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 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.

Keywords: quantitative automata, operator precedence languages, VPL, quantitative logic

1 Introduction

In the long history of formal languages the family of regular languages (RL), those that are recognized by finite state machines (FSM) or are generated by regular grammars, has always played a major role: thanks to its simplicity and naturalness it enjoys properties that are only partially extended to larger families. Among the many positive results that have been achieved for RL (e.g., expressiveness, decidability, minimization, ...), those of main interest in this paper are the following:

- RLs have been characterized in terms of various mathematical logics. The pioneering papers are due to Büchi, Elgot, and Trakhtenbrot [7,22,37] who independently developed a monadic second order (MSO) logic defining exactly

⋆ supported by Deutsche Forschungsgemeinschaft (DFG) Graduiertenkolleg 1763 (QuantLA).

the RL family. This work too has been followed by many further results; in particular those that exploited weaker but simpler logics such as first-order, propositional, and temporal ones which culminated in the breakthrough of model checking to support automatic verification [31,23,8].

- Weighted RLs have been introduced by Schützenberger in his pioneering paper [35]: by assigning a weight in a suitable algebra to each language word, we may specify several attributes of the word, e.g., relevance, probability, etc. Much research then followed and extended the original Schützenberger's work in various directions, cf. the books [4,21,26,34,14].

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 properties and, consequently, of possible tools supporting their automatic analysis. For instance, for CFL, the containment problem is undecidable and they are not closed under complement.

What was not possible for general CFL, however, has been possible for important subclasses of this family, which together we call *structured CFL*. Informally, with this term we denote those CFLs where the syntactic tree-structure of their words is immediately "visible" in the words themselves. A first historical example of such families is that of parenthesis languages, introduced by McNaughton in another seminal paper [30], which are generated by grammars whose right hand sides are enclosed within pairs of parentheses; not surprisingly an equivalent formalism of parenthesis grammars was soon defined, namely tree-automata which generalize the basics of FSM to tree-like structures instead of linear strings [36]. Among the many variations and generalizations of parenthesis languages the recent family of *input-driven languages (IDL)* [32,6], alias *visibly pushdown languages (VPL)* [2], have received much attention in recent literature. For most of these structured CFL, including in particular IDL, all of the algebraic properties of RL still hold [2]. One of the most noticeable results of this research field has been a characterization of IDL/VPL in terms of a MSO logic that is a fairly natural extension of the original Büchi's one for RL [27,2].

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 [12], associated with *weighted tree automata* [19] and *weighted VPAs* the automata recognizing VPLs, also called weighted NWAs [29,11].

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 [24] 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. After a few further studies [10], OPL's theoretical investigation has

been abandoned due to the advent of LR grammars which, unlike OPL grammars, generate all deterministic CFL.

OPL, however, enjoy a distinguishing property 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 to pop their stack– 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 arithmetic 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 [28].

This 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 [9] and to characterize them, besides the original Floyd's 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 [28]. 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 put together the two above research fields, 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 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 [33] 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.

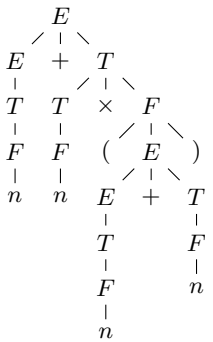
- A weighted MSO logic and, for arbitrary semirings, a Büchi-Elgot-Trakhtenbrot-Theorem proving its expressive equivalence to weighted OPA without weights at pop transitions. As a corollary, for commutative semirings this weighted logic is equivalent to weighted OPA including weights at pop transitions.

2 Preliminaries

We start with an example to provide an intuition of the idea by which R. Floyd made the hidden precedences between symbols occurring in a grammar explicit in parse trees [24]: consider arithmetic expressions with two operators, an additive one and a multiplicative one that takes precedence over the other one, in the sense that, during the interpretation of the expression, multiplications must be executed before sums. Parentheses are used to force different precedence hierarchies. Figure 1 (left) presents a grammar and (center) the derivation tree of the expression $n + n \times (n + n)$; all nonterminals are axioms.

Notice that the structure of the syntax tree (uniquely) corresponding to the input expression reflects the precedence order which drives computing the value attributed to the expression. This structure, however, is not immediately visible in the expression; if we used a parenthesis grammar, it would produce the string $(n + (n \times (n + n)))$ instead of the previous one, and the structure of the corresponding tree would be immediately visible. For this reason we say that such grammars "hide" the structure associated with a sentence, whereas parenthesis grammars and other input-driven ones make the structure explicit in the sentences they generate.

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



	+	×	(	)	n
+	⋗	⋖	⋖	⋗	⋖
×	⋗	⋗	⋖	⋗	⋖
(	⋖	⋖	⋖	≐	⋖
)	⋗	⋗		⋗	
n	⋗	⋗		⋗	

Fig. 1. A grammar generating arithmetic expressions (left), an example derivation tree (center), and the precedence matrix (right).

To model this hierarchical structure and make it accessible, we introduce the *chain relation* $\curvearrowright$. This new relation can be compared with the *nesting* or *matching relation* of [2], as it also is a non-crossing relation, going always forward and originating from additional information on the alphabet. However, it also features significant differences: Instead of adding unary information to symbols, which partition 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. Furthermore, the same position can be part of multiple chain relations.

More precisely, 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 $\lessdot$ (*yields precedence*), $\doteq$ (*equal in precedence*, $\gtrdot$ (*takes precedence*), or empty (no relation).

We use the symbol # to mark the beginning and the end of a word and let always be $\# \lessdot a$ and $a \gtrdot \#$ for all $a \in \Sigma$. As an example, Figure 1 (right) depicts the OPM of the grammar reported on its left, omitting the standard relations for #.

Let $w = (a_1 ... a_n) \in \Sigma^+$ be a word. We say $a_0 = a_{n+1} = \#$ and define a new relation $\curvearrowright$ on the set of all positions of $\#w\#$, inductively, as follows. Let $i, j \in \{0, 1, ..., n+1\}$, $i < j$. Then, we write $i \curvearrowright j$ if there exists a sequence of positions $k_1 ... k_m$ such that $i = k_1 < ... < k_m = j$, $a_{k_1} \lessdot a_{k_2} \doteq ... \doteq a_{k_{m-1}} \gtrdot a_{k_m}$, and either $k_s + 1 = k_{s+1}$ or $k_s \curvearrowright k_{s+1}$ for each $s \in \{1, ..., m-1\}$. In particular, $i \curvearrowright j$ holds if $a_i \lessdot a_{i+1} \doteq ... \doteq a_{j-1} \gtrdot a_j$.

We say w is *compatible* with M if for $\#w\#$ we have $0 \curvearrowright n+1$. In particular, this forces $M_{a_i a_j} \neq \emptyset$ for all $i + 1 = j$ and for all $i \curvearrowright j$. We denote by $(\Sigma, M)^+$ 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^+.

We recall the definition of an operator precedence automaton from [28].

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

- Q, a finite set of states,
- $I \subseteq Q$, the set of initial states,
- $F \subseteq Q$, a set of final states, 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 topmost alphabet symbol of Π and b is the next symbol of the input to read:

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

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 $(\Sigma, M)^+$ which have an accepting run on $\mathcal{A}$. We say that $L \subseteq (\Sigma, M)^+$ is an *OPL* if L is accepted by an OPA over (Σ, M). As proven by [28], the deterministic variant of an OPA, using a single initial state instead of I and transition functions instead of relations, is equally expressive to nondeterministic OPA.

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

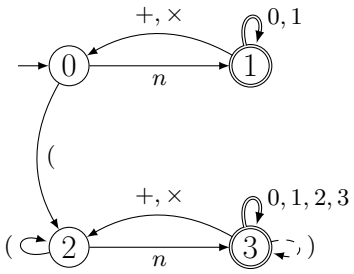


Fig. 2. Automaton for the language of the grammar of Figure 1. Shift, push and pop transitions are denoted by dashed, normal and double arrows, respectively.

Definition 2. The logic MSO(Σ, M), short MSO, is defined as

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

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

We define the natural semantics for this (unweighted) logic as in [28]. The relation $\curvearrowright$ refers to the chain relation introduced above.

Theorem 3 ([28]). *A language L over (Σ, M) is an OPL iff it is MSO-definable.*

3 Weighted OPA and Their Connection to Weighted VPA

In this section, we introduce a weighted extension of operator precedence automata. We show that weighted OPL include weighted 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 cover 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 $\mathbb{R}_{\min} = (\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$. Non-commutative semirings are given by $n \times n$-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 $\mathrm{wt}_{\mathrm{pop}} \equiv 1$, i.e. $\mathrm{wt}_{\mathrm{pop}}(q, p, r) = 1$ for each $(q, p, r) \in \delta_{\mathrm{pop}}$.

A *configuration* of a wOPA is a tuple $C = \langle \Pi, q, w\#, k \rangle$, where $(\Pi, q, w\#)$ is a configuration of the OPA $\mathcal{A}'$ and $k \in \mathbb{K}$. A *run* of $\mathcal{A}$ is a again a sequence of configurations $C_0 \vdash C_1 \ldots \vdash C_m$ satisfying the previous conditions and, additionally, the weight of a configuration is updated by multiplying with the weight of the encountered transition, as follows. As before, we denote with a the topmost symbol of Π and with b the next symbol of the input to read:

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

We call a run ρ *accepting* if it goes from $\langle \bot, q_I, 1, w\# \rangle$ 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 $\mathrm{wt}(\rho) = k$. We denote by $\mathrm{acc}(\mathcal{A}, w)$ the set of all accepting runs of $\mathcal{A}$ on w.

Finally, the *behavior of* $\mathcal{A}$ is a function $[\![\mathcal{A}]\!] : (\Sigma, M)^+ \to K$, defined as

$$[\![A]\!](w) = \sum_{\rho \in \mathrm{acc}(\mathcal{A}, w)} \mathrm{wt}(\rho) \ .$$

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

Example 5. Let us resume, in a simplified version, an example presented in [28] (Example 8) which exploits the ability of OPA to pop many items from the stack without advancing the input head: in this way we can model a system that manages calls and returns in a traditional LIFO policy but discards all pending calls if an interrupt occurs[4]. The weighted automaton of Figure 3 attaches weights to

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

the OPA's transitions in such a way that the final weight of a string is 1 only if no pending call is discarded by any interrupt; otherwise, the more calls are discarded the lower the "quality" of the input as measured by its weight.

More precisely, we define $\Sigma = \{\text{call}, \text{ret}, \text{int}\}$ and the precedence matrix M as a subset of the matrix of Example 8 of [28], i.e., call $\lessdot$ call, call $\doteq$ ret, call $\gtrdot$ int, int $\lessdot$ int, int $\gtrdot$ call, and ret $\gtrdot a$ for all $a \in \Sigma$.

By adopting the same graphical notation as in [28] pushes are normal arrows, shifts are dashed, pops are double arrows; weights are given in brackets at transitions. Let $\#\text{pcall}(w)$ be the number of pending calls of w, i.e.,

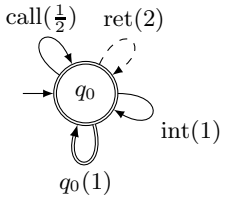


Fig. 3. The weighted OPA $\mathcal{A}_{\text{penalty}}$ penalizing unmatched calls

calls which are never answered by a return. Then the behavior of the automaton $\mathcal{A}_{\text{penalty}}$ over (Σ, M) and the semiring $(\mathbb{N}, +, \cdot, 0, 1)$ given in Figure 3 is $[\![\mathcal{A}_{\text{penalty}}]\!](w) = (\frac{1}{2})^{\#\text{pcall}(w)}$.

The example can be easily enriched by following the same path outlined in [28]: we could add symbols specifying the serving of an interrupt, add different types of calls and interrupts with different priorities and more sophisticated policies (e.g., lower level interrupts disable new calls but do not discard them, whereas higher level interrupts reset the whole system, etc.)

Example 6. The wOPA of Figure 3 is "rooted" in a deterministic OPA; thus the semiring of weights is exploited in a fairly trivial way since only the $\cdot$ operation is used. The automaton $\mathcal{A}_{\text{policy}}$ given in Figure 4, instead, formalizes a more complex system where the penalties for unmatched calls may change nondeterministically within intervals delimited by the special symbol \$. Precisely, the symbols \$ mark intervals during which sequences of calls, returns, and interrupts occur; "normally" unmatched calls are not penalized, but there is a special, nondeterministically chosen interval during which they are penalized; the global weight assigned to an input sequence is the maximum over all nondeterministic runs that are possible when recognizing the sequence.

Here, the alphabet is $\Sigma = \{\text{call}, \text{ret}, \text{int}, \$\}$, and the OPM M, with $a \lessdot \$$ and $\$ \gtrdot a$, for all $a \in \Sigma$ is a natural extension of the OPM of Example 5. As semiring, we take $\mathbb{R}_{\max} = (\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0)$. Then, $[\![\mathcal{A}_{\text{policy}}]\!](w)$ equals the maximal number of pending calls between two consecutive \$. Again, $\mathcal{A}_{\text{policy}}$ can be easily modified/enriched to formalize several variations of its policy: e.g.,

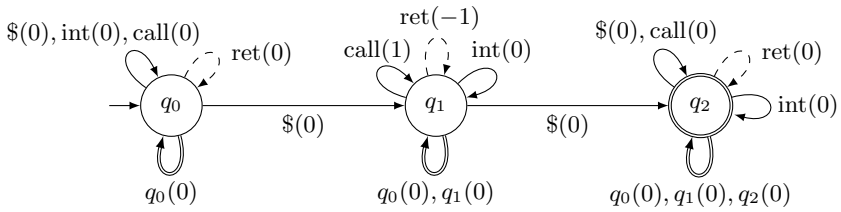


Fig. 4. The weighted OPA $\mathcal{A}_{\text{policy}}$ penalizing unmatched calls nondeterministically

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 both automata, $\mathcal{A}_{\text{penalty}}$ and $\mathcal{A}_{\text{policy}}$, do not use the weight assignment for pops.

Example 7. The next automaton $\mathcal{A}_{\text{log}}$, depicted in Figure 5 chooses non-deterministically between logging everything and logging only 'important' information, e.g., only interrupts (this could be a system dependent on energy, WiFi, ...). Notice that, unlike the previous examples, in this case assigning nontrivial weights to pop transitions is crucial.

Let $\Sigma = \{\text{call}, \text{ret}, \text{int}\}$, and define M as for $\mathcal{A}_{\text{penalty}}$. We employ the semiring $(\text{Fin}_{\Sigma'}, \cup, \circ, \emptyset, \{\varepsilon\})$ of all finite languages over $\Sigma' = \{c, r, p, i\}$. Then, $[\![\mathcal{A}_{\text{log}}]\!](w)$ yields all possible logs on w.

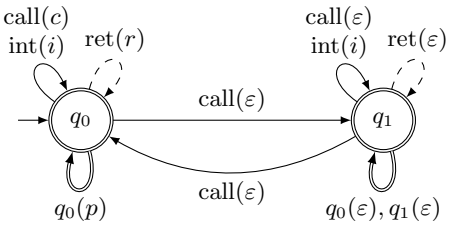


Fig. 5. The wOPA $\mathcal{A}_{\text{log}}$ nondeterministically writes logs at different levels of detail.

As hinted at by our last example, the following proposition shows that in general, wOPA are more expressive than rwOPA.

Proposition 8. *There exists an OP alphabet (Σ, M) and a semiring $\mathbb{K}$ such that there exists a weighted language S which is regular but not strictly regular.*

Proof. Let $\Sigma = \{c, r\}$, $c \lessdot c$, and $c \doteq r$. Consider the semiring $\mathrm{Fin}_{\{a,b\}}$ of all finite languages over $\{a, b\}$ together with union and concatenation. Let $n \in \mathbb{N}$ and $S : (\Sigma, M)^+ \to \mathrm{Fin}_{\{a,b\}}$ be the following series

$$S(w) = \begin{cases} \{a^n b a^n\} & , \text{ if } w = c^n r \\ \emptyset & , \text{ otherwise} \end{cases} .$$

Then, we can define a wOPA which only reads $c^n r$, assigns the weight $\{a\}$ to every push and pop, and the weight $\{b\}$ to the one shift, and therefore accepts S, as in Figure 6.

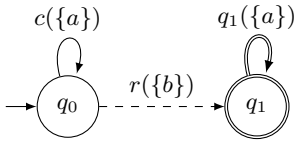


Fig. 6. The wOPA recognizing $S(c^n r) = \{a^n b a^n\}$ and $S(w) = 0$, otherwise.

Now, we show with a pumping argument that there exists no rwOPA which recognizes S. Assume there is an rwOPA $\mathcal{A}$ with $[\![\mathcal{A}]\!] = S$. Note that for all $n \in \mathbb{N}$, the structure of $c^n r$ is fixed as $c \lessdot c \lessdot ... \lessdot c \doteq r$. Let ρ be an accepting run of $\mathcal{A}$ on $c^n r$ with $\mathrm{wt}(\rho) = \{a^n b a^n\}$. Then, the transitions of ρ consist of n pushes, followed by a shift, followed by n pops and can be written as

$$q_0 \xrightarrow{c} q_1 \xrightarrow{c} ... \xrightarrow{c} q_{n-1} \xrightarrow{c} q_n \xrightarrow{r} q_{n+1} \overset{q_{n-1}}{\Longrightarrow} q_{n+2} \overset{q_{n-2}}{\Longrightarrow} ... \overset{q_1}{\Longrightarrow} q_{2n} \overset{q_0}{\Longrightarrow} q_{2n+1} .$$

Both the number of states and the amount of pairs of states are bound. If n is sufficiently large, there exists two pop transitions $\mathrm{pop}(q, p, r)$ and $\mathrm{pop}(q', p', r')$ in this sequence such that $q = q'$ and $p = p'$. This means that we have a loop in the pop transitions going from state q to $q' = q$. Furthermore, the corresponding push to the first transition of this loop was invoked when the automaton was in state p', while the corresponding push to the last pop was invoked in state p. Since $p = p'$, we also have a loop at the corresponding pushes. Then, the run where we skip both loops in the pops and in the pushes is an accepting run for $c^{n-k} r$, for some $k \in \mathbb{N} \setminus \{0\}$.

Since the weight of all pops is trivial, the weight of the pop-loop is ε. If the weight of the push-loop is also ε, then we have an accepting run for $c^{n-k} r$ of weight $\{a^n b a^n\}$, a contradiction. If the weight of the push-loop is not trivial, then by a simple case distinction it has to be either $\{a^i\}$ for some $i \in \mathbb{N} \setminus \{0\}$ or it has to contain the b. In the first case, the run without both loops has weight $\{a^{n-i} b a^n\}$ or $\{a^n b a^{n-i}\}$, in the second case it has weight $\{a^j\}$, for some $j \in \mathbb{N}$. All these runs are not of the form $a^{n-k} b a^{n-k}$, a contradiction. □

We notice that using the same arguments, we can show that also no weighted nested word automata as defined in [29,18] can recognize this series. Even stronger,

we can prove that restricted weighted OPLs are a generalization of weighted VPLs in the following sense. We shortly recall the important definitions. Let $\Sigma = \Sigma_{\text{call}} \sqcup \Sigma_{\text{int}} \sqcup \Sigma_{\text{ret}}$ be a *visibly pushdown alphabet*. A *VPA* is a pushdown automata which uses a push and pop transitions whenever it reads a call or return symbol, respectively.

In [9], it was shown that using the complete OPM of Fig. 7, for every VPA, there exists an equivalent operator precedence grammar which in turn can be transformed into an equivalent OPA.

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

Fig. 7. OPM for VPL

In [29] and [18] weighted extensions of VPA were introduced (in the form of *weighted nested word automata wNWA*). These add semiring weights at every transition again depending on the information what symbols are calls, internals, or returns. Note that every nested word has a representation as a word over a visibly pushdown alphabet Σ and therefore can be seen as a compatible word of $(\Sigma, M)^+$, where M is the OPM of Fig. 7, i.e., we can interpret the behavior of a wNWA as an OP-series $(\Sigma, M)^+ \to \mathbb{K}$.

Theorem 9. *Let $\mathbb{K}$ be a semiring, Σ be a visibly pushdown alphabet, and M be the OPM of Fig. 7. Then for every wNWA $\mathcal{A}$ defined as in [18], 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 although sharing some similarities, pushes, shifts, and pops are not the same thing as calls, internals, and returns. Indeed, a return of a (w)NWA reads and 'consumes' a symbol, while a pop of an (rw)OPA just pops the stack and leaves the next symbol untouched.

After studying Figure 7, this leads to the important observation that every symbol of Σ_{ret} and therefore every return transition of an NWA is simulated not by a pop, but by a shift transition of an OPA (in the unweighted and weighted case).

We give a short demonstrating example: Let $\Sigma_{\text{int}} = \{a\}$, $\Sigma_{\text{call}} = \{\langle c\}$, $\Sigma_{\text{ret}} = \{r\rangle\}$, $w = a\langle car\rangle$. Then every run of an NWA for this word looks like

$$q_0 \xrightarrow{\quad a \quad} q_1 \xrightarrow{\langle c} q_2 \xrightarrow{\quad a \quad} q_3 \xrightarrow{\quad r\rangle \quad} q_4 \ .$$

Every run of an OPA (using the OPM of Fig. 7) looks as follows:

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

where the return was substituted (by the OPM, not by a choice of ours) by a shift followed by a pop.

It follows that we can simulate a weighted call by a weighted push, a weighted internal by a weighted push together with a pop and a weighted return by a weighted shift together with a pop. Therefore, we may indeed omit weights at pop transitions.

Proof (of Theorem 9). Given a weighted NWA $\mathcal{A} = (Q, I, F, (\delta_{\text{call}}, \delta_{\text{int}}, \delta_{\text{ret}}), (\text{wt}_{\text{call}}, \text{wt}_{\text{int}}, \text{wt}_{\text{ret}}))$ over Σ and $\mathbb{K}$, we construct an rwOPA $\mathcal{B} = (Q', I', F', (\delta_{\text{push}}, \delta_{\text{shift}}, \delta_{\text{pop}}), (\text{wt}'_{\text{push}}, \text{wt}'_{\text{shift}}, \text{wt}'_{\text{pop}}))$ over (Σ, M) and $\mathbb{K}$. We set $Q' = Q \cup (Q \times Q)$, $I' = I$, and $F' = F$. We define the relations δ_{push}, δ_{shift}, δ_{pop}, and the functions wt'_{push}, $\text{wt}'_{\text{shift}}$, and wt'_{pop} as follows.

We let δ_{push} contain all triples (q, a, r) with $(q, a, r) \in \delta_{\text{call}}$, and all triples $(q, a, (q, r))$ with $(q, a, r) \in \delta_{\text{int}}$. We set $\text{wt}'_{\text{push}}(q, a, r) = \text{wt}_{\text{call}}(q, a, r)$ and $\text{wt}'_{\text{push}}(q, a, (q, r)) = \text{wt}_{\text{int}}(q, a, r)$. Moreover, we let δ_{shift} contain all triples $(q, a, (p, r))$ with $(q, p, a, r) \in \delta_{\text{ret}}$ and set $\text{wt}'_{\text{shift}}(q, a, (p, r)) = \text{wt}_{\text{ret}}(q, p, a, r)$. Furthermore, we let δ_{pop} contain all triples $((q, r), q, r)$ with $(q, a, r) \in \delta_{\text{int}}$, and all triples $((p, r), p, r)$ with $(q, p, a, r) \in \delta_{\text{ret}}$, and set $\text{wt}'_{\text{pop}}((q, r), q, r) = \text{wt}'_{\text{pop}}((p, r), p, r) = 1$.

Then, a run analysis of $\mathcal{A}$ and $\mathcal{B}$ shows that $[\![\mathcal{B}]\!] = [\![\mathcal{A}]\!]$. □

Together with the result that OPA are strictly more expressive than VPAs [9], this gives a complete picture of the expressive power of these three classes of weighted languages:

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

The following result shows that for commutative semirings the second part of this hierarchy collapses, i.e. restricted rwOPA are equally expressive as wOPA (and therefore can be seen as a kind of normal form in this case).

Theorem 10. *Let $\mathbb{K}$ be a commutative semiring and (Σ, M) an OP alphabet. Let $\mathcal{A}$ be a wOPA. Then, there exists an rwOPA $\mathcal{B}$ with $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$.*

Proof. Let $\mathcal{A} = (Q, I, F, \delta, \text{wt})$ be a wOPA over (Σ, M) and $\mathbb{K}$. Note that for every pop transition of a wOPA, there exists exactly one push transition. We construct an rwOPA $\mathcal{B}$ over the state set $Q' = Q \times Q \times Q$ and with the same behavior as $\mathcal{A}$ with the following idea in mind. In the first state component $\mathcal{B}$ simulates $\mathcal{A}$. In the second and third state component of Q' the automaton $\mathcal{B}$ preemptively 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.

The detailed construction of $\mathcal{B} = (Q', I', F', \delta', \text{wt}')$ over (Σ, M) and $\mathbb{K}$ is the following. If $Q = \emptyset$, then $[\![\mathcal{A}]\!] \equiv 0$ is trivially strictly regular. If Q is nonempty, let $q \in Q$ be a fixed state. Then, we set $Q' = Q \times Q \times Q$, $I' = \{(q_1, q_2, q_3) \mid q_1 \in$

$I, q_2, q_3 \in Q\}$, $F' = \{(q_1, q, q) \mid q_1 \in F\}$, and

$$\begin{aligned}
\delta'_{\text{push}} &= \{((q_1, q_2, q_3), a, (r_1, r_2, r_3)) \mid (q_1, a, r_1) \in \delta_{\text{push}} \text{ and } (q_2, q_1, q_3) \in \delta_{\text{pop}}\} \\
\delta'_{\text{shift}} &= \{((q_1, q_2, q_3), a, (r_1, q_2, q_3)) \mid (q_1, a, r_1) \in \delta_{\text{shift}}\} \\
\delta'_{\text{pop}} &= \{((q_1, q_2, q_3), (p_1, q_1, r_1), (r_1, q_2, q_3)) \mid (q_1, p_1, r_1) \in \delta_{\text{pop}}\} \ .
\end{aligned}$$

Here, every push of $\mathcal{B}$ controls that the previously guessed q_2 and q_3 can be used by a pop transition of $\mathcal{A}$ going from q_2 to q_3 with q_1 on top of the stack. Every pop controls that the symbols on top of the stack are exactly the ones used at this pop. Since the second and third state component are guessed for the next push, they are passed on whenever we read a shift or pop. The second and third component pushed at the first position of a word are guessed by an initial state. At the last push, which therefore has no following push and will propagate the second and third component to the end of the run, the automaton $\mathcal{B}$ has to guess the distinguished state used in the final states.

Therefore, $\mathcal{B}$ has exactly one accepting run (of the same length) for every accepting run of $\mathcal{A}$, and vice versa. Finally, we define the transition weights as follows.

$$\begin{aligned}
\text{wt}'_{\text{push}}((q_1, q_2, q_3), a, (r_1, r_2, r_3)) &= \text{wt}_{\text{push}}(q_1, a, r_1) \cdot \text{wt}_{\text{pop}}(q_2, q_1, q_3) \\
\text{wt}'_{\text{shift}}((q_1, q_2, q_3), a, (r_1, r_2, r_3)) &= \text{wt}_{\text{shift}}(q_1, a, r_1) \\
\text{wt}'_{\text{pop}} &\equiv 1 \ .
\end{aligned}$$

Then, the runs of $\mathcal{A}$ simulated by $\mathcal{B}$ have exactly the same weights but in a different ordering. Since $\mathbb{K}$ is commutative, it follows that $[\![\mathcal{A}]\!] = [\![\mathcal{B}]\!]$. □

In the following, we study closure properties of weighted OPA and restricted weighted OPA. As usual, we extend the operation $+$ and $\cdot$ to series $S, T : (\Sigma, M)^+ \to K$ by means of pointwise definitions as follows:

$$\begin{aligned}
(S + T)(w) &= S(w) + T(w) \text{ for each } w \in (\Sigma, M)^+ \\
(S \odot T)(w) &= S(w) \cdot T(w) \text{ for each } w \in (\Sigma, M)^+ \ .
\end{aligned}$$

Proposition 11. *The sum of two regular (resp. strictly regular) series over $(\Sigma, M)^+$ is again regular (resp. strictly regular).*

Proof. We use a standard disjoint union of two (r)wOPA accepting the given series to obtain a (r)wOPA for the sum as follows.

Let $\mathcal{A} = (Q, I, F, \delta, \text{wt})$ and $\mathcal{B} = (Q', I', F', \delta', \text{wt}')$ be two wOPA over (Σ, M) and $\mathbb{K}$. We construct a wOPA $\mathcal{C} = (Q'', I'', F'', \delta'', \text{wt}'')$ over (Σ, M) and $\mathbb{K}$ by defining $Q'' = Q \sqcup Q'$, $I'' = I \cup I$, $F'' = F \cup F'$, $\delta'' = \delta \cup \delta'$. The weight function is defined by

$$\text{wt}''(t) = \begin{cases} \text{wt}(t) \ , & \text{if } t \in \delta \\ \text{wt}'(t) \ , & \text{if } t \in \delta' \end{cases} \ .$$

Then, $[\![\mathcal{C}]\!] = [\![\mathcal{A}]\!] + [\![\mathcal{B}]\!]$. Furthermore, if $\mathcal{A}$ and $\mathcal{B}$ are restricted, i.e. $\text{wt} \equiv 1$ and $\text{wt}' \equiv 1$, it follow that $\text{wt}'' \equiv 1$, and therefore $\mathcal{C}$ is also restricted. □

Proposition 12. *Let $S : (\Sigma, M)^+ \to K$ be a regular (resp. strictly regular) series and $L \subseteq (\Sigma, M)^+$ an OPL. Then, the series $(S \cap L)(w) = \left\{ \begin{matrix} S(w) \ , & \text{if } w \in L \\ 0 \ , & \text{otherwise} \end{matrix} \right\}$ is regular (resp. strictly regular). Furthermore, if $\mathbb{K}$ is commutative, then the product of two regular (resp. strictly regular) series over $(\Sigma, M)^+$ is again regular (resp. strictly regular).*

Proof. We use a product construction of automata.

Let $\mathcal{A} = (Q, I, F, \delta, \mathrm{wt})$ be a wOPA over (Σ, M) and $\mathbb{K}$ with $[\![\mathcal{A}]\!] = S$ and let $\mathcal{B} = (Q', q_0', F', \delta')$ be a deterministic OPA over (Σ, M) with $L(\mathcal{B}) = L$. We construct a wOPA $\mathcal{C} = (Q'', I'', F'', \delta'', \mathrm{wt}'')$ over (Σ, M) and $\mathbb{K}$, with $[\![\mathcal{C}]\!] = (S \cap L)(w) = \left\{ \begin{matrix} S(w) \ , & \text{if } w \in L \\ 0 \ , & \text{otherwise} \end{matrix} \right\}$, as follows. We define $Q'' = Q \times Q'$, $I'' = I \times \{q_0'\}$, $F'' = F \times F'$, and

$$\begin{aligned}
\delta''_{\mathrm{push}} &= \{((q,q'),a,(r,r')) \mid (q,a,r) \in \delta_{\mathrm{push}} \text{ and } \delta'_{\mathrm{push}}(q',a) = r'\} \ , \\
\delta''_{\mathrm{shift}} &= \{((q,q'),a,(r,r')) \mid (q,a,r) \in \delta_{\mathrm{shift}} \text{ and } \delta'_{\mathrm{shift}}(q',a) = r'\} \ , \\
\delta''_{\mathrm{pop}} &= \{((q,q'),(p,p'),(r,r')) \mid (q,p,r) \in \delta_{\mathrm{pop}} \text{ and } \delta'_{\mathrm{pop}}(q',p') = r'\} \ .
\end{aligned}$$

Then the weights of $\mathcal{C}$ are defined as

$$\begin{aligned}
\mathrm{wt}''_{\mathrm{push}}((q,q'),a,(r,r')) &= \mathrm{wt}_{\mathrm{push}}(q,a,r) \ , \\
\mathrm{wt}''_{\mathrm{shift}}((q,q'),a,(r,r')) &= \mathrm{wt}_{\mathrm{shift}}(q,a,r) \ , \\
\mathrm{wt}''_{\mathrm{pop}}((q,q'),(p,p'),(r,r')) &= \mathrm{wt}_{\mathrm{pop}}(q,p,r) \ .
\end{aligned}$$

Note that given a word w, the automata $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ have to use pushes, shifts, and pops at the same positions. Hence, every accepting run of $\mathcal{C}$ on w defines exactly one accepting run of $\mathcal{B}$ and exactly one accepting run of $\mathcal{A}$ on w with matching weights, and vice versa. We obtain

$$\begin{aligned}
[\![\mathcal{C}]\!](w) &= \sum_{\rho \in \mathrm{acc}(\mathcal{C},w)} \mathrm{wt}(\rho) \\
&= \sum_{\substack{\rho, \text{ such that} \\ \rho_{\restriction Q} \in \mathrm{acc}(\mathcal{A},w) \\ \rho_{\restriction Q'} \in \mathrm{acc}(\mathcal{B},w)}} \mathrm{wt}(\rho) \\
&= \left\{ \begin{matrix} \sum_{\rho \in \mathrm{acc}(\mathcal{A},w)} \mathrm{wt}(\rho) \ , & \text{if the run of } \mathcal{B} \text{ on } w \text{ is accepting} \\ 0 \ , & \text{otherwise} \end{matrix} \right. \\
&= (S \cap L)(w) \ .
\end{aligned}$$

It follows that, $[\![\mathcal{C}]\!] = S \cap L$.

For the second part of the proposition, let $\mathcal{A} = (Q, I, F, \delta, \mathrm{wt})$ and $\mathcal{B} = (Q', I', F', \delta', \mathrm{wt}')$ be two wOPA. We construct a wOPA $\mathcal{P}$ as $\mathcal{P} = (Q \times Q', I \times$

$I', F \times F', \delta^{\mathcal{P}}, \mathrm{wt}^{\mathcal{P}})$ where $\delta^{\mathcal{P}} = (\delta^{\mathcal{P}}_{\mathrm{push}}, \delta^{\mathcal{P}}_{\mathrm{shift}}, \delta^{\mathcal{P}}_{\mathrm{pop}})$ and set

$$\begin{aligned}
\delta^{\mathcal{P}}_{\mathrm{push}} &= \{((q,q'),a,(r,r')) \mid (q,a,r) \in \delta_{\mathrm{push}} \text{ and } (q',a,r') \in \delta'_{\mathrm{push}}\} \ , \\
\delta^{\mathcal{P}}_{\mathrm{shift}} &= \{((q,q'),a,(r,r')) \mid (q,a,r) \in \delta_{\mathrm{shift}} \text{ and } (q',a,r') \in \delta'_{\mathrm{shift}}\} \ , \\
\delta^{\mathcal{P}}_{\mathrm{pop}} &= \{((q,q'),(p,p'),(r,r')) \mid (q,p,r) \in \delta_{\mathrm{pop}} \text{ and } (q',p',r') \in \delta'_{\mathrm{pop}}\} \ ,
\end{aligned}$$

and

$$\begin{aligned}
\mathrm{wt}^{\mathcal{P}}_{\mathrm{push}}((q,q'),a,(r,r')) &= \mathrm{wt}'_{\mathrm{push}}(q,a,r) \cdot \mathrm{wt}''_{\mathrm{push}}(q',a,r') \ , \\
\mathrm{wt}^{\mathcal{P}}_{\mathrm{shift}}((q,q'),a,(r,r')) &= \mathrm{wt}'_{\mathrm{shift}}(q,a,r) \cdot \mathrm{wt}''_{\mathrm{shift}}(q',a,r') \ , \\
\mathrm{wt}^{\mathcal{P}}_{\mathrm{pop}}((q,q'),(p,p'),(r,r')) &= \mathrm{wt}'_{\mathrm{pop}}(q,p,r) \cdot \mathrm{wt}''_{\mathrm{pop}}(q',p',r') \ .
\end{aligned}$$

It follows that $[\![\mathcal{P}]\!] = [\![\mathcal{A}]\!] \odot [\![\mathcal{B}]\!]$. Furthermore, if $\mathcal{A}$ and $\mathcal{B}$ are restricted, then so is $\mathcal{P}$. □

Next, we show that regular series are closed under projections which preserve the OPM. For two OP alphabets (Σ, M), (Γ, M') and a mapping $h : \Sigma \to \Gamma$, we write $h : (\Sigma, M) \to (\Gamma, M')$ and say h is *OPM-preserving* if for all $\odot \in \{\lessdot, \doteq, \gtrdot\}$, we have $a \odot b$ if and only if $h(a) \odot h(b)$. We can extend such an h to a function $h : (\Sigma, M)^+ \to (\Gamma, M')^+$ as follows. Given a word $w = (a_1 a_2 ... a_n) \in (\Sigma, M)^+$, we define $h(w) = h(a_1 a_2 ... a_n) = h(a_1)h(a_2)...h(a_n)$. Let $S : (\Sigma, M)^+ \to K$ be a series. Then, we define $h(S) : (\Gamma, M')^+ \to K$ for each $v \in (\Gamma, M')^+$ by

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

Proposition 13. *Let $\mathbb{K}$ be a semiring, $S : (\Sigma, M)^+ \to K$ regular (resp. strictly regular), and $h : \Sigma \to \Gamma$ an OPM-preserving projection. Then, $h(S) : (\Gamma, M')^+ \to K$ is regular (resp. strictly regular).*

Proof. We follow an idea of [20] and its application in [18] and [11]. Let $\mathcal{A} = (Q, I, F, \delta, \mathrm{wt})$ be a wOPA over (Σ, M) and $\mathbb{K}$ with $[\![\mathcal{A}]\!] = S$. The main idea is to remember the last symbol read in the next state to distinguish different runs of $\mathcal{A}$ which would otherwise coincide in $\mathcal{B}$. We construct the wOPA $\mathcal{B} = (Q', I', F', \delta', \mathrm{wt}')$ over (Σ, M) and $\mathbb{K}$ as follows. We set $Q' = Q \times \Sigma$, $I' = I \times \{a_0\}$ for some fixed $a_0 \in \Sigma$, and $F' = F \times \Sigma$. We define the transition relations $\delta' = (\delta'_{\mathrm{push}}, \delta'_{\mathrm{shift}}, \delta'_{\mathrm{pop}})$ for every $b \in \Gamma$ and $(q,a),(q',a'),(q'',a'') \in Q'$, as

$$\begin{aligned}
\delta'_{\mathrm{push}} &= \{((q,a),b,(q',a')) \mid (q,a',q') \in \delta_{\mathrm{push}} \text{ and } b = h(a')\} \ , \\
\delta'_{\mathrm{shift}} &= \{((q,a),b,(q',a')) \mid (q,a',q') \in \delta_{\mathrm{shift}} \text{ and } b = h(a')\} \ , \\
\delta'_{\mathrm{pop}} &= \{((q,a),(q',a'),(q'',a)) \mid (q,q',q'') \in \delta_{\mathrm{pop}}\} \ .
\end{aligned}$$

Then, the weight functions are defined by

$$\begin{aligned}
\mathrm{wt}'_{\mathrm{push}}((q,a),h(a'),(q',a')) &= \mathrm{wt}_{\mathrm{push}}(q,a',q') \ , \\
\mathrm{wt}'_{\mathrm{shift}}((q,a),h(a'),(q',a')) &= \mathrm{wt}_{\mathrm{shift}}(q,a',q') \ , \\
\mathrm{wt}'_{\mathrm{pop}}((q,a),(q',a'),(q'',a'')) &= \mathrm{wt}_{\mathrm{pop}}(q,q',q'') \ .
\end{aligned}$$

Analogously to [18] and [11], this implies that for every run ρ of $\mathcal{A}$ on w, there exists exactly one run ρ' of $\mathcal{B}$ on v with $h(w) = v$ and $\mathrm{wt}(\rho)$=$\mathrm{wt}(\rho')$. One difference to previous works is that a pop of a wOPA is not consuming the symbol. Therefore, we have to make sure to not change the symbol, which we are currently remembering while processing a pop.

It follows that $[\![\mathcal{A}']\!](v) = h([\![\mathcal{A}]\!])(v)$, so $h(S) = [\![\mathcal{A}']\!]$ is regular. Furthermore, if $\mathcal{A}$ is restricted, then so is $\mathcal{B}$. □

4 A Nivat Theorem

In this section, we establish a connection between weighted OPLs and strictly regular series. We show that strictly regular 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.

Let $h : \Sigma' \to \Sigma$ be a map between two alphabets. Given an OP alphabet (Σ, M), we define $h^{-1}(M)$ by setting $h^{-1}(M)_{a'b'} = M_{h(a')h(b')}$ for all $a', b' \in \Sigma'$. As h is OPM-preserving, for every series $S : (\Sigma, M)^+ \to K$, we get a series $h(S) : (\Sigma', h^{-1}(M))^+ \to K$, 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 Σ', a map $h : \Sigma' \to \Sigma$, and a one-state rwOPA $\mathcal{B}$ over $(\Sigma', h^{-1}(M))$ and $\mathbb{K}$ and an OPL L over $(\Sigma', h^{-1}(M))$ such that $S = h([\![\mathcal{B}]\!] \cap L)$.

Now, we show that every strictly regular series can be decomposed into the above introduced fragments.

Proposition 14. *Let $S : (\Sigma, M)^+ \to K$ be a series. If S is strictly regular, then S is in $\mathcal{N}(A, B, \mathbb{K})$.*

Proof. We follow some ideas of [15] and [17].

Let $\mathcal{A} = (Q, I, F, \delta, \mathrm{wt})$ be a rwOPA over (Σ, M) and $\mathbb{K}$ with $[\![\mathcal{A}]\!] = S$. We set $\Sigma' = Q \times \Sigma \times Q$ as the extended alphabet. The intuition is that Σ' consists of the push and the shift transitions of $\mathcal{A}$. Let h be the projection of Σ' to Σ and let $M' = h^{-1}(M)$.

Let $L \subseteq (\Sigma', M')^+$ be the language consisting of all words w' over the extended alphabet such that $h(w')$ has an accepting run on $\mathcal{A}$ which uses at every position the push, resp. the shift transition defined by the symbol of Σ' at this position.

We construct the unweighted OPA $\mathcal{A}' = (Q', I', F', \delta')$ over (Σ', M'), accepting L, as follows. We set $Q' = Q$, $I' = I$, $F' = F$, and define δ' as follows

$$\begin{aligned}\delta'_{\text{push}} &= \{\, (q,(q,a,p),p) \mid (q,a,p) \in \delta_{\text{push}} \,\} \ , \\ \delta'_{\text{shift}} &= \{\, (q,(q,a,p),p) \mid (q,a,p) \in \delta_{\text{shift}} \,\} \ , \\ \delta'_{\text{pop}} &= \delta_{\text{pop}} \ .\end{aligned}$$

Hence, $\mathcal{A}'$ has an accepting run on a word $w' \in (\Sigma', M')^+$ if and only if $\mathcal{A}$ has an accepting run on $h(w')$, using the push and shift transitions defined by w'.

We construct the one-state rwOPA $\mathcal{B} = (Q'', I'', F'', \delta'', \text{wt}'')$ over (Σ', M') and $\mathbb{K}$ as follows. Set $Q'' = I'' = F'' = \{q\}$, $\delta''_{\text{push}} = \delta''_{\text{shift}} = \{(q,a',q) \mid a' \in \Sigma'\}$, $\delta''_{\text{pop}} = \{(q,q,q)\}$, $\text{wt}''_{\text{push}}(q,a',q) = \text{wt}_{\text{push}}(a')$, $\text{wt}''_{\text{shift}}(q,a',q) = \text{wt}_{\text{shift}}(a')$, for all $a' \in \Sigma'$, and $\text{wt}''_{\text{pop}}(q,q,q) = 1$.

Let ρ be a run of $w = a_1 ... a_n \in (\Sigma, M)^+$ on $\mathcal{A}$ and ρ' a run of $w' = a'_1 ... a'_n \in (\Sigma', M')^+$ on $\mathcal{B}$. We denote with $\text{wt}_{\mathcal{A}}(\rho, w, i)$, resp. $\text{wt}_{\mathcal{B}}(\rho', w', i)$, the weight of the push or shift transition used by the run ρ, resp. ρ', at position i. Since $\mathcal{A}$ and $\mathcal{B}$ are restricted, for all their runs ρ, ρ', we have $\text{wt}(\rho) = \prod_{i=1}^{|w|} \text{wt}_{\mathcal{A}}(\rho, w, i)$, resp. $\text{wt}(\rho') = \prod_{i=1}^{|w'|} \text{wt}_{\mathcal{B}}(\rho', w', i)$. Furthermore, following its definition, the rwOPA $\mathcal{B}$ has exactly one run ρ for every word $w' \in (\Sigma', M')$ and for all $h(w') = w$ and for all $i \in \{1...n\}$, we have $\text{wt}_{\mathcal{B}}(\rho', w', i) = \text{wt}_{\mathcal{A}}(\rho, w, i)$. It follows that

$$\begin{aligned} h([\![\mathcal{B}]\!] \cap L)(w) &= \sum_{\substack{w' \in (\Sigma', M')^+ \\ h(w') = w}} ([\![\mathcal{B}]\!] \cap L)(w') \\ &= \sum_{\substack{w' \in L(\mathcal{A}') \\ h(w') = w}} [\![\mathcal{B}]\!](w') \\ &= \sum_{\rho \in \text{acc}(\mathcal{A}, w)} \prod_{i=1}^{|w|} \text{wt}_{\mathcal{A}}(\rho, w, i) \\ &= \sum_{\rho \in \text{acc}(\mathcal{A}, w)} \text{wt}(\rho) \\ &= [\![\mathcal{A}]\!](w) = S(w) \ . \end{aligned}$$

Hence, $S = h([\![\mathcal{B}]\!] \cap L)$, thus $S \in \mathcal{N}(\Sigma, M, \mathbb{K})$. □

Using this proposition and closure properties of series, we get the following Nivat-Theorem for weighted operator precedence automata.

Theorem 15. *Let $\mathbb{K}$ be a semiring and $S : (\Sigma, M)^+ \to K$ be a series. Then S is strictly regular if and only if $S \in \mathcal{N}(\Sigma, M, \mathbb{K})$.*

Proof. The "only if"-part of is immediate by Proposition 14.

For the converse, let Σ' be an alphabet, $h : \Sigma' \to \Sigma$, $L \subseteq (\Sigma', h^{-1}(M))^+$ be an OPL, $\mathcal{B}$ a one-state rwOPA, and $S = h([\![\mathcal{B}]\!] \cap L)$. Then Proposition 12 shows that $[\![\mathcal{B}]\!] \cap L$ is strictly regular. Now, Proposition 13 yields the result. □

5 Weighted MSO-Logic for OPL

We use modified ideas from Droste and Gastin [12], also incorporating the distinction into an unweighted (boolean) and a weighted part by Bollig and Gastin [5].

Definition 16. We define the weighted logic $\mathrm{MSO}(\mathbb{K}, (\Sigma, M))$, short $\mathrm{MSO}(\mathbb{K})$, 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 $k \in \mathbb{K}$; x, y are first-order variables; and X is a second order variable.

We call β boolean and φ weighted formulas. Let $w \in (\Sigma, M)^+$ and $\varphi \in \mathrm{MSO}(\mathbb{K})$. Following classical approaches for logics , we denote with $[w] = \{1, ..., |w|\}$ the set of all positions of w. Let $\mathrm{free}(\varphi)$ be the set of all free variables in φ, and let $\mathcal{V}$ be a finite set of variables containing $\mathrm{free}(\varphi)$. 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 $\sigma[x \to i]$ as the $(\mathcal{V} \cup \{x\}, w)$-assignment mapping x to i and equaling σ everywhere else. The assignment $\sigma[X \to I]$ is defined analogously.

Consider the extended alphabet $\Sigma_\mathcal{V} = \Sigma \times \{0,1\}^\mathcal{V}$ together with its natural OPM $M_\mathcal{V}$ defined such that for all $(a, s), (b, t) \in \Sigma_\mathcal{V}$ and all $\odot \in \{\lessdot, \doteq, \gtrdot\}$, we have $(a, s) \odot (b, t)$ if and only if $a \odot b$. 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 regular property which can be checked by an OPA.

We define the *semantics* of $\varphi \in \mathrm{MSO}(\mathbb{K})$ as a function $[\![\varphi]\!]_\mathcal{V} : (\Sigma_\mathcal{V}, M)^+ \to K$ inductively for all valid $(w, \sigma) \in (\Sigma_\mathcal{V}, M)^+$, as seen in Fig. 8. For not valid (w, σ), we set $[\![\varphi]\!]_\mathcal{V}(w, \sigma) = 0$. We write $[\![\varphi]\!]$ for $[\![\varphi]\!]_{\mathrm{free}(\varphi)}$.

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

Fig. 8. Semantics

We write $[\![\varphi]\!]$ for $[\![\varphi]\!]_{\mathrm{free}(\varphi)}$, so $[\![\varphi]\!] : (\Sigma_{\mathrm{free}(\varphi)}, M)^+ \to K$. If φ contains no free variables, φ is a *sentence* and $[\![\varphi]\!] : (\Sigma, M)^+ \to K$.

Example 17. Let us go back to the automaton $\mathcal{A}_{\text{policy}}$ depicted in Figure 4. 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.

$$\begin{aligned}\beta = \quad & x \in X_0 \leftrightarrow \exists y \exists z (y > x \wedge z > x \wedge \text{Lab}_\$(y) \wedge \text{Lab}_\$(z)) \\ & \wedge x \in X_1 \leftrightarrow \exists y \exists z \begin{pmatrix} y \leq x \leq z \wedge \text{Lab}_\$(y) \wedge \text{Lab}_\$(z) \\ \wedge (x \neq y \wedge x \neq z \rightarrow \neg\, \text{Lab}_\$(x)) \end{pmatrix} \\ & \wedge x \in X_2 \leftrightarrow \exists y \exists z (y < x \wedge z < x \wedge \text{Lab}_\$(y) \wedge \text{Lab}_\$(z)) \ .\end{aligned}$$

Weight assignment is formalized by

$$\varphi_{0,2} = \neg((x \in X_0 \vee x \in X_2) \wedge (\text{Lab}_{\text{call}}(x) \vee \text{Lab}_{\text{ret}}(x) \vee \text{Lab}_{\text{int}}(x))) \oplus 0 \ ,$$

which assigns weight 0 to calls, returns, and ints outside portion X_1; and

$$\begin{aligned}\varphi_1 = \quad & (\neg(x \in X_1 \wedge \text{Lab}_{\text{call}}(x)) \oplus 1) \\ & \otimes (\neg(x \in X_1 \wedge \text{Lab}_{\text{ret}}(x)) \oplus -1) \\ & \otimes (\neg(x \in X_1 \wedge \text{Lab}_{\text{int}}(x)) \oplus 0) \\ & \otimes (\neg\, \text{Lab}_\$(x) \oplus 0) \ ,\end{aligned}$$

which assigns weights $1, -1, 0$ to calls, returns, and ints, respectively, within portion X_1.

Then, the formula $\psi = \prod_x (\beta \otimes \varphi_{0,2} \otimes \varphi_1)$ defines the weight assigned by $\mathcal{A}_{\text{policy}}$ 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 $\mathcal{A}_{\text{policy}}$.

Lemma 18. *Let $\varphi \in \text{MSO}(\mathbb{K})$ and let $\mathcal{V}$ be a finite set of variables with $\text{free}(\varphi) \subseteq \mathcal{V}$. Then, $[\![\varphi]\!]_{\mathcal{V}}(w, \sigma) = [\![\varphi]\!](w, \sigma|_{\text{free}(\varphi)})$ for each valid $(w, \sigma) \in (\Sigma_{\mathcal{V}}, M)^+$. Furthermore, $[\![\varphi]\!]$ is regular (resp. strictly regular) iff $[\![\varphi]\!]_{\mathcal{V}}$ is regular (resp. strictly regular).*

Proof. This is shown by means of Proposition 13 analogously to Proposition 3.3 of [12]. □

As shown by [12] 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. The main idea for this is to allow only functions with finitely many different values (step functions) after a product quantification. Furthermore, in the non-commutative case, we either also restrict the application of $\otimes$ to step functions or we enforce all occurring weights (constants) of $\varphi \otimes \theta$ to commute.

Definition 19. The *set of almost boolean formulas* is the smallest set of all formulas of $\text{MSO}(\mathbb{K})$ containing all constants $k \in \mathbb{K}$ and all boolean formulas which is closed under $\oplus$ and $\otimes$.

The following propositions show that almost boolean formulas are describing precisely a certain form of rwOPA's behaviors, which we call *OPL step functions*. We adapt ideas from [16].

Definition 20. For $k \in \mathbb{K}$ and a language $L \subseteq (\Sigma, M)^+$, we define $\mathbb{1}_L : (\Sigma, M)^+ \to \mathbb{K}$, the *characteristic series* of L, i.e. $\mathbb{1}_L(w) = 1$ if $w \in L$, and $k\mathbb{1}_L(w) = 0$ otherwise. We denote by $k\mathbb{1}_L : (\Sigma, M)^+ \to \mathbb{K}$ the characteristic series of L multiplied by k, i.e. $k\mathbb{1}_L(w) = k$ if $w \in L$, and $k\mathbb{1}_L(w) = 0$ otherwise.

A series S is called an *OPL step function*, if it has a representation

$$S = \sum_{i=1}^{n} k_i \mathbb{1}_{L_i} \ ,$$

where L_i are OPL forming a partition of $(\Sigma, M)^+$ and $k_i \in \mathbb{K}$ for each $i \in \{1, ..., n\}$; so $[\![\varphi]\!](w) = k_i$ iff $w \in L_i$, for each $i \in \{1, ..., n\}$.

Lemma 21. *The set of all OPL step functions is closed under $+$ and $\odot$.*

Proof. Let $S = \sum_{i=1}^{k} k_i \mathbb{1}_{L_i}$ and $S' = \sum_{j=1}^{\ell} k'_j \mathbb{1}_{L'_j}$ be OPL step functions. Then the following holds

$$S + S' = \sum_{i=1}^{k} \sum_{j=1}^{\ell} (d_i + d'_j) \mathbb{1}_{L_i \cap L'_j} \ ,$$

$$S \odot S' = \sum_{i=1}^{k} \sum_{j=1}^{\ell} (d_i \cdot d'_j) \mathbb{1}_{L_i \cap L'_j} \ .$$

Since $(L_i \cap L'_j)$ are also OPL and form a partition of $(\Sigma, M)^+$, it follows that $S + S'$ and $S \odot S'$ are also OPL step functions. □

Proposition 22. *(a) For every almost boolean formula φ, $[\![\varphi]\!]$ is an OPL step function.*

(b) If S is an OPL step function, then there exists an almost boolean formula φ such that $S = [\![\varphi]\!]$.

Proof. (a) We show the first statement by structural induction on φ. If φ is boolean, then $[\![\varphi]\!] = \mathbb{1}_{L(\varphi)}$, were $L(\varphi)$ and $L(\neg\varphi)$ are OPL due to Theorem 3. Therefore, $[\![\varphi]\!] = 1_K \mathbb{1}_{L(\varphi)} + 0_K \mathbb{1}_{L(\neg\varphi)}$ is an OPL step function. If $\varphi = k$, $k \in \mathbb{K}$, then $[\![k]\!] = k\mathbb{1}_{(\Sigma,M)^+}$ is an OPL step function. Let $\mathcal{V} = \text{free}(\varphi_1) \cup \text{free}(\varphi_2)$. By lifting Lemma 18 to OPL step functions as in [17] and by Lemma 21, we see that $[\![\varphi_1 \oplus \varphi_2]\!] = [\![\varphi_1]\!]_\mathcal{V} + [\![\varphi_2]\!]_\mathcal{V}$ and $[\![\varphi_1 \otimes \varphi_2]\!] = [\![\varphi_1]\!]_\mathcal{V} \odot [\![\varphi_2]\!]_\mathcal{V}$ are also OPL step functions.

(b) Given an OPL step function $[\![\varphi]\!] = \sum_{i=1}^{n} k_i \mathbb{1}_{L_i}$, we use Theorem 3 to get φ_i with $[\![\varphi_i]\!] = \mathbb{1}_{L_i}$. Then, the second statement follows from setting $\varphi = \bigvee_i^n (k_i \wedge \varphi_i)$ and the fact that the OPL $(L_i)_{1 \le i \le n}$ form a partition of $(\Sigma, M)^+$. □

Proposition 23. *Let S be an OPL step function. Then S is strictly regular.*

Proof. Let $n \in \mathbb{N}$, $(L_i)_{1 \leq i \leq n}$ be OPL forming a partition of $(\Sigma, M)^+$ and $k_i \in \mathbb{K}$ for each $i \in \{1, ..., n\}$ such that

$$S = \sum_{i=1}^{n} k_i \mathbb{1}_{L_i} \ .$$

Its easy to construct a 2 state rwOPA recognizing the constant series $[\![k_i]\!]$ which assigns the weight k_i to every word. Hence, $k_i \mathbb{1}_{L_i} = [\![k_i]\!] \cap L_i$ is strictly regular by Proposition 12. Therefore, by Proposition 11, S is strictly regular. □

Definition 24. Let $\varphi \in \mathrm{MSO}(\mathbb{K})$. We denote by $\mathrm{const}(\varphi)$ all weights of $\mathbb{K}$ occurring in φ and we call φ $\otimes$-*restricted* if for all subformulas $\psi \otimes \theta$ of φ either ψ is almost boolean or $\mathrm{const}(\psi)$ and $\mathrm{const}(\theta)$ commute elementwise. We call φ $\prod$-*restricted* if for all subformulas $\prod_x \psi$ of φ, ψ is almost boolean. We call φ *restricted* if it is both $\otimes$- and $\prod$-restricted.

In Example 17, the formula β is boolean, the formulas ϕ are almost boolean, and ψ and χ are restricted. Notice that ψ and χ would be restricted even if $\mathbb{K}$ were not commutative.

For use in Section 6, we note:

Proposition 25. *Let $S : (\Sigma, M)^+ \to K$ be a regular (resp. strictly regular) series and $k \in \mathbb{K}$. Then $[\![k]\!] \odot S$ is regular (resp. strictly regular).*

Proof. Let $\mathcal{A} = (Q, I, F, \delta, \mathrm{wt})$ be an (r)wOPA such that $[\![\mathcal{A}]\!] = S$. Then we construct an rwOPA $\mathcal{B} = (Q', I', F, \delta', \mathrm{wt}')$ as follows.

We set $Q \cup I'$ and $I' = \{q'_I \mid q_I \in I\}$. The new transition relations δ' and weight functions wt' consists of all transitions of $\mathcal{A}$ with their respective weights and the following additional transitions: For every push transition (q_I, a, q) of δ_{push}, we add a push transition (q'_I, a, q) to δ'_{push} with $\mathrm{wt}'_{\mathrm{push}}(q'_I, a, q) = k \cdot \mathrm{wt}_{\mathrm{push}}(q_I, a, q)$.

Note that every run of an (w)OPA has to start with a push transition. Therefore, the two automata have the same respective runs, but $\mathcal{B}$ is exactly once in a state $q'_I \in I$. This together with the weight assignment ensures that $\mathcal{B}$ uses the same weights as $\mathcal{A}$ except at the very first transition of every run which is multiplied by k from the left. In particular, we do not change the weight of any pop transition. It follows that $[\![\mathcal{B}]\!] = [\![k]\!] \odot S$. Also, if $\mathcal{A}$ is restricted, so is $\mathcal{B}$. □

6 Characterization of Regular Series

Lemma 26 (Closure under weighted disjunction). *Let φ and ψ be two formulas of* $\mathrm{MSO}(\mathbb{K})$ *such that $[\![\varphi]\!]$ and $[\![\psi]\!]$ are regular (resp. strictly regular). Then, $[\![\varphi \oplus \psi]\!]$ is regular (resp. strictly regular).*

Proof. We put $\mathcal{V} = \mathrm{free}(\varphi) \cup \mathrm{free}(\psi)$. Then, $[\![\varphi \oplus \psi]\!] = [\![\varphi]\!]_{\mathcal{V}} + [\![\psi]\!]_{\mathcal{V}}$ is regular (resp. strictly regular) by Lemma 18 and Proposition 11. □

Proposition 27 (Closure under restricted weighted conjunction). *Let $\psi \otimes \theta$ be a subformula of a $\otimes$-restricted formula φ of* MSO($\mathbb{K}$) *such that $[\![\psi]\!]$ and $[\![\theta]\!]$ are regular (resp. strictly regular). Then, $[\![\psi \otimes \theta]\!]$ is regular (resp. strictly regular).*

Proof. Since φ is $\otimes$-restricted, either ψ is almost boolean or the constants of both formulas commute.

Case 1: Let us assume ψ is almost boolean. Then, we can write $[\![\psi]\!]$ as OPL step function, i.e., $[\![\psi]\!] = \sum_{i=1}^{n} k_i \mathbb{1}_{L_i}$, where L_i are OPL. So, the series $[\![\psi \otimes \theta]\!]$ equals a sum of series of the form $([\![k_i \otimes \theta]\!] \cap L_i)$. Then, by Proposition 25, $[\![k_i \otimes \theta]\!]$ is a regular (resp. strictly regular) series. Therefore, $([\![k_i \otimes \theta]\!] \cap L_i)$ is regular (resp. strictly regular) by Proposition 12. Hence, $[\![\psi \otimes \theta]\!]$ is (strictly) regular by Proposition 11.

Case 2: Let us assume that the constants of ψ and θ commute. Then, the second part of Proposition 12 yields the claim. □

Lemma 28 (Closure under $\sum_x$, $\sum_X$). *Let φ be a formula of* MSO($\mathbb{K}$) *such that $[\![\varphi]\!]$ is regular (resp. strictly regular). Then, $[\![\sum_x \varphi]\!]$ and $[\![\sum_X \varphi]\!]$ are regular (resp. strictly regular).*

Proof (Compare [12]). Let $\mathcal{X} \in \{x, X\}$ and $\mathcal{V} = \text{free}(\sum_{\mathcal{X}} \varphi)$. We define $\pi : (\Sigma_{\mathcal{V} \cup \{\mathcal{X}\}}, M)^+ \to (\Sigma_{\mathcal{V}}, M)^+$ by $\pi(w, \sigma) = (w, \sigma|_{\mathcal{V}})$ for any $(w, \sigma) \in (\Sigma_{\mathcal{V} \cup \{\mathcal{X}\}}, M)^+$. Then, for $(w, \gamma) \in (\Sigma_{\mathcal{V}}, M)^+$, the following holds

$$\begin{aligned}
[\![\textstyle\sum_X \varphi]\!](w, \gamma) &= \sum_{I \subseteq \{1, \dots, |w|\}} [\![\varphi]\!]_{\mathcal{V} \cup \{X\}}(w, \gamma[X \to I]) \\
&= \sum_{(w,\sigma) \in \pi^{-1}(w,\gamma)} [\![\varphi]\!]_{\mathcal{V} \cup \{X\}}(w, \sigma) \\
&= \pi([\![\varphi]\!]_{\mathcal{V} \cup \{X\}})(w, \gamma) \ .
\end{aligned}$$

Analogously, we show that $[\![\sum_x \varphi]\!](w, \gamma) = \pi([\![\varphi]\!]_{\mathcal{V} \cup \{x\}})(w, \gamma)$ for all $(w, \gamma) \in (\Sigma_{\mathcal{V}}, M)^+$. By Lemma 18, $[\![\varphi]\!]_{\mathcal{V} \cup \{\mathcal{X}\}}$ is regular because $\text{free}(\varphi) \subseteq \mathcal{V} \cup \{\mathcal{X}\}$. Then, $[\![\sum_{\mathcal{X}} \varphi]\!]$ is regular by Proposition 13. □

Proposition 29 (Closure under restricted $\prod_x$). *Let φ be an almost boolean formula of* MSO($\mathbb{K}$). *Then, $[\![\prod_x \varphi]\!]$ is strictly regular.*

Proof. We use ideas of [12] and the extensions in [18] and [11] with the following intuition.

In the first part, we write $[\![\varphi]\!]$ as OPL step function and encode the information to which language $(w, \sigma[x \to i])$ belongs in a specially extended language $\tilde{L}$. Then we construct an MSO-formula for this language. Therefore, by Theorem 3, we get a deterministic OPA recognizing $\tilde{L}$. In the second part, we add the weights k_i to this automaton and return to our original alphabet.

More detailed, let $\varphi \in \text{MSO}(\mathbb{K}, (\Sigma, M))$. We define $\mathcal{V} = \text{free}(\prod x.\varphi)$ and $\mathcal{W} = \text{free}(\varphi) \cup \{x\}$. We consider the extended alphabets $\Sigma_{\mathcal{V}}$ and $\Sigma_{\mathcal{W}}$ together with their natural OPMs $M_{\mathcal{V}}$ and $M_{\mathcal{W}}$. By Proposition 22 and lifting Lemma

18 to OPL step functions, $[\![\varphi]\!]$ is an OPL step function. Let $[\![\varphi]\!] = \sum_{j=1}^{m} k_j \mathbb{1}_{L_j}$ where L_j is an OPL over $(\Sigma_{\mathcal{W}}, M_{\mathcal{W}})$ for all $j \in \{1, ..., m\}$ and (L_j) is a partition of $(\Sigma_{\mathcal{W}}, M_{\mathcal{W}})^+$. By the semantics of the product quantifier, we get

$$\begin{aligned}[\![\textstyle\prod_x \varphi]\!](w, \sigma) &= \prod_{i \in [w]} ([\![\varphi]\!]_{\mathcal{W}}(w, \sigma[x \to i])) \\ &= \prod_{i \in [w]} (k_{g(i)}),\end{aligned}$$

$$\text{where} \qquad g(i) = \begin{cases} 1 \ , \text{ if } (w, \sigma[x \to i]) \in L_1 \\ \dots \\ m \ , \text{ if } (w, \sigma[x \to i]) \in L_m \end{cases} , \text{ for all } i \in [w] \ . \tag{2}$$

Now, in the first part, we encode the information to which language $(w, \sigma[x \to i])$ belongs in a specially extended language $\tilde{L}$ and construct an MSO-formula for this language. We define the extended alphabet $\tilde{\Sigma} = \Sigma \times \{1, ..., n\}$, together with its natural OPM $\tilde{M}$ which only refers to Σ, so:

$$(\tilde{\Sigma}_{\mathcal{V}}, \tilde{M}_{\mathcal{V}})^+ = \{(w, g, \sigma) \mid (w, \sigma) \in (\Sigma_{\mathcal{V}}, M_V) \text{ and } g \in \{1, ..., m\}^{[w]}\} \ .$$

We define the languages $\tilde{L}, \tilde{L}_j, \tilde{L}'_j \subseteq (\tilde{\Sigma}_{\mathcal{V}}, \tilde{M}_{\mathcal{V}})^+$ as follows:

$$\tilde{L} = \left\{ (w, g, \sigma) \;\middle|\; \begin{array}{l} (w, \sigma) \in (\tilde{\Sigma}_{\mathcal{V}}, \tilde{M}_{\mathcal{V}})^+ \text{ is valid and} \\ \text{for all } i \in [w],\ j \in \{1, ..., m\} :\ g(i) = j \Rightarrow (w, \sigma[x \to i]) \in L_j \end{array} \right\} \ ,$$

$$\tilde{L}_j = \left\{ (w, g, \sigma) \;\middle|\; \begin{array}{l} (w, \sigma) \in (\tilde{\Sigma}_{\mathcal{V}}, \tilde{M}_{\mathcal{V}})^+ \text{ is valid and} \\ \text{for all } i \in [w] :\ g(i) = j \Rightarrow (w, \sigma[x \to i]) \in L_j \end{array} \right\} \ ,$$

$$\tilde{L}'_j = \{ \, (w, g, \sigma) \mid \text{for all } i \in [w] :\ g(i) = j \Rightarrow (w, \sigma[x \to i]) \in L_j \, \} \ .$$

Then, $\tilde{L} = \bigcap_{j=1}^{m} \tilde{L}_j$. Hence, in order to show that $\tilde{L}$ is an OPL, it suffices to show that each $\tilde{L}_j$ is an OPL. By a standard procedure, compare [12], we obtain a formula $\tilde{\varphi}_j \in \mathrm{MSO}(\tilde{\Sigma}_{\mathcal{V}}, \tilde{M}_{\mathcal{V}})$ with $L(\tilde{\varphi}_j) = \tilde{L}'_j$. Therefore, by Theorem 3, $\tilde{L}'_j$ is an OPL. It is straightforward to define an OPA accepting $\tilde{N}_{\mathcal{V}}$, the language of all valid words. By closure under intersection, $\tilde{L}_j = \tilde{L}'_j \cap \tilde{N}_{\mathcal{V}}$ is also an OPL and so is $\tilde{L}$. Hence, there exists a deterministic OPA $\tilde{\mathcal{A}} = (\Sigma, q_0, F, \tilde{\delta})$ recognizing $\tilde{L}$.

In the second part, we add weights to $\tilde{\mathcal{A}}$ as follows. We construct the wOPA $\mathcal{A} = (Q, I, F, \delta, \mathrm{wt})$ over $(\Sigma_{\mathcal{V}}, M_{\mathcal{V}})$ and $\mathbb{K}$ by adding to every transition of $\tilde{\mathcal{A}}$ with $g(i) = j$ the weight k_j.

That is, we keep the states, the initial state, and the accepting states, and for $\delta = (\delta_{\text{push}}, \delta_{\text{shift}}, \delta_{\text{pop}})$ and all $q, q', p \in Q$ and $(a, j, s) \in \tilde{\Sigma}_{\mathcal{V}}$, we define

$$\delta_{\text{push/shift}}(q, (a, s), q') = \begin{cases} k_j \ , \text{ if } (q, (a, j, s), q') \in \tilde{\delta}_{\text{push/shift}} \\ 0 \ , \text{ otherwise} \end{cases} \ .$$

Since $\tilde{\mathcal{A}}$ is deterministic, for every $(w, g, \sigma) \in \tilde{L}$, there exists exactly one accepted run $\tilde{r}$ of $\tilde{\mathcal{A}}$. On the other hand, for every $(w, g, \sigma) \notin \tilde{L}$, there is no accepted run

of $\tilde{\mathcal{A}}$. Since (L_j) is a partition of $(\Sigma_{\mathcal{W}}, M_{\mathcal{W}})^+$, for every $(w,\sigma) \in (\Sigma_{\mathcal{V}}, M_{\mathcal{V}})$, there exists exactly one g with $(w,g,\sigma) \in \tilde{L}$. Thus, every $(w,\sigma) \in (\Sigma_{\mathcal{V}}, M_{\mathcal{V}})$ has exactly one run r of $\mathcal{A}$ determined by the run $\tilde{r}$ of (w,g,σ) of $\tilde{\mathcal{A}}$. We denote with $\mathrm{wt}_{\mathcal{A}}(r,(w,\sigma),i)$ the weight used by the run r on (w,σ) over $\mathcal{A}$ at position i, which is always the weight of the push or shift transition used at this position. Then by definition of $\mathcal{A}$ and $\tilde{L}$, the following holds for all $i \in [w]$

$$g(i) = j \Rightarrow \mathrm{wt}_{\mathcal{A}}(r,(w,\sigma),i) = k_j \wedge (w, \sigma[x \to i]) \in L_j \ .$$

By formula (2), we obtain

$$[\![\varphi]\!]_{\mathcal{W}}(w, \sigma[x \to i]) = k_j = \mathrm{wt}_{\mathcal{A}}(r,(w,\sigma),i) \ .$$

Hence, for the behavior of the automaton $\mathcal{A}$ the following holds

$$\begin{aligned}
[\![\mathcal{A}]\!](w,\sigma) &= \sum_{r' \in \mathrm{acc}(\mathcal{A},w)} \mathrm{wt}(r') \\
&= \prod_{i=1}^{|w|} \mathrm{wt}_{\mathcal{A}}(r,(w,\sigma),i) \\
&= \prod_{i=1}^{|w|} [\![\varphi]\!]_{\mathcal{W}}(w, \sigma[x \to i]) \\
&= [\![\textstyle\prod_x \varphi]\!](w,\sigma) \ .
\end{aligned}$$

Thus, $\mathcal{A}$ recognizes $[\![\prod_x \varphi]\!]$. □

The following proposition is a summary of the previous results.

Proposition 30. *For every restricted* MSO($\mathbb{K}$)*-sentence φ, there exists an rwOPA $\mathcal{A}$ with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$.*

Proof. We use structural induction on φ. If φ is an almost boolean formula, then by Proposition 22 $[\![\varphi]\!]$ is an OPL step function. By Proposition 23 every OPL step function is strictly regular.

Closure under $\oplus$ is dealt with by Lemma 26, closure under $\otimes$ by Proposition 27. The sum quantifications $\sum_x$ and $\sum_X$ are dealt with by Lemma 28. Since φ is restricted, we know that for every subformula $\bigotimes_x \psi$, the formula ψ is an almost boolean formula. Therefore, we can apply Proposition 29 to maintain recognizability of our formula in this case.

The next proposition shows that the converse also holds.

Proposition 31. *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 [12,18] with the one of [28] 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; $X^{\text{pop}}_{p,q,r}$ 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.

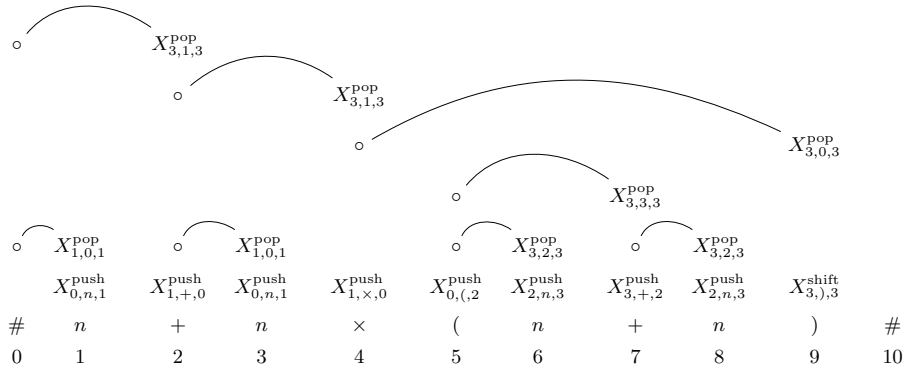


Fig. 9. The string of Figure 1 with the second order variables evidenced for the automaton of Figure 2. The symbol ∘ marks the positions of the symbols that precede the push corresponding to the bound pop transition.

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. $(p, q, r) \in \delta_{\text{pop}}$. Since Σ and Q are finite, there is an enumeration $\bar{X} = (X_1, .., X_m)$ of all variables of $\mathcal{V}$. We denote by $\bar{X}^{\text{push}}$, $\bar{X}^{\text{shift}}$, and $\bar{X}^{\text{pop}}$ enumerations over only the respective set of second order variables.

We use the following usual abbreviations for unweighted formulas of MSO:

$$
\begin{aligned}
(\beta \wedge \varphi) &= \neg(\neg\beta \vee \neg\varphi),\\
(\beta \rightarrow \varphi) &= (\neg\beta \vee \varphi),\\
(\beta \leftrightarrow \varphi) &= (\beta \rightarrow \varphi) \wedge (\varphi \rightarrow \beta),\\
(\forall x.\varphi) &= \neg(\exists x.\neg\varphi),\\
(y = x) &= (x \leq y) \wedge (y \leq x),\\
(y = x + 1) &= (x \leq y) \wedge \neg(y \leq x) \wedge \forall z.(z \leq x \vee y \leq z),\\
\min(x) &= \forall y.(x \leq y),\\
\max(x) &= \forall y.(y \leq x),
\end{aligned}
$$

Additionally, we use the shortcuts $\mathrm{Tree}(x,z,v,y)$, $\mathrm{Next}_i(x,y)$, $Q_i(x,y)$, and $\mathrm{Tree}_{p,q}(x,z,v,y)$, originally defined in [28], reported and adapted here for convenience:

$$x \circ y := \bigvee_{a,b\in\Sigma, M_{a,b}=\circ} \mathrm{Lab}_a(x) \wedge \mathrm{Lab}_b(y), \text{ for } \circ \in \{\lessdot, \doteq, \gtrdot\}$$

$$\mathrm{Tree}(x,z,v,y) := x \curvearrowright y \wedge \begin{pmatrix} (x+1=z \ \vee \ x \curvearrowright z) \wedge \neg\exists t(z<t<y \wedge x \curvearrowright t) \\ \wedge \\ (v+1=y \ \vee \ v \curvearrowright y) \wedge \neg\exists t(x<t<v \wedge t \curvearrowright y) \end{pmatrix}$$

In other words, Tree holds among the four positions (x,z,v,y) iff, at the time when a pop transition is executed: x (resp. y) is the rightmost leaf at the left (resp. the leftmost at the right) of the subtree whose scanning (and construction if used as a parser) is completed by the OPA through the current transition; z and y are the leftmost and rightmost terminal characters of the right hand side of the grammar production that is reduced by the pop transition of the OPA [28]. For instance, with reference to Figures 1 and 9, $\mathrm{Tree}(5,7,7,9)$ and $\mathrm{Tree}(4,5,9,10)$ hold.

$$\mathrm{Succ}_q(x,y) := (x+1=y) \wedge \bigvee_{p\in Q, a\in\Sigma} (x \in X^{\mathrm{push}}_{p,a,q} \vee x \in X^{\mathrm{shift}}_{p,a,q} \vee \min(x))$$

I.e., y is the position adjacent to x, $\mathrm{Lab}_a(y)$ and, while reading a, the OPA reaches state q, either through a push or through a shift move.

$$\mathrm{Next}_r(x,y) := \exists z \exists v. \left(\mathrm{Tree}(x,z,v,y) \wedge \bigvee_{p,q\in Q} v \in X^{\mathrm{pop}}_{p,q,r} \right)$$

I.e., $\mathrm{Next}_r(x,y)$ holds when a pop move reduces a subtree enclosed between positions x and y reaching state r.

$$Q_i(x,y) := \mathrm{Succ}_i(x,y) \vee \mathrm{Next}_i(x,y)$$

Finally,

$$\mathrm{Tree}_{i,j}(x,z,v,y) := \mathrm{Tree}(x,z,v,y) \wedge Q_i(v,y) \wedge Q_j(x,z)$$

refines the predicate Tree by making explicit that i and j are, respectively, the current state and the state on top of the stack when the pop move is executed.

We now define the unweighted formula ψ to characterize all accepted runs

$$\psi = \mathit{Partition}(\bar{X}^{\mathrm{push}}, \bar{X}^{\mathrm{shift}}) \wedge \mathit{Unique}(\bar{X}^{\mathrm{pop}}) \wedge \mathit{InitFinal} \\ \wedge \mathit{Trans}_{\mathrm{push}} \wedge \mathit{Trans}_{\mathrm{shift}} \wedge \mathit{Trans}_{\mathrm{pop}} \ .$$

Here, the subformula *Partition* will enforce the push and shift sets to be (together) a partition of all positions. $\mathit{InitFinal}$ controls the initial and the acceptance condition and Trans_{op} the transitions of the run together with the

labels.

$$Partition(X_1,...,X_n) = \forall x. \bigvee_{i=1}^{n} \big[(x \in X_i) \wedge \bigwedge_{i \neq j} \neg(x \in X_j)\big] \ ,$$

$$Unique(X_1^{\text{pop}},..,X_n^{\text{pop}}) = \forall x. \bigwedge_{i \neq j} \neg(x \in X_i^{\text{pop}} \wedge x \in X_j^{\text{pop}}) \ ,$$

$$\begin{aligned} InitFinal = \exists x \exists y \exists x' \exists y'. \big[& \min(x) \wedge \max(y) \wedge x + 1 = x' \wedge y' + 1 = y \\ & \wedge \bigvee_{\substack{i \in I,\, q \in Q \\ a \in \Sigma}} x' \in X_{i,a,q}^{\text{push}} \\ & \wedge \bigvee_{\substack{f \in F,\, q \in Q \\ a \in \Sigma}} (y' \in X_{q,a,f}^{\text{push}} \vee y' \in X_{q,a,f}^{\text{shift}}) \\ & \wedge \bigvee_{f \in F} (\text{Next}_f(x,y) \wedge \bigwedge_{j \neq f} \neg \text{Next}_j(x,y))\big] \ , \end{aligned}$$

$$Trans_{\text{push}} = \forall x. \bigwedge_{p,q \in Q, a \in \Sigma} \big(x \in X_{p,a,q}^{\text{push}} \rightarrow \big[\text{Lab}_a(x) \wedge \exists z.(z \lessdot x \wedge Q_p(z,x))\big]\big)$$

$$Trans_{\text{shift}} = \forall x. \bigwedge_{p,q \in Q, a \in \Sigma} \big(x \in X_{p,a,q}^{\text{shift}} \rightarrow \big[\text{Lab}_a(x) \wedge \exists z.(z \doteq x \wedge Q_p(z,x))\big]\big) \ .$$

I.e., if $x \in X_{p,a,q}^{\text{push}}$ (resp. X^{shift}) the formula holds in a run where, reading character a in position x, the automaton performs a push (resp. a shift) reaching state q from p; this may occur when $z \lessdot x$ (resp., $z \doteq x$) is immediately adjacent to x or after a subtree between positions z and x has been built. Notice that the converse too of the above implications holds, due to the fact that the whole set of string positions is partitioned into the two disjoint sets X^{push}, X^{shift}.

$$Trans_{\text{pop}} = \forall v. \bigwedge_{p,q \in Q} \big(\big[\bigvee_{r \in Q} v \in X_{p,q,r}^{\text{pop}}\big] \leftrightarrow \big[\exists x \exists y \exists z.(\text{Tree}_{p,q}(x,z,v,y))\big]\big)$$

Thus, with arguments similar to [28] it can be shown that the sentences satisfying ψ are exactly those recognized by the unweighted OPA subjacent to $\mathcal{A}$.

For an unweighted formula β and two weights k_1 and k_2, we define the following shortcut for an almost boolean weighted formula:

$$\text{IF } \beta \text{ THEN } k_1 \text{ ELSE } k_2 = (\beta \otimes k_1) \oplus (\neg\beta \otimes k_2) \ .$$

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

$$\begin{aligned} \theta = \psi \otimes \prod_x \bigotimes_{p,q \in Q} \big(& \bigotimes_{a \in \Sigma} (\text{IF } x \in X_{p,a,q}^{\text{push}} \text{ THEN } \text{wt}_{\text{push}}(p,a,q) \text{ ELSE } 1) \\ & \otimes \bigotimes_{a \in \Sigma} (\text{IF } x \in X_{p,a,q}^{\text{shift}} \text{ THEN } \text{wt}_{\text{shift}}(p,a,q) \text{ ELSE } 1) \\ & \otimes \bigotimes_{r \in Q} (\text{IF } x \in X_{p,q,r}^{\text{pop}} \text{ THEN } \text{wt}_{\text{pop}}(p,q,r) \text{ ELSE } 1)\big) \ . \end{aligned}$$

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. Furthermore, ψ is boolean and so θ is $\otimes$-restricted. Thus, θ is a restricted formula.

Finally, we define

$$\varphi = \bigoplus_{X_1} \bigoplus_{X_2} \cdots \bigoplus_{X_m} \theta \ .$$

This implies $[\![\varphi]\!](w) = [\![\mathcal{A}]\!](w)$, for all $w \in (\Sigma, M)^+$. Therefore, φ is our required sentence with $[\![\mathcal{A}]\!] = [\![\varphi]\!]$. □

The following theorem summarizes the main results of this section.

Theorem 32. *Let $\mathbb{K}$ be a semiring and $S : (\Sigma, M)^+ \to K$ a series.*

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

Theorem 32 documents a further step in the path of generalizing a series of results beyond the barrier of regular and structured –or visible– CFLs. Up to a few years ago, major properties of regular languages, such as closure w.r.t. all main language operations, decidability results, logic characterization, and, in this case, weighted language versions, could be extended to several classes of structured CFLs, among which the VPL one certainly obtained much attention. OPLs further generalize the above results not only in terms of strict inclusion, but mainly because they are not visible, in the sense explained in the introduction, nor are they necessarily real-time: this allows them to cover important applications that could not be adequately modeled through more restricted classes.

Theorem 32 also 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. This is due to the fact that pop transitions are applied in the reverse order than that of positions to which they refer (position v in formula $Trans_{\mathrm{pop}}$). Notice, however, that rwOPL do not forbid unbounded pop sequences; thus, they too include languages that are neither real-time nor visible. This remark naturally raises new intriguing questions which we will briefly address in the conclusion.

7 Conclusion

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.

This 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 [19], very similar problems arose for weighted automata on unranked trees and weighted MSO logic. In [13], 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 [28], operator precedence languages of infinite words were investigated and shown to be practically important. Therefore, the problem arises to develop a theory of wOPA on infinite words. In order to define their infinitary quantitative behaviors, one could try to use valuation monoids as in [16].

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 [25].

References

1. Alur, R., Fisman, D.: Colored nested words. In: Dediu, A.H., Janousek, J., Martín-Vide, C., Truthe, B. (eds.) Language and Automata Theory and Applications, LATA 2016. LNCS, vol. 9618, pp. 143–155. Springer (2016)
2. Alur, R., Madhusudan, P.: Adding nesting structure to words. J. ACM 56(3), 16:1–16:43 (2009)
3. Barenghi, A., Crespi Reghizzi, S., Mandrioli, D., Panella, F., Pradella, M.: Parallel parsing made practical. Sci. Comput. Program. 112(3), 195–226 (2015)
4. Berstel, J., Reutenauer, C.: Rational Series and Their Languages, EATCS Monographs in Theoretical Computer Science, vol. 12. Springer (1988)
5. Bollig, B., Gastin, P.: Weighted versus probabilistic logics. In: Diekert, V., Nowotka, D. (eds.) Developments in Language Theory, DLT 2009. LNCS, vol. 5583, pp. 18–38. Springer (2009)
6. von Braunmühl, B., Verbeek, R.: Input-driven languages are recognized in log n space. In: Proceedings of the Symposium on Fundamentals of Computation Theory. LNCS, vol. 158, pp. 40–51. Springer (1983)

7. Büchi, J.R.: Weak second-order arithmetic and finite automata. Z. Math. Logik und Grundlagen Math. 6, 66–92 (1960)
8. Choffrut, C., Malcher, A., Mereghetti, C., Palano, B.: First-order logics: some characterizations and closure properties. Acta Inf. 49(4), 225–248 (2012)
9. Crespi Reghizzi, S., Mandrioli, D.: Operator precedence and the visibly pushdown property. J. Comput. Syst. Sci. 78(6), 1837–1867 (2012)
10. Crespi-Reghizzi, S., Mandrioli, D., Martin, D.F.: Algebraic properties of operator precedence languages. Information and Control 37(2), 115–133 (1978)
11. Droste, M., Dück, S.: Weighted automata and logics for infinite nested words. Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.06.010
12. Droste, M., Gastin, P.: Weighted automata and weighted logics. Theor. Comput. Sci. 380(1-2), 69–86 (2007), extended abstract in ICALP 2005
13. Droste, M., Heusel, D., Vogler, H.: Weighted unranked tree automata over tree valuation monoids and their characterization by weighted logics. In: Maletti, A. (ed.) Conference Algebraic Informatics CAI 2015. LNCS, vol. 9270, pp. 90–102. Springer (2015)
14. Droste, M., Kuich, W., Vogler, H. (eds.): Handbook of Weighted Automata. EATCS Monographs in Theoretical Computer Science, Springer (2009)
15. Droste, M., Kuske, D.: Weighted automata. In: Pin, J.E. (ed.) Handbook: "Automata: from Mathematics to Applications". Europ. Mathematical Soc. (to appear)
16. Droste, M., Meinecke, I.: Weighted automata and weighted MSO logics for average and long-time behaviors. Inf. Comput. 220, 44–59 (2012)
17. Droste, M., Perevoshchikov, V.: A Nivat theorem for weighted timed automata and weighted relative distance logic. In: International Colloquium on Automata, Languages, and Programming, ICALP 2014, Part II. LNCS, vol. 8573, pp. 171–182. Springer (2014)
18. Droste, M., Pibaljommee, B.: Weighted nested word automata and logics over strong bimonoids. Int. J. Found. Comput. Sci. 25(5), 641–666 (2014)
19. Droste, M., Vogler, H.: Weighted tree automata and weighted logics. Theor. Comput. Sci. 366(3), 228–247 (2006)
20. Droste, M., Vogler, H.: Weighted automata and multi-valued logics over arbitrary bounded lattices. Theor. Comput. Sci. 418, 14–36 (2012)
21. Eilenberg, S.: Automata, Languages, and Machines, Pure and Applied Mathematics, vol. 59-A. Academic Press (1974)
22. Elgot, C.C.: Decision problems of finite automata design and related arithmetics. Trans. Am. Math. Soc. 98(1), 21–52 (1961)
23. Emerson, E.A.: Temporal and modal logic. In: Handbook of Theoretical Computer Science, Volume B, pp. 995–1072. MIT Press (1990)
24. Floyd, R.W.: Syntactic analysis and operator precedence. J. ACM 10(3), 316–333 (1963)
25. Knuth, D.E.: Semantics of context-free languages. Mathematical Systems Theory 2(2), 127–145 (1968)
26. Kuich, W., Salomaa, A.: Semirings, Automata, Languages, EATCS Monographs in Theoretical Computer Science, vol. 6. Springer (1986)
27. Lautemann, C., Schwentick, T., Thérien, D.: Logics for context-free languages. In: Pacholski, L., Tiuryn, J. (eds.) Computer Science Logic, Selected Papers. LNCS, vol. 933, pp. 205–216. Springer (1994)
28. Lonati, V., Mandrioli, D., Panella, F., Pradella, M.: Operator precedence languages: Their automata-theoretic and logic characterization. SIAM J. Comput. 44(4), 1026–1088 (2015)

29. Mathissen, C.: Weighted logics for nested words and algebraic formal power series. Logical Methods in Computer Science 6(1) (2010), selected papers of ICALP 2008
30. McNaughton, R.: Parenthesis grammars. J. ACM 14(3), 490–500 (1967)
31. McNaughton, R., Papert, S.: Counter-free Automata. MIT Press, Cambridge, USA (1971)
32. Mehlhorn, K.: Pebbling mountain ranges and its application of DCFL-recognition. In: Automata, Languages and Programming, ICALP 1980. LNCS, vol. 85, pp. 422–435 (1980)
33. Nivat, M.: Transductions des langages de Chomsky. Ann. de l'Inst. Fourier 18, 339–455 (1968)
34. Salomaa, A., Soittola, M.: Automata-Theoretic Aspects of Formal Power Series. Texts and Monographs in Computer Science, Springer (1978)
35. Schützenberger, M.P.: On the definition of a family of automata. Inf. Control 4(2-3), 245–270 (1961)
36. Thatcher, J.: Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. Journ. of Comp. and Syst.Sc. 1, 317–322 (1967)
37. Trakhtenbrot, B.A.: Finite automata and logic of monadic predicates (in Russian). Doklady Akademii Nauk SSR 140, 326–329 (1961)