Science*, vol. 8573, pp. 171–182. Springer (2014)
13. Droste, M., Vogler, H.: Weighted tree automata and weighted logics. Theoretical Computer Science **366**(3), 228–247 (2006)
14. Droste, M., Vogler, H.: Weighted logics for unranked tree automata. Theory of Computing Systems **48**, 23–47 (2011)
15. Elgot, C.C.: Decision problems of finite automata design and related arithmetics. Transactions of the American Mathematical Society **98**, 21–52 (1961)
16. Fülöp, Z., Stüber, T., Vogler, H.: A Büchi-like theorem for weighted tree automata over multioperator monoids. Theory of Computing Systems **50**, 241–278 (2012)
17. Knight, K., May, J.: Applications of weighted automata in natural language processing. In: M. Droste, W. Kuich, H. Vogler (eds.) Handbook of Weighted Automata, EATCS Monographs in Theoretical Computer Science, chap. 14, pp. 555–579. Springer (2009)
18. Kuich, W., Salomaa, A.: Semirings, Automata and Languages. EATCS Monographs in Theoretical Computer Science. Springer-Verlag (1985)
19. Perevoshchikov, V.: Multi-weighted automata models and quantitative logics. Ph.D. thesis, Universität Leipzig (2015)
20. Schützenberger, M.P.: On the definition of a family of automata. Information and Control **4**, 245–270 (1961)
21. Thatcher, J.W., Wright, J.B.: Generalized finite automata theory with an application to a decision problem of second-order logic. Mathematical Systems Theory **2**(1), 57–81 (1968)
22. Trakhtenbrot, B.A.: Finite automata and logic of monadic predicates. Doklady Akademii Nauk SSSR **149**, 326–329 (1961)