Weighted Operator Precedence Languages*

Manfred Droste[1], Stefan Dück[1], Dino Mandrioli[2], and Matteo Pradella[2,3]

1 **Institute of Computer Science, Leipzig University, D-04109 Leipzig, Germany**
{droste,dueck}@informatik.uni-leipzig.de
2 **Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Piazza Leonardo Da Vinci 32, 20133 Milano, Italy**
{dino.mandrioli,matteo.pradella}@polimi.it
3 **IEIIT, Consiglio Nazionale delle Ricerche, via Ponzio 34/5, 20133 Milano, Italy**

Abstract

In the last years renewed investigation of operator precedence languages (OPL) led to discover important properties thereof: OPL are closed with respect to all major operations, are characterized, besides the original grammar family, in terms of an automata family (OPA) and an MSO logic; furthermore they significantly generalize the well-known visibly pushdown languages (VPL). In another area of research, quantitative models of systems are also greatly in demand. In this paper, we lay the foundation to marry these two research fields. We introduce weighted operator precedence automata and show how they are both strict extensions of OPA and weighted visibly pushdown automata. We prove a Nivat-like result which shows that quantitative OPL can be described by unweighted OPA and very particular weighted OPA. In a Büchi-like theorem, we show that weighted OPA are expressively equivalent to a weighted MSO-logic for OPL.

1998 ACM Subject Classification F.1.1 Models of Computation, F.4.1 Mathematical Logic, F.4.3 Formal Languages

Keywords and phrases Quantitative automata, operator precedence languages, input-driven languages, visibly pushdown languages, quantitative logic.

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.31

1 Introduction

In the long history of formal languages the family of regular languages (RL) has always played a major role: thanks to its simplicity and naturalness, it enjoys many positive mathematical properties which have been thoroughly exploited in disparate practical applications; among them, those of main interest in this paper are the following:

- RL have been characterized in terms of various mathematical logics. Originally, Büchi, Elgot, and Trakhtenbrot [6, 18, 34] independently developed a monadic second order (MSO) logic defining exactly the RL family. This work has been followed by many further results; in particular those that exploited weaker but simpler logics such as first-order, propositional, and temporal ones culminated in the breakthrough of model checking to support automatic verification [28, 19, 7].
- Weighted RL have been introduced by Schützenberger in [32]: by assigning a weight in a suitable algebra to each language word, we may specify several attributes of the word, e.g.,

* This work was supported by Deutsche Forschungsgemeinschaft (DFG) Graduiertenkolleg 1763 (QuantLA).

relevance, probability, etc. Much research then followed and extended Schützenberger's original work in various directions, cf. the books [4, 17, 23, 31, 13].

Unfortunately, all families with greater expressive power than RL –typically context-free languages (CFL), which are the most widely used family in practical applications– pay a price in terms of algebraic and logic properties and, consequently, of possible tools supporting their automatic analysis. For instance, for CFL, the containment problem is undecidable.

What was not possible for general CFL, however, has been possible for important subclasses of this family, which together we call *structured CFL*. Informally, by this term we denote those CFL where the syntactic tree-structure of their words is immediately "visible" in the words themselves. Two first equivalent examples of such families are parenthesis languages [27], which are generated by grammars whose right hand sides are enclosed within pairs of parentheses, and tree-automata [33], which generalize finite state machines (FSM) from the recognition of linear strings to tree-like structures. Among the many variations of parenthesis languages the recent family of *input-driven languages* [29, 35], alias *visibly pushdown languages (VPL)* [2], has received much attention in recent literature. For most of these structured CFL, including VPL, the relevant algebraic properties of RL still hold [2]. One of the most noticeable results has been a characterization of VPL in terms of a MSO logic that is a natural extension of Büchi's original one for RL [24, 2].

This fact has suggested to extend the investigation of weighted RL to various cases of structured languages. The result of such a fertile approach is a rich collection of *weighted logics*, first studied by Droste and Gastin [11], associated with *weighted tree automata* [16] and weighted extensions of VPA (the automata recognizing VPL) [26].

In an originally unrelated way *operator precedence languages (OPL)* have been defined and studied in two phases temporally separated by four decades. In his seminal work [20] Floyd was inspired by the precedence of multiplicative operations over additive ones in the execution of arithmetic expressions and extended such a relation to the whole input alphabet in such a way that it could drive a deterministic parsing algorithm that builds the syntax tree of any word that reflects the word's semantics; Fig. 1 and Section 2 give an intuition of how an OP grammar generates arithmetic expressions and assigns them a natural structure.

OPL do not cover all deterministic CFL, but they enjoy a distinguishing property, not possessed by general deterministic CFL, which we can intuitively describe as "*OPL are input driven but not visible*". They can be claimed as *input-driven* since the parsing actions on their words –whether to push or pop– depend exclusively on the input alphabet and on the relation defined thereon, but their structure is *not visible* in their words: e.g, they can include unparenthesized expressions where the precedence of multiplicative operators over additive ones is explicit in the syntax trees but hidden in their frontiers (see Fig. 1). Furthermore, unlike other structured CFL, OPL include deterministic CFL that are not real-time [25].

This recent remark suggested to resume their investigation systematically at the light of the recent technological advances and related challenges. Such a renewed investigation led to prove their closure under all major language operations [8] and to characterize them, besides Floyd's original grammars, in terms of an appropriate class of pushdown automata (OPA) and in terms of a MSO logic which is a fairly natural but not trivial extension of the previous ones defined to characterize RL and VPL [25]. Thus, OPL enjoy the same nice properties of RL and many structured CFL but considerably extend their applicability by breaking the barrier of visibility and real-time push-down recognition.

In this paper we join the two research fields above, namely we introduce *weighted OPL* and show that they are able to model system behaviors that cannot be specified by means of less powerful weighted formalisms such as weighted VPL. For instance, one might be

interested in the behavior of a system which handles calls and returns but is subject to some emergency interrupts. Then it is important to evaluate how critically the occurrences of interrupts affect the normal system behavior, e.g., by counting the number of pending calls that have been preempted by an interrupt. As another example we consider a system logging all hierarchical calls and returns over words where this structural information is hidden. Depending on changing exterior factors like energy level, such a system could decide to log the above information in a selective way.

Our main contributions in this paper are the following.

- The model of *weighted OPA*, which have semiring weights at their transitions, significantly increases the descriptive power of previous weighted extensions of VPA, and has desired closure and robustness properties.
- For arbitrary semirings, there is a relevant difference in the expressive power of the model depending on whether it permits assigning weights to pop transitions or not. For commutative semirings, however, weights on pop transitions do not increase the expressive power of the automata. The difference in descriptive power between weighted OPA with arbitrary weights and without weights at pop transitions is due to the fact that OPL may be non-real-time and therefore OPA may execute several pop moves without advancing their reading heads.
- An extension of the classical result of Nivat [30] to weighted OPL. This robustness result shows that the behaviors of weighted OPA without weights at pop transitions are exactly those that can be constructed from weighted OPA with only one state, intersected with OPL, and applying projections which preserve the structural information.
- A weighted MSO logic and, for arbitrary semirings, a Büchi-Elgot-Trakhtenbrot-Theorem proving its expressive equivalence to weighted OPA without weights at pop transitions. As a corollary, for commutative semirings this weighted logic is equivalent to weighted OPA including weights at pop transitions.

Various possibilities arise for future research concerning theory and applications of our new model which will be discussed in the conclusion. The full version of this paper [10] provides all omitted technicalities and more explanatory comments and examples.

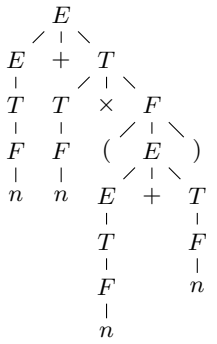
2 Preliminaries

Consider the CFG of Fig. 1 (left) and the syntax tree (center) which makes the structure of its frontier $n+n\times(n+n)$ visible. To drive a parsing algorithm in the deterministic construction of the tree associated with the string, Floyd introduced three *precedence relations*, $\lessdot$ (*yields precedence*), $\doteq$ (*equal in precedence*), $\gtrdot$ (*takes precedence*), (algorithmically derived from the grammar) between terminal symbols (Fig. 1 right). They do not satisfy any order axioms and are used to mark, respectively, the beginning, the internal elements, and the end of a grammar right hand side in the substitution rules of a shift-reduce parsing algorithm. For a complete description of Floyd's parsing algorithms driven by these relations, see, e.g, [21].

In this paper, instead, we exploit the more recent characterization of OPL in terms of recognizing automata [25], which are defined on a given alphabet *and* precedence matrix. We define an *OP alphabet* as a pair (Σ, M), where Σ is an alphabet and M, the *operator precedence matrix (OPM)*, is a $|\Sigma\cup\{\#\}|^2$ array describing for each ordered pair of symbols at most one (operator precedence) relation, that is, every entry of M is either $\lessdot$, $\doteq$, $\gtrdot$, or empty (no relation). *We use the symbol # to mark the beginning and the end of a word and always let* $\# \lessdot a$ *and* $a \gtrdot \#$ *for all* $a \in \Sigma$.

Let $w = (a_1...a_n) \in \Sigma^+$ be a non-empty word. We say $a_0 = a_{n+1} = \#$ and define

$E \to E + T \mid T$
$T \to T \times F \mid F$
$F \to n \mid (E)$



	$+$	$\times$	$($	$)$	n
$+$	$\gtrdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\lessdot$
$\times$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\gtrdot$	$\lessdot$
$($	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	$\lessdot$
$)$	$\gtrdot$	$\gtrdot$		$\gtrdot$	
n	$\gtrdot$	$\gtrdot$		$\gtrdot$	

Figure 1 A grammar generating arithmetic expressions (left), an example derivation tree (center), and the precedence matrix (right). E.g. $M[1,2] = \lessdot$ means that $+$ yields precedence to $\times$.

a new *chain* relation $\curvearrowright$ on the set of all positions of $\#w\#$, inductively, as follows. Let $0 \leq i < j \leq n+1$. We write $i \curvearrowright j$ if there exists a sequence of positions $i = k_1 < ... < k_m = j$, $m \geq 3$, such that $a_{k_1} \lessdot a_{k_2} \doteq ... \doteq a_{k_{m-1}} \gtrdot a_{k_m}$ and either $k_s + 1 = k_{s+1}$ or $k_s \curvearrowright k_{s+1}$ for each $s \in \{1, ..., m-1\}$. We say that w is *compatible* with M if for $\#w\#$, we have $0 \curvearrowright n+1$. We denote by (Σ^+, M) the set of all non-empty words over Σ which are compatible with M. For a *complete* OPM M, i.e. one without empty entries, this is Σ^+.

The chain relation can be compared with the *nesting* or *matching relation* of [2], which is also originating from additional information on the alphabet. However, instead of partitioning the alphabet into three disjoint parts (calls, internals, and returns), we add a *binary* relation for every pair of symbols denoting their precedence relation. Therefore, in contrast to the nesting relation, the same symbol can be either call or return depending on its context, and the same position can be part of multiple chain relations.

▶ **Definition 1.** A *(nondeterministic) operator precedence automaton (OPA)* $\mathcal{A}$ over an OP alphabet (Σ, M) is a tuple $\mathcal{A} = (Q, I, F, \delta)$, where $\delta = (\delta_{\text{shift}}, \delta_{\text{push}}, \delta_{\text{pop}})$, consisting of

- a finite set of states Q, the set of initial states $I \subseteq Q$, the set of final states $F \subseteq Q$, and
- the transition relations $\delta_{\text{shift}}, \delta_{\text{push}} \subseteq Q \times \Sigma \times Q$, and $\delta_{\text{pop}} \subseteq Q \times Q \times Q$.

Let $\Gamma = \Sigma \times Q$. A *configuration* of $\mathcal{A}$ is a triple $C = \langle \Pi, q, w\# \rangle$, where $\Pi \in \bot\Gamma^*$ represents a stack, $q \in Q$ the current state, and w the remaining input to read. A *run* of $\mathcal{A}$ on $w = a_1...a_n$ is a finite sequence of configurations $C_0 \vdash ... \vdash C_m$, such that every transition $C_i \vdash C_{i+1}$ has one of the following forms, where a is the first component of the topmost symbol of the stack Π, or $\#$ if the stack is $\bot$, and b is the next symbol of the input to read:

$$\begin{aligned}
\textit{push move}: \quad & \langle \Pi, q, bx \rangle \vdash \langle \Pi[b, q], r, x \rangle && \text{if } a \lessdot b \text{ and } (q, b, r) \in \delta_{\text{push}}, \\
\textit{shift move}: \quad & \langle \Pi[a, p], q, bx \rangle \vdash \langle \Pi[b, p], r, x \rangle && \text{if } a \doteq b \text{ and } (q, b, r) \in \delta_{\text{shift}}, \\
\textit{pop move}: \quad & \langle \Pi[a, p], q, bx \rangle \vdash \langle \Pi, r, bx \rangle && \text{if } a \gtrdot b \text{ and } (q, p, r) \in \delta_{\text{pop}}.
\end{aligned}$$

An *accepting run* of $\mathcal{A}$ on w is a run from $\langle \bot, q_I, w\# \rangle$ to $\langle \bot, q_F, \# \rangle$, where $q_I \in I$ and $q_F \in F$. The *language accepted by* $\mathcal{A}$, denoted $L(\mathcal{A})$, consists of all words over (Σ^+, M) which have an accepting run on $\mathcal{A}$. We say $L \subseteq (\Sigma^+, M)$ is an *OPL* if L is accepted by an OPA over (Σ, M). As proven in [25], the deterministic variant of an OPA, using a single initial state and transition functions instead of relations, is as expressive as nondeterministic OPA.

An example automaton is depicted in Fig. 2: with the OPM of Fig. 1 (right), it accepts the same language as the grammar of Fig. 1 (left).

▶ **Definition 2.** The logic MSO(Σ, M), short MSO, and its semantics is defined as in [25]

$$\beta ::= \text{Lab}_a(x) \mid x \leq y \mid x \curvearrowright y \mid x \in X \mid \neg\beta \mid \beta \vee \beta \mid \exists x.\beta \mid \exists X.\beta$$

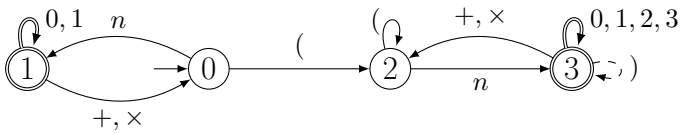


Figure 2 An OPA recognizing the language of the grammar of Fig. 1. The graphical notation is imported from [25]: pushes are normal arrows, shifts are dashed, pops are double arrows.

where $a \in \Sigma \cup \{\#\}$ and x, y, X are first resp. second order variables. The predicate $\mathrm{Lab}_a(x)$ asserts that position x is labeled a. The semantics of $\curvearrowright$ is defined by the chain relation.

▶ **Theorem 3** ([25]). *A language L over (Σ, M) is an OPL iff it is MSO-definable.*

3 Weighted OPL and Their Relation to Weighted VPL

In this section, we introduce a weighted extension of OPA. We show that weighted OPL include weighted visibly pushdown automata (VPL) and give examples showing how these weighted automata can express behaviors which were not expressible before.

Let $\mathbb{K} = (K, +, \cdot, 0, 1)$ be a *semiring*, i.e., $(K, +, 0)$ is a commutative monoid, $(K, \cdot, 1)$ is a monoid, $(x+y)\cdot z = x\cdot z + y\cdot z$, $x\cdot(y+z) = x\cdot y + x\cdot z$, and $0\cdot x = x\cdot 0 = 0$ for all $x, y, z \in K$. $\mathbb{K}$ is called *commutative* if $(K, \cdot, 1)$ is commutative. Important examples of commutative semirings include the Boolean semiring $\mathbb{B} = (\{0,1\}, \vee, \wedge, 0, 1)$, the semiring of the natural numbers $\mathbb{N} = (\mathbb{N}, +, \cdot, 0, 1)$, or the tropical semirings $\mathbb{R}_{\max} = (\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0)$ and $\mathbb{R}_{\min} = (\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$. Significant non-commutative semirings are $n \times n$-matrices over semirings $\mathbb{K}$ with matrix addition and multiplication as usual ($n \geq 2$), or the semiring $(\mathcal{P}(\Sigma^*), \cup, \cdot, \emptyset, \{\varepsilon\})$ of languages over Σ.

▶ **Definition 4.** A *weighted OPA (wOPA)* $\mathcal{A}$ over an OP alphabet (Σ, M) and a semiring $\mathbb{K}$ is a tuple $\mathcal{A} = (Q, I, F, \delta, \mathrm{wt})$, where $\mathrm{wt} = (\mathrm{wt}_{\mathrm{shift}}, \mathrm{wt}_{\mathrm{push}}, \mathrm{wt}_{\mathrm{pop}})$, consisting of

- an OPA $\mathcal{A}' = (Q, I, F, \delta)$ over (Σ, M) and
- the weight functions $\mathrm{wt}_{op} : \delta_{op} \to K$, $op \in \{\mathrm{shift}, \mathrm{push}, \mathrm{pop}\}$.

We call a wOPA *restricted*, denoted by *rwOPA*, if $\mathrm{wt}_{\mathrm{pop}}(q, p, r) = 1$ for each $(q, p, r) \in \delta_{\mathrm{pop}}$.

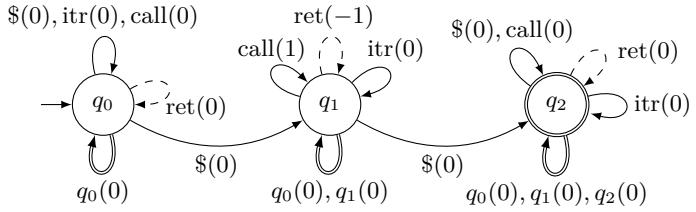
A *configuration* of a wOPA is a tuple $\langle \Pi, q, w\#, k\rangle$, where $(\Pi, q, w\#)$ is a configuration of the OPA $\mathcal{A}'$ and $k \in K$. A *run* of $\mathcal{A}$ is defined as for OPA, where, additionally, the weight k is updated by multiplying with the weight of the encountered transition, as follows.

$$\begin{aligned}
\langle \Pi, q, bx, k\rangle &\vdash \langle \Pi[b, q], r, x, k \cdot \mathrm{wt}_{\mathrm{push}}(q, b, r)\rangle && \text{if } a \lessdot b \text{ and } (q, b, r) \in \delta_{\mathrm{push}},\\
\langle \Pi[a, p], q, bx, k\rangle &\vdash \langle \Pi[b, p], r, x, k \cdot \mathrm{wt}_{\mathrm{shift}}(q, b, r)\rangle && \text{if } a \doteq b \text{ and } (q, b, r) \in \delta_{\mathrm{shift}},\\
\langle \Pi[a, p], q, bx, k\rangle &\vdash \langle \Pi, r, bx, k \cdot \mathrm{wt}_{\mathrm{pop}}(q, p, r)\rangle && \text{if } a \gtrdot b \text{ and } (q, p, r) \in \delta_{\mathrm{pop}}.
\end{aligned}$$

We call a run ρ *accepting* if it goes from $\langle \bot, q_I, w\#, 1\rangle$ to $\langle \bot, q_F, \#, k\rangle$, where $q_I \in I$ and $q_F \in F$. For such an accepting run, the *weight of* ρ is defined as $\mathrm{wt}(\rho) = k$. Finally, the *behavior of* $\mathcal{A}$ is a function $[\![\mathcal{A}]\!] : (\Sigma^+, M) \to K$, defined as

$$[\![\mathcal{A}]\!](w) = \sum_{\rho \text{ acc. run of } \mathcal{A} \text{ on } w} \mathrm{wt}(\rho) \ .$$

Every function $S : (\Sigma^+, M) \to K$ is called an *OP-series* (short: *series*, also *weighted language*). A wOPA $\mathcal{A}$ *accepts* a series S if $[\![\mathcal{A}]\!] = S$. A series S is called *recognizable* or a *wOPL* if there exists an wOPA $\mathcal{A}$ accepting it. S is *strictly recognizable* or an *rwOPL* if there exists an rwOPA $\mathcal{A}$ accepting it.



	call	ret	itr	$
call	⋖	≐	⋗	⋖
ret	⋗	⋗	⋗	⋖
itr	⋗		⋖	⋖
$	⋗	⋗	⋗	

Figure 3 The weighted OPA $\mathcal{A}_{\text{itr}}$ penalizing unmatched calls nondeterministically, and its precedence matrix (right). Weights are given in parentheses at transitions. The weight semiring is $\mathbb{Z}_{\max} = (\mathbb{Z} \cup \{-\infty\}, \max, +, -\infty, 0)$.

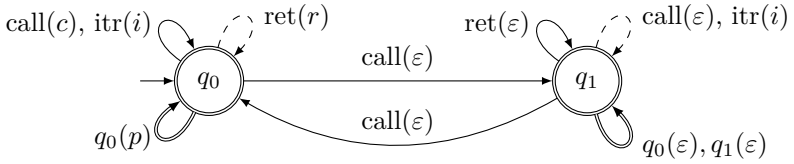


Figure 4 The wOPA $\mathcal{A}_{\log}$ nondeterministically writes logs at different levels of detail.

▶ **Example 5.** Consider a system that manages calls and returns (in VPL terminology) in a traditional LIFO policy but discards all pending calls if an interrupt (*itr*) occurs. Such a system can be naturally modeled by suitable OPA that can formalize various types of policies to manage interrupts [25][1]. We can use weights to, for instance, count the number of interrupted calls. A first simple wOPA could attach a negative weight to calls and a compensating one to corresponding returns so that the final weight assigned to the string would be "neutral" only if no call is discarded.

Consider now a more complex system where the penalties for unmatched calls may change nondeterministically. Here, we assume words to be separated into different intervals by the symbol \$, of which one nondeterministically chosen represents, e.g., a critical operating time, during which unmatched calls are penalized. The wOPA $\mathcal{A}_{\text{itr}}$ given in Fig. 3 formalizes such a system by assigning to an input sequence a global weight that is the maximal number of unmatched calls in one interval.

$\mathcal{A}_{\text{itr}}$ can be easily modified to formalize several variations of its policy: e.g., different policies could be associated with different intervals, different weights could be assigned to different types of calls and/or interrupts, different policies could also be defined by choosing different semirings, etc. Note that $\mathcal{A}_{\text{itr}}$ is restricted. ◀

▶ **Example 6.** The automaton $\mathcal{A}_{\log}$, depicted in Fig. 4, chooses non-deterministically between logging everything and logging only 'important' information, e.g., only interrupts (this could be a system dependent on energy, WiFi, etc.). Notice that in this case assigning nontrivial weights to pop transitions is crucial. Let $\Sigma = \{\text{call}, \text{ret}, \text{itr}\}$, and define M as the obvious projection of $\mathcal{A}_{\text{itr}}$'s OPM. We employ the semiring $(\text{Fin}_{\Sigma'}, \cup, \circ, \emptyset, \{\varepsilon\})$ of all finite languages over $\Sigma' = \{c, r, p, i\}$. Then, $[\![\mathcal{A}_{\log}]\!](w)$ yields all possible logs on w. ◀

The above example can be exploited to show by a pumping-like argument that wOPA are more expressive than rwOPA. This is due to the fact that a number of consecutive pops can attach to one position a product of size only bounded by the word-length and it is impossible to attach these weights at other positions without destroying their sequential order.

[1] A similar motivation inspired the recent extension of VPL as colored nested words by [1].

	Σ_{call}	Σ_{ret}	Σ_{int}
Σ_{call}	$\lessdot$	$\doteq$	$\lessdot$
Σ_{ret}	$\gtrdot$	$\gtrdot$	$\gtrdot$
Σ_{int}	$\gtrdot$	$\gtrdot$	$\gtrdot$

E.g. $w = a\langle car\rangle$, over $\Sigma_{\text{int}} = \{a\}$, $\Sigma_{\text{call}} = \{\langle c\}$, $\Sigma_{\text{ret}} = \{r\rangle\}$

NWA: $q_0 \xrightarrow{a} q_1 \xrightarrow{\langle c} q_2 \xrightarrow{a} q_3 \xrightarrow{r\rangle} q_4$

OPA: $q_0 \xrightarrow{a} q_1' \Rightarrow q_1 \xrightarrow{\langle c} q_2 \xrightarrow{a} q_3' \Rightarrow q_3 \overset{r\rangle}{\dashrightarrow} q_4' \Rightarrow q_4$

Figure 5 The OPM M for VPL and an example of the translation of runs from NWA to OPA.

▶ **Proposition 7.** *There exist an OP alphabet (Σ, M), a semiring $\mathbb{K}$, and a weighted language $S : (\Sigma^+, M) \to K$ such that S is recognizable but not strictly recognizable.*

On the other hand, for commutative semirings rwOPA and wOPA are equally expressive.

▶ **Theorem 8.** *Let $\mathcal{A}$ be a wOPA over an OP alphabet (Σ, M) and a commutative semiring $\mathbb{K}$. Then, there exists an rwOPA $\mathcal{B}$ over (Σ, M) and $\mathbb{K}$ with $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$.*

Proof (Sketch). Let $\mathcal{A} = (Q, I, F, \delta, \text{wt})$ be a wOPA over (Σ, M) and $\mathbb{K}$. We construct an rwOPA $\mathcal{B}$ over (Σ, M) and $\mathbb{K}$ with the state set $Q' = Q \times Q \times Q$ and with the same behavior as $\mathcal{A}$ as follows. In the first state component $\mathcal{B}$ simulates $\mathcal{A}$. In the second and third state component of Q' the automaton $\mathcal{B}$ guesses the states q and r of the pop transition (q, p, r) of $\mathcal{A}$ which corresponds to the next push transition following after this configuration. This enables us to transfer the weight from the pop transition to the correct push transition. ◀

In the following, we show that rwOPL strictly include *weighted visibly pushdown languages* (wVPL). VPL is the class of languages corresponding to nested words [2] and recognized by *visibly pushdown automata* (VPA) or the expressively equivalent *nested word automata* (NWA). Let Σ be a *visibly pushdown alphabet* consisting of calls, internals, and returns. In [8], it was shown that for every VPA over Σ, there exists an equivalent OPA [25] over (Σ, M), where M is the OPM defined in Fig. 5.

In [26, 15], *weighted nested word automata* (wNWA) were introduced. These add semiring weights at every transition again depending on the information which symbols are calls, internals, or returns. Since every nested word has a unique representation over a visibly pushdown alphabet Σ, it can be interpreted as a compatible word of (Σ^+, M). In particular, we can interpret a wVPL, i.e., the language of a wNWA, as an OP-series $(\Sigma^+, M) \to \mathbb{K}$.

▶ **Theorem 9.** *Let $\mathbb{K}$ be a semiring and M be the OPM of Fig. 5. Then for every wNWA $\mathcal{A}$ as defined in [15], there exists an rwOPA $\mathcal{B}$ with $[\![\mathcal{A}]\!](w) = [\![\mathcal{B}]\!](w)$ for all $w \in (\Sigma^+, M)$.*

We give an intuition for this result as follows. Note that pushes, shifts, and pops significantly differ from calls, internals, and returns. Indeed, a return of a NWA reads and 'consumes' a symbol, while a pop of an OPA just pops the stack and leaves the next symbol untouched. Studying the OPM M and the example runs of Fig. 5, we see that every symbol of Σ_{ret} forces a shift transition of an OPA immediately followed by a pop. This suggests a fairly natural construction where we can simulate every weighted call by a weighted push, every weighted internal by a weighted push together with a pop and every weighted return by a weighted shift together with a pop. Hence, we may omit weights at pop transitions.

Since OPA are strictly more expressive than VPA [8], this gives, together with Proposition 7, a complete picture of the expressive power of these three classes of weighted languages:

$$\text{wVPL} \subsetneq \text{rwOPL} \subsetneq \text{wOPL} \ .$$

Note that in the context of formal power series, wVPL strictly contain recognizable power series and wOPL form a proper subset of the class of algebraic power series, i.e., series recognized by weighted pushdown automata [23].

4 A Nivat Theorem

In this section, we show that strictly recognizable series are exactly those series which can be derived from a restricted weighted OPA with only one state, intersected with an unweighted OPL, and using an OPM-preserving projection of the alphabet.

In the following, we study closure properties of wOPL and rwOPL. As usual, we extend the operations $+$ and $\cdot$ to series by means of pointwise definitions.

▶ **Proposition 10.** *Let $S : (\Sigma^+, M) \to K$ be a recognizable (resp. strictly recognizable) series and $L \subseteq (\Sigma^+, M)$ an OPL. Then, the series $(S \cap L)(w) = \left\{ \begin{array}{ll} S(w) & , \text{ if } w \in L \\ 0 & , \text{ otherwise} \end{array} \right\}$ is recognizable (resp. strictly recognizable).*

Furthermore, if $\mathbb{K}$ is commutative, then the product of two recognizable (resp. strictly recognizable) series over (Σ^+, M) is again recognizable (resp. strictly recognizable).

Next, we show that recognizable series are closed under projections which preserve the OPM. For two OP alphabets (Σ, M), (Γ, M'), we write $h : (\Sigma, M) \to (\Gamma, M')$ for a mapping $h : \Sigma \to \Gamma$ such that for all $\bullet \in \{\lessdot, \doteq, \gtrdot\}$, we have $a \bullet b$ if and only if $h(a) \bullet h(b)$. We can extend h to a function $h : (\Sigma^+, M) \to (\Gamma^+, M')$ by setting $h(a_1 a_2 ... a_n) = h(a_1)h(a_2)...h(a_n)$. Let $S : (\Sigma^+, M) \to K$ be a series. We define $h(S) : (\Gamma^+, M') \to K$ for each $v \in (\Gamma^+, M')$ by

$$h(S)(v) = \sum_{w \in (\Sigma^+, M), h(w) = v} S(w) \ . \tag{1}$$

▶ **Proposition 11.** *Let $\mathbb{K}$ be a semiring, $S : (\Sigma^+, M) \to K$ recognizable (resp. strictly recognizable), and $h : (\Sigma, M) \to (\Gamma, M')$. Then, $h(S) : (\Gamma^+, M') \to K$ is recognizable (resp. strictly recognizable).*

Let h be a map between two alphabets. Given $h : \Gamma \to \Sigma$ and an OP alphabet (Σ, M), we define $h^{-1}(M)$ by setting $h^{-1}(M)_{a'b'} = M_{h(a')h(b')}$ for all $a', b' \in \Gamma$. As h is OPM-preserving, for every series $S : (\Sigma^+, M) \to K$, we get a series $h(S) : (\Gamma^+, h^{-1}(M)) \to K$, using the sum over all pre-images as in formula (1).

Let $\mathcal{N}(\Sigma, M, \mathbb{K})$ comprise all series $S : (\Sigma^+, M) \to K$ for which there exist an alphabet Γ, a map $h : \Gamma \to \Sigma$, and a one-state rwOPA $\mathcal{B}$ over $(\Gamma, h^{-1}(M))$ and $\mathbb{K}$ and an OPL L over $(\Gamma, h^{-1}(M))$ such that $S = h([\![\mathcal{B}]\!] \cap L)$.

Then, we can show that every rwOPL can be decomposed into the above introduced fragments. Using this decomposition and the closure properties of Prop. 10 and Prop. 11, we get the following Nivat-Theorem for weighted operator precedence automata.

▶ **Theorem 12.** *Let $\mathbb{K}$ be a semiring and $S : (\Sigma^+, M) \to K$ be a series. Then S is strictly recognizable if and only if $S \in \mathcal{N}(\Sigma, M, \mathbb{K})$.*

5 Weighted MSO-Logic for OPL

We use modified ideas from Droste and Gastin [11], also incorporating the distinction into *boolean* formulas β and *weighted* formulas φ as in [5]. The boolean formulas model classical unweighted features, whereas weighted formulas may deal with quantitative aspects.

▶ **Definition 13.** We define the weighted logic MSO$(\mathbb{K}, (\Sigma, M))$, short MSO$(\mathbb{K})$, as

$$\beta ::= \mathrm{Lab}_a(x) \mid x \leq y \mid x \curvearrowright y \mid x \in X \mid \neg\beta \mid \beta \vee \beta \mid \exists x.\beta \mid \exists X.\beta$$
$$\varphi ::= \beta \mid k \mid \varphi \oplus \varphi \mid \varphi \otimes \varphi \mid \bigoplus_x \varphi \mid \bigoplus_X \varphi \mid \prod_x \varphi$$

where $a \in \Sigma \cup \{\#\}$, $k \in \mathbb{K}$; x, y are first order variables; and X is a second order variable.

$$
\begin{aligned}
[\![\beta]\!]_{\mathcal{V}}(w,\sigma) &= \begin{cases} 1\,, & \text{if } (w,\sigma) \models \beta \\ 0\,, & \text{otherwise} \end{cases} & [\![\bigoplus_x \varphi]\!]_{\mathcal{V}}(w,\sigma) &= \sum_{i\in|w|} [\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(w,\sigma[x\to i]) \\
[\![k]\!]_{\mathcal{V}}(w,\sigma) &= k \quad \text{for all } k\in\mathbb{K} & [\![\bigoplus_X \varphi]\!]_{\mathcal{V}}(w,\sigma) &= \sum_{I\subseteq|w|} [\![\varphi]\!]_{\mathcal{V}\cup\{X\}}(w,\sigma[X\to I]) \\
[\![\varphi\oplus\psi]\!]_{\mathcal{V}}(w,\sigma) &= [\![\varphi]\!]_{\mathcal{V}}(w,\sigma) + [\![\psi]\!]_{\mathcal{V}}(w,\sigma) & [\![\prod_x \varphi]\!]_{\mathcal{V}}(w,\sigma) &= \prod_{i\in|w|} [\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(w,\sigma[x\to i]) \\
[\![\varphi\otimes\psi]\!]_{\mathcal{V}}(w,\sigma) &= [\![\varphi]\!]_{\mathcal{V}}(w,\sigma)\cdot[\![\psi]\!]_{\mathcal{V}}(w,\sigma)
\end{aligned}
$$

Figure 6 Semantics of weighted MSO logic for OPL.

Let $w \in (\Sigma^+, M)$ and $\varphi \in \mathrm{MSO}(\mathbb{K})$. As usual, let $[w] = \{1, ..., |w|\}$ and $\mathcal{V}$ be a finite set of variables containing *free*(φ), all free variables of φ. A $(\mathcal{V}, w)$-*assignment* σ is a function assigning to every first order variable of $\mathcal{V}$ an element of $[w]$ and to every second order variable a subset of $[w]$. We define $\sigma[x \to i]$ (and analogously $\sigma[X \to I]$) as the $(\mathcal{V}\cup\{x\}, w)$-assignment mapping x to i and coinciding with σ on all variables different from x.

By following classical approaches, we consider the extended alphabet $\Sigma_{\mathcal{V}} = A \times \{0,1\}^{\mathcal{V}}$ together with its natural OPM $M_{\mathcal{V}}$ defined such that for all $(a,s),(b,t) \in \Sigma_{\mathcal{V}}$ and all $\bullet \in \{\lessdot, \doteq, \gtrdot\}$, we have $(a,s) \bullet (b,t)$ if and only if $a \bullet b$. We represent the word w together with the assignment σ as a word (w,σ) over $(\Sigma_{\mathcal{V}}, M_{\mathcal{V}})$ such that 1 denotes every position where x resp. X holds. A word over $\Sigma_{\mathcal{V}}$ is called *valid*, if every first order variable is assigned to exactly one position. Being valid is a property which can be checked by an OPA.

We define the *semantics* of $\varphi \in \mathrm{MSO}(\mathbb{K})$ as a function $[\![\varphi]\!]_{\mathcal{V}} : (\Sigma_{\mathcal{V}}^+, M_{\mathcal{V}}) \to K$ inductively for all valid $(w,\sigma) \in (\Sigma_{\mathcal{V}}^+, M_{\mathcal{V}})$ in Fig. 6. For not valid (w,σ), we set $[\![\varphi]\!]_{\mathcal{V}}(w,\sigma) = 0$. We write $[\![\varphi]\!]$ for $[\![\varphi]\!]_{\mathrm{free}(\varphi)}$. We can show that semantics $\varphi_{\mathcal{V}}$ for different $\mathcal{V}$ are consistent with each other as long as $\mathcal{V}$ contains all free variables of φ. If φ contains no free variables, φ is a *sentence* and $[\![\varphi]\!] : (\Sigma^+, M) \to K$.

▶ **Example 14.** Let us go back to the automaton $\mathcal{A}_{\mathrm{itr}}$ depicted in Fig. 3. The following boolean formula β defines three subsets of string positions, X_0, X_1, X_2, representing, respectively, the string portions where unmatched calls are not penalized, namely X_0, X_2, and the portion where they are, namely X_1:

$$
\begin{aligned}
\beta = \quad & x \in X_0 \leftrightarrow \exists y \exists z (y > x \wedge z > x \wedge \mathrm{Lab}_{\$}(y) \wedge \mathrm{Lab}_{\$}(z)) \\
& \wedge\, x \in X_1 \leftrightarrow \exists y \exists z (y \le x \le z \wedge \mathrm{Lab}_{\$}(y) \wedge \mathrm{Lab}_{\$}(z) \wedge (x \ne y \wedge x \ne z \to \neg\, \mathrm{Lab}_{\$}(x))) \\
& \wedge\, x \in X_2 \leftrightarrow \exists y \exists z (y < x \wedge z < x \wedge \mathrm{Lab}_{\$}(y) \wedge \mathrm{Lab}_{\$}(z))\ .
\end{aligned}
$$

Weight assignment is formalized by the formula φ which assigns weight 0 to calls, returns, and interrupts outside portion X_1; and weights $1, -1, 0$ to calls, returns, and interrupts, respectively, within portion X_1:

$$
\begin{aligned}
\varphi = \quad & (\neg((x \in X_0 \vee x \in X_2) \wedge (\mathrm{Lab}_{\mathrm{call}}(x) \vee \mathrm{Lab}_{\mathrm{ret}}(x) \vee \mathrm{Lab}_{\mathrm{itr}}(x))) \oplus 0) \\
& \otimes (\neg(x \in X_1 \wedge \mathrm{Lab}_{\mathrm{call}}(x)) \oplus 1) \otimes (\neg(x \in X_1 \wedge \mathrm{Lab}_{\mathrm{ret}}(x)) \oplus -1) \\
& \otimes (\neg(x \in X_1 \wedge \mathrm{Lab}_{\mathrm{itr}}(x)) \oplus 0) \otimes (\neg\, \mathrm{Lab}_{\$}(x) \oplus 0)\ .
\end{aligned}
$$

Then, the formula $\psi = \prod_x (\beta \otimes \varphi)$ defines the weight assigned by $\mathcal{A}_{\mathrm{itr}}$ to an input string through a single nondeterministic run and finally $\chi = \bigoplus_{X_0} \bigoplus_{X_1} \bigoplus_{X_2} \psi$ defines the global weight of every string in an equivalent way as the one defined by $\mathcal{A}_{\mathrm{itr}}$. ◀

As shown by [11] in the case of words, the full weighted logic is strictly more powerful than weighted automata. A similar example also applies here. Therefore, in the following, we restrict our logic in an appropriate way.

▶ **Definition 15.** The *set of almost boolean formulas* is the smallest set of all formulas of MSO($\mathbb{K}$) containing all $k \in \mathbb{K}$ and all boolean formulas which is closed under $\oplus$ and $\otimes$.

Adapting ideas from [14], we can show by structural induction that almost boolean formulas describe precisely a certain form of wOPA's behaviors, called *OPL step functions*, which are all series S that can be written as $S = \sum_{i=1}^{n} k_i \mathbb{1}_{L_i}$, where L_i are OPL forming a partition of (Σ^+, M) and $k_i \in \mathbb{K}$ for each $i \in \{1, ..., n\}$. Furthermore, OPL step functions are recognizable by rwOPA and are closed under the natural extension of the semiring's $+$ and $\cdot$ to series.

▶ **Definition 16.** We call $\varphi \in$ MSO($\mathbb{K}$) *restricted* if for all subformulas $\psi \otimes \theta$ of φ either ψ is almost boolean or all semiring weights occurring in ψ and θ commute elementwise, and additionally, for all subformulas $\prod_x \psi$ of φ, ψ is almost boolean.

In Example 14, the formula β is boolean, the formula φ is almost boolean, and ψ and χ are restricted. Notice that ψ and χ would be restricted even if $\mathbb{K}$ were not commutative.

▶ **Proposition 17.** *Let φ and ψ be two formulas of* MSO($\mathbb{K}$) *such that $[\![\varphi]\!]$ and $[\![\psi]\!]$ are recognizable (resp. strictly recognizable). Then we have*

- *$[\![\varphi \oplus \psi]\!]$, $[\![\sum_x \varphi]\!]$, and $[\![\sum_X \varphi]\!]$ are recognizable (resp. strictly recognizable).*
- *$[\![\varphi \otimes \psi]\!]$ is (resp. strictly) recognizable if $\varphi \otimes \psi$ is a subformula of a restricted formula.*
- *$[\![\prod_x \varphi]\!]$ is strictly recognizable if φ is an almost boolean formula of* MSO($\mathbb{K}$).

Proof (Sketch). Closure under $\oplus$ is dealt with by an usual disjoint union of two wOPA (resp. rwOPA). Closure under restricted $\otimes$ is dealt with by Proposition 10. For the sum quantification, we utilize Proposition 11. The closure under the restricted product quantification is non-trivial, but can be proved by adapting previous techniques to OPL step functions. ◀

Then, by induction on the structure of a weighted formula and using Proposition 17, we get

▶ **Proposition 18.** *For every restricted* MSO($\mathbb{K}$)*-sentence φ, there exists an rwOPA $\mathcal{A}$ with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$.*

Now, we show that the converse of Proposition 18 holds as well.

▶ **Proposition 19.** *For every rwOPA $\mathcal{A}$, there exists a restricted* MSO($\mathbb{K}$)*-sentence φ with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$. If $\mathbb{K}$ is commutative, then for every wOPA $\mathcal{A}$, there exists a restricted* MSO($\mathbb{K}$)*-sentence φ with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$.*

Proof. The rationale adopted to build formula φ from $\mathcal{A}$ integrates the approach followed in [11, 15] with the one of [25]. On the one hand we need second order variables suitable to "carry" weights; on the other hand, unlike previous non-OP cases which are managed through real-time automata, an OPA can perform several transitions while remaining in the same position. Thus, we introduce the following second order variables: $X^{\text{push}}_{p,a,q}$ represents the set of positions where $\mathcal{A}$ performs a push move from state p, reading symbol a and reaching state q; $X^{\text{shift}}_{p,a,q}$ has the same meaning as $X^{\text{push}}_{p,a,q}$ for a shift operation; and $X^{\text{pop}}_{p,q,r}$ represents the set of positions of the symbol that is on top of the stack when $\mathcal{A}$ performs a pop transition from state p, with q on top of the stack, reaching r.

Let $\mathcal{V}$ consist of all $X^{\text{push}}_{p,a,q}$, $X^{\text{shift}}_{p,a,q}$, and $X^{\text{pop}}_{p,q,r}$ such that $a \in \Sigma$, $p, q, r \in Q$ and $(p, a, q) \in \delta_{\text{push}}$, resp. δ_{shift}, resp. $(p, q, r) \in \delta_{\text{pop}}$. We denote by $\bar{X}^{\text{push}}$, $\bar{X}^{\text{shift}}$, and $\bar{X}^{\text{pop}}$ enumerations over the respective set of second order variables. Using usual abbreviations for MSO-formulas and some adapted shortcuts from [25] and [11], we define the following unweighted formula ψ to characterize all accepted runs of $\mathcal{A}$

$$\psi = \mathit{Part}(\bar{X}^{\text{push}}, \bar{X}^{\text{shift}}) \wedge \mathit{Unique}(\bar{X}^{\text{pop}}) \wedge \mathit{InitFinal} \wedge \mathit{Tr}_{\text{push}} \wedge \mathit{Tr}_{\text{shift}} \wedge \mathit{Tr}_{\text{pop}} \ .$$

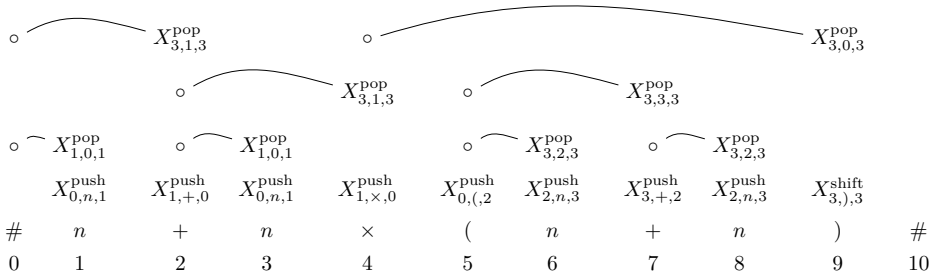


Figure 7 The string of Fig. 1 with the 2nd order variables evidenced for the automaton of Fig. 2.

Here, the subformula *Part* will enforce the push and shift sets to be (together) a partition of all positions, while the *Unique* will make sure that we mark every position with at most one X^{pop}. *InitFinal* controls the initial and the acceptance condition and Tr_{push}, Tr_{shift}, and Tr_{pop} the respective transitions of the run according to their labels as follows.

$$\begin{aligned} Tr_{\text{push}} &= \forall x.\, \textstyle\bigwedge_{p,q\in Q,a\in\Sigma} \left(x \in X^{\text{push}}_{p,a,q} \to \left[\text{Lab}_a(x) \wedge \exists z.(z \lessdot x \wedge (\text{Next}_p(z,x) \vee \text{Succ}_p(z,x)))\right]\right) \\ Tr_{\text{shift}} &= \forall x.\, \textstyle\bigwedge_{p,q\in Q,a\in\Sigma} \left(x \in X^{\text{shift}}_{p,a,q} \to \left[\text{Lab}_a(x) \wedge \exists z.(z \doteq x \wedge (\text{Next}_p(z,x) \vee \text{Succ}_p(z,x)))\right]\right) \\ Tr_{\text{pop}} &= \forall v.\, \textstyle\bigwedge_{p,q\in Q} \left(\left[\bigvee_{r\in Q} v \in X^{\text{pop}}_{p,q,r}\right] \leftrightarrow \left[\exists x \exists y \exists z.(\text{Tree}_{p,q}(x,z,v,y))\right]\right) \ . \end{aligned}$$

The main idea is that for every $x \curvearrowright y$, we encode in $\text{Tree}(x,z,v,y)$ the two other 'critical' positions for this chain, namely z, which is the (either direct or hierarchical) successor of x in this chain and which is the position where we execute the push resulting from $x \lessdot z$; and v, which is the 'chain-predecessor' of y and the position we mark with the respective X^{pop} resulting from $v \gtrdot y$. E.g., with reference to Fig. 1 and Fig. 7, we have $\text{Tree}(4,5,9,10)$.

Furthermore, $\text{Succ}_q(x,y)$ holds for two successive positions where the OPA reaches state q through a push or shift at position y, while $\text{Next}_q(x,y)$ holds when a pop move reaches state q while completing a chain $x \curvearrowright y$. Then $\text{Tree}_{p,q}$ explicitly controls the current state and the state on top of the stack when the pop move is executed as follows.

$$\begin{aligned} \text{Tree}(x,z,v,y) &:= x \curvearrowright y \wedge \begin{pmatrix} (x+1=z \ \vee \ x \curvearrowright z) \wedge \neg\exists t(z<t<y \wedge x \curvearrowright t) \wedge \\ (v+1=y \ \vee \ v \curvearrowright y) \wedge \neg\exists t(x<t<v \wedge t \curvearrowright y) \end{pmatrix} \\ \text{Next}_r(x,y) &:= \exists z \exists v.\, \Big(\text{Tree}(x,z,v,y) \wedge \textstyle\bigvee_{p,q\in Q} v \in X^{\text{pop}}_{p,q,r}\Big) \\ \text{Tree}_{i,j}(x,z,v,y) &:= \text{Tree}(x,z,v,y) \wedge (\text{Succ}_i(v,y) \vee \text{Next}_i(v,y)) \wedge (\text{Succ}_i(x,z) \vee \text{Next}_i(x,z)) \end{aligned}$$

Notice that in the transition formulas, the partition (resp. uniqueness) axioms guarantee that in every run, the left side of the implication (resp. equivalence) is satisfied for only one triple (p,a,q), resp. (p,q,r). Thus, with arguments similar to [25], it can be shown that the sentences satisfying ψ are exactly those accepted by the unweighted OPA subjacent to $\mathcal{A}$.

Now, we add weights to ψ by defining the following restricted weighted formula

$$\begin{aligned} \theta = \psi \otimes \textstyle\prod_x \bigotimes_{p,q\in Q} \Big(& \bigotimes_{a\in\Sigma} (x \in X^{\text{push}}_{p,a,q} \otimes \text{wt}_{\text{push}}(p,a,q)) \oplus (\neg(x \in X^{\text{push}}_{p,a,q}) \otimes 1) \\ & \otimes \bigotimes_{a\in\Sigma} (x \in X^{\text{shift}}_{p,a,q} \otimes \text{wt}_{\text{push}}(p,a,q)) \oplus (\neg(x \in X^{\text{shift}}_{p,a,q}) \otimes 1) \\ & \otimes \bigotimes_{r\in Q} (x \in X^{\text{pop}}_{p,q,r} \otimes \text{wt}_{\text{push}}(p,q,r)) \oplus (\neg(x \in X^{\text{pop}}_{p,q,r}) \otimes 1)\Big) \ . \end{aligned}$$

Here, the second part of θ multiplies up all weights of the encountered transitions. This is the crucial part where we either need that $\mathbb{K}$ is commutative or all pop weights are trivial because the product quantifier of θ assigns the pop weight at a different position than the occurrence of the respective pop transition in the automaton. Using only one product

quantifier (weighted universal quantifier) this is unavoidable, since the number of pops at a given position is only bounded by the word length.

Since the subformulas $x \in X_{()}^{()} \otimes \mathrm{wt}(...)$ of θ are almost boolean, the subformula $\prod_x(...)$ of θ is $\prod$-restricted. Also, ψ is boolean and so θ is $\otimes$-restricted. Thus, θ is a restricted formula. Finally, we define $\varphi = \bigoplus_{X_1} \bigoplus_{X_2} ... \bigoplus_{X_m} \theta$. This implies $[\![\varphi]\!](w) = [\![\mathcal{A}]\!](w)$, for all $w \in (\Sigma^+, M)$. Therefore, φ is our required restricted sentence with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$. ◀

By Proposition 18 and Proposition 19, we obtain the main result of this section.

▶ **Theorem 20.** *Let $\mathbb{K}$ be a semiring and $S : (\Sigma^+, M) \to K$ a series.*

1. *The following are equivalent:*
   - **a.** *$S = [\![\mathcal{A}]\!]$ for some rwOPA.*
   - **b.** *$S = [\![\varphi]\!]$ for some restricted sentence φ of $\mathrm{MSO}(\mathbb{K})$.*
2. *Let $\mathbb{K}$ be commutative. Then, the following are equivalent:*
   - **a.** *$S = [\![\mathcal{A}]\!]$ for some wOPA.*
   - **b.** *$S = [\![\varphi]\!]$ for some restricted sentence φ of $\mathrm{MSO}(\mathbb{K})$.*

Theorem 20 shows that the typical logical characterization of weighted languages does not generalize in the same way to the whole class wOPL: for non-rwOPL we need the extra hypothesis that $\mathbb{K}$ be commutative. Notice, however, that rwOPL may execute unbounded pop sequences; thus, they are powerful enough to include languages that are neither real-time nor visible. This remark naturally raises new intriguing questions which we will briefly address in the conclusion.

6 Conclusion

This paper moves a further step in the path of generalizing a series of results beyond the barrier of regular and structured –or visible– CFL [27, 33, 2, 25]. We introduced and investigated weighted operator precedence automata and a corresponding weighted MSO logic. In our main results we show, for any semiring, that wOPA without pop weights and a restricted weighted MSO logic have the same expressive power. Furthermore, these behaviors can also be described as homomorphic images of the behaviors of particularly simple wOPA reduced to arbitrary unweighted OPA. If the semiring is commutative, these results apply also to wOPA with arbitrary pop weights.

Theorem 20 also raises the problems to find, for arbitrary semirings and for wOPA with pop weights, both an expressively equivalent weighted MSO logic and a Nivat-type result. In [16], very similar problems arose for weighted automata on unranked trees and weighted MSO logic. In [12], the authors showed that with another definition of the behavior of weighted unranked tree automata, an equivalence result for the restricted weighted MSO logic could be derived. Is there another definition of the behavior of wOPA (with pop weights) making them expressively equivalent to our restricted weighted MSO logic?

In [25], OPL of infinite words were investigated and shown to be practically important, so the problem arises to develop a theory of wOPA on infinite words. In order to define their quantitative behaviors, one could try to use valuation monoids as in [14, 9].

Finally, a new investigation field can be opened by exploiting the natural suitability of OPL towards parallel elaboration [3]. Computing weights, in fact, can be seen as a special case of semantic elaboration which can be performed hand-in-hand with parsing. In this case too, we can expect different challenges depending on whether the weight semiring is commutative or not and/or weights are attached to pop transitions too, which would be the natural way to follow the traditional semantic evaluation through synthesized attributes [22].

References

1 Rajeev Alur and Dana Fisman. Colored nested words. In Adrian Horia Dediu, Jan Janousek, Carlos Martín-Vide, and Bianca Truthe, editors, *Language and Automata Theory and Applications, LATA 2016*, volume 9618 of *LNCS*, pages 143–155. Springer, 2016.

2 Rajeev Alur and Parthasarathy Madhusudan. Adding nesting structure to words. *J. ACM*, 56(3):16:1–16:43, 2009.

3 Alessandro Barenghi, Stefano Crespi Reghizzi, Dino Mandrioli, Federica Panella, and Matteo Pradella. Parallel parsing made practical. *Sci. Comput. Program.*, 112(3):195–226, 2015.

4 Jean Berstel and Christophe Reutenauer. *Rational Series and Their Languages*, volume 12 of *EATCS Monographs in Theoretical Computer Science*. Springer, 1988.

5 Benedikt Bollig and Paul Gastin. Weighted versus probabilistic logics. In Volker Diekert and Dirk Nowotka, editors, *Developments in Language Theory, DLT 2009*, volume 5583 of *LNCS*, pages 18–38. Springer, 2009.

6 J. Richard Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik und Grundlagen Math.*, 6:66–92, 1960.

7 Christian Choffrut, Andreas Malcher, Carlo Mereghetti, and Beatrice Palano. First-order logics: some characterizations and closure properties. *Acta Inf.*, 49(4):225–248, 2012.

8 Stefano Crespi Reghizzi and Dino Mandrioli. Operator precedence and the visibly pushdown property. *J. Comput. Syst. Sci.*, 78(6):1837–1867, 2012.

9 Manfred Droste and Stefan Dück. Weighted automata and logics for infinite nested words. *Inf. Comput.*, 253:448–466, 2017.

10 Manfred Droste, Stefan Dück, Dino Mandrioli, and Matteo Pradella. Weighted operator precedence languages. *CoRR*, abs/1702.04597, 2017. URL: `http://arXiv.org/abs/1702.04597`.

11 Manfred Droste and Paul Gastin. Weighted automata and weighted logics. *Theor. Comput. Sci.*, 380(1-2):69–86, 2007. extended abstract in ICALP 2005.

12 Manfred Droste, Doreen Heusel, and Heiko Vogler. Weighted unranked tree automata over tree valuation monoids and their characterization by weighted logics. In Andreas Maletti, editor, *Conference Algebraic Informatics CAI 2015*, volume 9270 of *LNCS*, pages 90–102. Springer, 2015.

13 Manfred Droste, Werner Kuich, and Heiko Vogler, editors. *Handbook of Weighted Automata*. EATCS Monographs in Theoretical Computer Science. Springer, 2009.

14 Manfred Droste and Ingmar Meinecke. Weighted automata and weighted MSO logics for average and long-time behaviors. *Inf. Comput.*, 220:44–59, 2012.

15 Manfred Droste and Bundit Pibaljommee. Weighted nested word automata and logics over strong bimonoids. *Int. J. Found. Comput. Sci.*, 25(5):641–666, 2014.

16 Manfred Droste and Heiko Vogler. Weighted tree automata and weighted logics. *Theor. Comput. Sci.*, 366(3):228–247, 2006.

17 Samuel Eilenberg. *Automata, Languages, and Machines*, volume 59-A of *Pure and Applied Mathematics*. Academic Press, 1974.

18 Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Trans. Am. Math. Soc.*, 98(1):21–52, 1961.

19 E. Allen Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B*, pages 995–1072. MIT Press, 1990.

20 Robert W. Floyd. Syntactic analysis and operator precedence. *J. ACM*, 10(3):316–333, 1963.

21 D. Grune and C. J. Jacobs. *Parsing techniques: a practical guide*. Springer, New York, 2008.

22 Donald E. Knuth. Semantics of context-free languages. *Mathematical Systems Theory*, 2(2):127–145, 1968.

23 Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*, volume 6 of *EATCS Monographs in Theoretical Computer Science*. Springer, 1986.

24 Clemens Lautemann, Thomas Schwentick, and Denis Thérien. Logics for context-free languages. In Leszek Pacholski and Jerzy Tiuryn, editors, *Computer Science Logic, Selected Papers*, volume 933 of *LNCS*, pages 205–216. Springer, 1994.

25 Violetta Lonati, Dino Mandrioli, Federica Panella, and Matteo Pradella. Operator precedence languages: Their automata-theoretic and logic characterization. *SIAM J. Comput.*, 44(4):1026–1088, 2015.

26 Christian Mathissen. Weighted logics for nested words and algebraic formal power series. *Logical Methods in Computer Science*, 6(1), 2010. Selected papers of ICALP 2008.

27 Robert McNaughton. Parenthesis grammars. *J. ACM*, 14(3):490–500, 1967.

28 Robert McNaughton and Seymour Papert. *Counter-free Automata*. MIT Press, Cambridge, USA, 1971.

29 Kurt Mehlhorn. Pebbling mountain ranges and its application of DCFL-recognition. In *Automata, Languages and Programming, ICALP 1980*, volume 85 of *LNCS*, pages 422–435, 1980.

30 Maurice Nivat. Transductions des langages de Chomsky. *Ann. de l'Inst. Fourier*, 18:339–455, 1968.

31 Arto Salomaa and Matti Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Texts and Monographs in Computer Science. Springer, 1978.

32 Marcel Paul Schützenberger. On the definition of a family of automata. *Inf. Control*, 4(2-3):245–270, 1961.

33 James Thatcher. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journ. of Comp. and Syst.Sc.*, 1:317–322, 1967.

34 Boris A. Trakhtenbrot. Finite automata and logic of monadic predicates (in Russian). *Doklady Akademii Nauk SSR*, 140:326–329, 1961.

35 Burchard von Braunmühl and Rutger Verbeek. Input-driven languages are recognized in log n space. In *Proceedings of the Symposium on Fundamentals of Computation Theory*, volume 158 of *LNCS*, pages 40–51. Springer, 1983.