Higher-order Operator Precedence Languages

Stefano Crespi Reghizzi
DEIB, Politecnico di Milano, and
IEIIT, Consiglio Nazionale delle Ricerche
via Ponzio 34/5, 20134 Milano, Italy
stefano.crespireghizzi@polimi.it

Matteo Pradella
DEIB, Politecnico di Milano, and
IEIIT, Consiglio Nazionale delle Ricerche
via Ponzio 34/5, 20134 Milano, Italy
matteo.pradella@polimi.it

Floyd's Operator Precedence (OP) languages are a deterministic context-free family having many desirable properties. They are locally and parallely parsable, and languages having a compatible structure are closed under Boolean operations, concatenation and star; they properly include the family of Visibly Pushdown (or Input Driven) languages. OP languages are based on three relations between any two consecutive terminal symbols, which assign syntax structure to words. We extend such relations to k-tuples of consecutive terminal symbols, by using the model of strictly locally testable regular languages of order $k \geq 3$. The new corresponding class of *Higher-order Operator Precedence languages* (HOP) properly includes the OP languages, and it is still included in the deterministic (also in reverse) context free family. We prove Boolean closure for each subfamily of structurally compatible HOP languages. In each subfamily, the top language is called *max-language*. We show that such languages are defined by a simple cancellation rule and we prove several properties, in particular that max-languages make an infinite hierarchy ordered by parameter k. HOP languages are a candidate for replacing OP languages in the various applications where they have have been successful though sometimes too restrictive.

Keywords Operator Precedence Languages, Input-Driven Languages, Deterministic Context-Free Languages, Syntactic Tags, Boolean Closure, Locally Testable Languages, Local Parsability, Grammar Inference.

1 Introduction

We propose a new way of extending the classic language family of *operator-precedence* (OP) languages, invented by R. Floyd [11] to design a very efficient parsing algorithm, still used within compilers. It is worth outlining the main characteristics of OP languages. OP languages have been also exploited for grammar inference [2], thanks to their lattice-theoretical properties. They offer promise for model-checking of infinite-state systems due to the Boolean closure, ω-languages, logic and automata characterizations, and the ensuing decidability of relevant problems [17]. Recently, a generator of fast parallel parsers has been made available [3]. Their bottom-up deterministic parser localizes the edges of the handle (a factor to be reduced by a grammar rule) by means of three precedence relations, represented by the *tags* $\lessdot, \gtrdot, \doteq$. (Since our model generalizes OP, we represent the tags as $[,], \odot$.) Such relations are defined between two consecutive terminals (possibly separated by a nonterminal). E.g., the *yield precedence* relation $a \lessdot b$ says that b is the leftmost terminal of the handle and a is the last terminal of the left context. The no-conflict condition of OP grammars ensures that the edge positions are unambiguous and the handles can be localized by means of a local test. An OP parser configuration is essentially a word consisting of alternated terminals and tags, i.e., a *tagged word*; notice that nonterminal symbols, although present in the configuration, play no role in determining the handle positions, but are of course necessary for checking syntactic correctness. In general, any language having the property that handles can be localized by a local test is called *locally parsable* and its parser is amenable to parallelization.

Submitted to:
AFL 2017

If the parser is abstracted as a pushdown automaton, each pair of terminals associated to a left or to a right edge of a handle, respectively triggers a push or a pop move; i.e., the move choice is driven by two consecutive input symbols. Therefore, the well-known model of *input-driven* [21, 4] (or "visibly pushdown" [1]) languages is a special case of the OP model, since just one terminal suffices to choose the move. This is shown in [8], where the precedence relations characterizing the input-driven languages are computed. The syntax structures permitted by such relations are sometimes too restrictive for the constructs of modern languages, e.g., a markup language like HTML5 has special rules that allow dropping some closing tags.

Since OP grammars, although used by compilers, are sometimes inconvenient or inadequate for specifying some syntactic constructs, a natural question is: can we increase the generative capacity of OP grammars, without jeopardizing their nice properties, by allowing the parser to examine more than two consecutive terminals to determine the handle position? Quite surprisingly, to our knowledge the question remained unanswered until now, but in the last section we mention some related research.

We intuitively present the main ideas of the new hierarchical family of languages and grammars called *Higher-order Operator Precedence* (HOP). Let $k \geq 3$ be and odd integer specifying the number of consecutive terminals and intervening tags to be used for localizing handles: the value of k is 3 for OP, which thus coincide with the HOP(3) subfamily. The main contributions are: a precise definition of HOP(k) grammars, a decidable condition for testing whether a grammar has the HOP(k) property, the proof that the OP family is properly included into the HOP one, and an initial set of nice properties that carry over from OP to HOP. The Boolean closure of each structurally compatible (this concept cannot be defined at this point but is standard for OP and input-driven languages) HOP subfamily is determinant for model checking. Concerning local parsability, we mention in the conclusions how it should be obtained. Last but not least, our definition of HOP grammars permits to use regular expressions in the right part of rules, in contrast with the classical definition of OP grammars.

Moreover, we prove that each structurally compatible HOP subfamily has a maximal element, called max-language. Interestingly, max-languages can be defined by a simple cancellation rule that applies to tagged words, and iteratively deletes innermost handles by a process called a *reduction*. Before each cancellation, the word, completed with tags, has to pass local tests, defined by means of a *strictly locally testable* [20] regular language of order k. We prove several properties of the max-language family, in particular that they form a strict infinite hierarchy ordered by parameter k. Since the model based on cancellation is simpler, it will be the first presented in this paper, before the HOP grammar model.

Paper organization: Section 2 contains the basic notation and definitions. Section 3 introduces the max-languages and their basic properties. Section 4 defines the HOP grammars and proves their properties. Section 5 compares HOP with some related existing models, and lists open problems and future research directions.

2 Basic definitions

For terms not defined here, we refer to any textbook on formal languages, e.g. [14]. For a generic alphabet we use the symbol Υ. The empty word is denoted by ε. Unless stated otherwise, all languages considered are free from the empty word. For any $k \geq 1$, for a word w, $|w| \geq k$, let $i_k(w)$ and $t_k(w)$ be the prefix and, respectively, the suffix of w of length k. If a word w has length at least k, $f_k(w)$ denotes the set of factors of w of length k, otherwise the empty set. Obviously, $i_k(w)$, $t_k(w)$ and f_k can be extended to languages. The i-th character of w is denoted by $w(i), 1 \leq i \leq |w|$.

A (nondeterministic) *finite automaton* (FA) is denoted by $M = (\Upsilon, Q, \delta, I, T)$, where $I, T \subseteq Q$ are respectively the initial and final states and δ is a relation (or its graph) over $Q \times \Upsilon \times Q$. A (labeled) *path* is a sequence $q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} q_n$, such that, for each $1 \leq i < n$, $(q_i, a, q_{i+1}) \in \delta$. The *path label* is $a_1 a_2 \ldots a_{n-1}$, the *path states* are the sequence $q_1 q_2 \ldots q_n$. An FA is *unambiguous* if each sentence in $L(M)$ is recognized by just one computation.

An *extended context-free* (ECF) grammar is a 4-tuple $G = (V_N, \Upsilon, P, S)$, where Υ is the terminal alphabet, V_N is the nonterminal alphabet, P is the set of rules, and $S \subseteq V_N$ is the set of axioms. Each rule has the form $X \to R_X$, where $X \in V_N$ and R_X is a *regular* language over the alphabet $V = \Upsilon \cup V_N$. R_X will be defined by means of an unambiguous FA, $M_X = (V, Q_X, \delta_X, I_X, T_X)$. We safely assume that for each nonterminal X there is exactly one rule, to be written as $X \to M_X$ or $X \to R_X$. A rule $X \to R_X$ is a *copy rule* if $\exists Y \in V_N : Y \in R_X$; we assume that there are no copy rules. A *context-free* (CF) grammar is an ECF grammar such that for each rule $X \to R_X$, R_X is a finite language over V.
The *derivation relation* $\Rightarrow \subseteq V^* \times V^*$ is defined as follows for an ECF grammar: $u \Rightarrow v$ if $u = u'Xu''$, $v = u'wu''$, $X \to R_X \in P$, and $w \in R_X$.
A word is *X-grammatical* if it derives from a nonterminal symbol X. If X is an axiom, the word is *sentential*. The language generated by G starting from a nonterminal X is denoted by $L(G, X) \subseteq \Upsilon^+$ and $L(G) = \bigcup_{X \in S} L(G, X)$.
The usual assumption that all parts of a CF grammar are productive can be reformulated for ECF grammars by combining reduction (as in a CF grammar) and trimming of the M_X FA for each rule $X \to M_X$, but we omit details for brevity.
An ECF grammar is in *operator* (normal) *form* if for all rules $X \to R_X$ and for each $x \in R_X$, $f_2(x) \cap V_N V_N = \emptyset$, i.e. it is impossible to find two adjacent nonterminals. Throughout the paper we only consider ECF grammars in operator form.

Let $G = (V_N, \Upsilon, P, S)$ and assume that $\{(,)\} \cap \Upsilon = \emptyset$. The *parenthesis grammar* $G_{()}$ is defined by the 4-tuple $(V_N, \Upsilon \cup \{(,)\}, P', S)$ where $P' = \{X \to (R_X) \mid X \to R_X \in P\}$. Let σ' be the homomorphism which erases parentheses, a grammar G is *structurally ambiguous* if there exist $w, z \in L(G_{()}), w \neq z$, such that $\sigma'(w) = \sigma'(z)$. Two grammars G' and G'' are *structurally equivalent* if $L(G'_{()}) = L(G''_{()})$.

Strict local testability and tagged languages Words of length k are called *k-words*. The following definition, equivalent to the classical ones (e.g., in [20],[5]), assumes that any input word $x \in \Upsilon^+$ is enclosed between two special words of sufficient length, called *end-words* and denoted by $\circled{\#}$. Let # be a character, tacitly assumed to be in Υ and used only in the end-words. We actually use two different end-words, without or with tags, depending on the context: $\circled{\#} \in \#^+$ (e.g. in Definition 2.1) or $\circled{\#} \in (\# \odot)^* \#$, (e.g. in Definition 2.2).

Definition 2.1. Let $k \geq 2$ be an integer, called *width*. A language L is *k-strictly locally testable*, if there exists a k-word set $F_k \subseteq \Upsilon^k$ such that $L = \{x \in \Upsilon^* \mid f_k(\circled{\#} x \circled{\#}) \subseteq F_k\}$; then we write $L = \mathrm{SLT}(F_k)$. A language is *strictly locally testable* (*SLT*) if it is k-strictly locally testable for some k. $\square$

We assume that the three characters, called *tags*, $[,]$, and $\odot$ are distinct from terminals and nonterminals characters and we denote them as $\Delta = \{[,], \odot\}$. For any alphabet, the projection σ erases all the tags, i.e. $\sigma(x) = \varepsilon$, if $x \in \Delta$, otherwise $\sigma(x) = x$. Here we apply the SLT definition to words that contain tags and are the base of our models. Let Σ be the terminal alphabet. A *tagged word* starts with a terminal and alternates tags and terminals.

Definition 2.2 (tagged word and tagged language)**.** Let here and throughout the paper $k \geq 3$ be an odd integer. A *tagged word* is a word w in the set $\Sigma(\Delta\Sigma)^*$, denoted by $\Sigma^{\square}$. A *tagged sub-word* of w is a factor of w that is a tagged word. A *tagged language* is a set of tagged words. Let $\Sigma^{\square k} = \{w \in \Sigma^{\square} \mid |w| = k\}$. We call *tagged k-word* any word in $\Sigma^{\square k}$. The set of all tagged k-words that occur in w is denoted by $\varphi_k(w)$.

A language $L \subseteq \Sigma^{\square}$ is a *k-strictly locally testable tagged language* if there exists a set of tagged k-words $\Phi_k \subseteq \Sigma^{\square k}$ such that $L = \left\{ w \in \Sigma^{\square} \mid \varphi_k (\circledast [\, w \,] \circledast) \subseteq \Phi_k \right\}$. In that case we write $L = \mathrm{SLT}(\Phi_k)$. The k-word set $F_k \subseteq (\Sigma \cup \Delta)^k$ *derived from* Φ_k is $F_k = \bigcup_{x \in \mathrm{SLT}(\Phi_k)} f_k(x)$.

A tagged k-word set Φ_k is *conflictual* if, and only if, $\exists x, y \in \Phi_k, x \neq y$, such that $\sigma(x) = \sigma(y)$. □

E.g., $\mathrm{SLT}(\{\#[a,\ a \odot b,\ b \odot a,\ a]\#\}) = (a \odot b \odot)^* a$.

We observe that, for each word $w \in \Sigma^{\square}$, the set $\varphi_k(w)$ is included in $f_k(w)$. E.g., from $\Phi_3 = \{\#[a,\ a \odot b,\ b \odot a,\ a]\#\}$ we derive the 3-word set $F_3 = \Phi_3 \cup \{[a\odot,\ [a],\ \odot b \odot,\ \odot a \odot,\ \odot a]\}$. Yet, although $\Phi_k \subset F_k$, the languages defined by strict local testing obviously coincide: $\mathrm{SLT}(F_k) = \mathrm{SLT}(\Phi_k)$.

In what follows all tagged word sets considered are not conflictual, unless stated otherwise. An important remark is that for every word w over Σ, $\sigma^{-1}(w) \cap \mathrm{SLT}(\Phi_k)$ is either empty or a singleton: the tagged word corresponding to w.
The following technical lemma is useful for later proofs.

Lemma 2.3. *Let $w \in \Sigma^{\square k}$; let $s', s'' \in \Delta$ be two distinct tags. Then, for every $3 \leq h \leq k+2$, the tagged word set $\varphi_h(ws'ws''w)$ is conflictual.*

Proof. Let $w = a_1 s_2 a_3 \ldots s_{k-1} a_k$. It suffices to observe that the conflicting tagged h-words $t_h(a_1 s_2 a_3 \ldots s_{k-1} a_k s' a_1)$ and $t_h(a_1 s_2 a_3 \ldots s_{k-1} a_k s'' a_1)$ are contained in $\varphi_h(ws'ws''w)$. □

An immediate corollary: when $w = a \in \Sigma$, for any sufficiently long word $z \in a(\Delta a)^*$, if z contains two distinct tags, the set $\varphi_k(z)$ is conflictual.

3 Reductions and Maximal languages

We show that the SLT tagged words, defined by a set Φ of (non-conflictual) k-words, can be interpreted as defining another language over the terminal alphabet; the language is context-free but not necessarily regular, and is called *maximal language or max-language*. We anticipate from Section 4 the reason of the name "max-language": such languages belong to the family of Higher-order Operator Precedence languages (Definition 4.3), and they include any other HOP language that is structurally compatible. We first define the reduction process, then we prove some properties of the language family.

Consider a set $\Phi_k \subseteq \Sigma^{\square k}$ and a word w over Σ; let $x \in \mathrm{SLT}(\Phi)$ be the tagged word corresponding to w, if it exists. Word w belongs to the max-language if x reduces to a specified word by the repeated application of a reduction operation. A reduction cancels from the current x a sub-word of a special form called a handle, replaces it with a tag, and thus produces a new tagged word. All the tagged words thus obtained by successive reductions must belong to $\mathrm{SLT}(\Phi)$.

Definition 3.1 (maximal language). Let $\Phi \subseteq \Sigma^{\square k}$. A *handle* is a word of the form $[x]$ where $x \in (\Sigma - \{\#\}) \cdot (\odot (\Sigma - \{\#\}))^*$, i.e., a handle is a tagged word enclosed between the tags [and], and not containing symbols in $\{[,], \#\}$.
A *reduction* is a binary relation $\rightsquigarrow_\Phi \subseteq (\Sigma \cup \Delta)^+ \times (\Sigma \cup \Delta)^+$ between tagged words, defined as:

$$w[u]z \rightsquigarrow_\Phi wsz \text{ if, and only if, } w[u]z \in \mathrm{SLT}(\Phi) \text{ where } [u] \text{ is a handle, and } \exists s \in \Delta : wsz \in \mathrm{SLT}(\Phi). \quad (1)$$

The handle $[u]$ is called *reducible*. A reduction is called *leftmost* if no handle occurs in w. The definition of *rightmost* reduction is similar. Observe that at most one tag s may satisfy Condition (1) since Φ is non-conflictual. The subscript Φ may be dropped from $\rightsquigarrow_\Phi$ when clear from context; $\overset{*}{\rightsquigarrow}$ is the reflexive and transitive closure of $\rightsquigarrow$.

The *tagged maximal language* defined by Φ via *reduction* is $\overline{\mathrm{Red}}(\Phi) = \{w \in \Sigma^{\Box} \mid \textcircled{\#}[\,w\,]\textcircled{\#} \overset{*}{\rightsquigarrow}_{\Phi} \textcircled{\#}\odot\textcircled{\#}\}$.
The *maximal language* defined by Φ, is $\mathrm{Red}(\Phi) = \sigma\left(\overline{\mathrm{Red}}(\Phi)\right)$.
We say that languages $\mathrm{Red}(\Phi)$ and $\overline{\mathrm{Red}}(\Phi)$ are in the families $\mathrm{Red}(k)$ and $\overline{\mathrm{Red}}(k)$ respectively; a language is in the Red family if it is in $\mathrm{Red}(k)$ for some k. $\Box$

Notice that in Definition 3.1 the reductions may be applied in any order, without affecting $\overline{\mathrm{Red}}(\Phi)$.

Example 3.2. This and the following examples were checked by a program. The Dyck language (without ε) over the alphabet $\{a,a',b,b'\}$, which can be easily extended to an arbitrary number of matching pairs, is a Red(3) language defined by the tagged word set $\Phi = \{\#\odot\#,\ b']a',\ a']b',\ \#[b,\ b'[b,\ b[b,\ a']a',\ \#[a,\ b']\#,\ a\odot a',\ a[b,\ b[a,\ b'[a,\ a'[b,\ b']b',\ a']\#,\ b\odot b',\ a'[a,\ a[a\}$. Word $aaa'a'aa'$ is recognized by the following reductions, respectively leftmost and rightmost:

$$\begin{array}{l|l}
\textcircled{\#}[a[a\odot a']a'[a\odot a']\textcircled{\#} \rightsquigarrow \textcircled{\#}[a\odot a'[a\odot a']\textcircled{\#} & \textcircled{\#}[a[a\odot a']a'[a\odot a']\textcircled{\#} \rightsquigarrow \textcircled{\#}[a[a\odot a']a']\textcircled{\#} \\
\rightsquigarrow \textcircled{\#}[a\odot a']\textcircled{\#} \rightsquigarrow \textcircled{\#}\odot\textcircled{\#} & \rightsquigarrow \textcircled{\#}[a\odot a']\textcircled{\#} \rightsquigarrow \textcircled{\#}\odot\textcircled{\#}
\end{array}$$

Some elementary properties of max-languages come next.

Lemma 3.3. *1.* $\nexists x,y \in \overline{\mathrm{Red}}(\Phi)$*,* $x \neq y$*:* $\sigma(x) = \sigma(y)$*. (Unambiguity)*

2. $\forall x,y$*, if* $x \in \overline{\mathrm{Red}}(\Phi)$ *and* $\textcircled{\#}[\,x\,]\textcircled{\#} \overset{*}{\rightsquigarrow}_{\Phi} \textcircled{\#}[\,y\,]\textcircled{\#}$*, then* $y \in \overline{\mathrm{Red}}(\Phi)$*. (Closure under* $\rightsquigarrow$*)*

3. Let $F_h = \sigma(\Phi)$ *(hence* $h = \lceil k/2 \rceil$*). Then* $\mathrm{SLT}(F_h) \supseteq \mathrm{Red}(\Phi)$*. (Refinement over SLT)*

Proof. Stat. 1. and 2. follow from Definition 3.1. Define the tagged k-words set $\hat{\Phi} = \sigma^{-1}(F_h) \cap \Sigma^{\Box k}$, which clearly includes Φ. From the identity $\mathrm{SLT}(F_h) = \sigma(\mathrm{SLT}(\hat{\Phi}))$, Stat. 3 follows. $\Box$

Example 3.4. This is a running example. $L = \{a^n(cb^+)^n \mid n > 0\}$ is a Red(3) language specified by $\Phi = \{\#\odot\#,\ \#[a,\ b]\#,\ b]c,\ c\odot b,\ b\odot b,\ a\odot c,\ a[a\}$. The reduction steps for word $aaacbbbbcbcbbb$ are:
$\textcircled{\#}[a[a[a\odot c\odot b\odot b\odot b\odot b]c\odot b]c\odot b\odot b\odot b]\textcircled{\#} \rightsquigarrow$
$\textcircled{\#}[a[a\odot c\odot b]c\odot b\odot b\odot b]\textcircled{\#} \rightsquigarrow \textcircled{\#}[a\odot c\odot b\odot b\odot b]\textcircled{\#} \rightsquigarrow \textcircled{\#}\odot\textcircled{\#}$
Therefore $[a[a[a\odot c\odot b\odot b\odot b\odot b]c\odot b]c\odot b\odot b\odot b] \in \overline{\mathrm{Red}}(\Phi)$ and $aaacbbbbcbcbbb \in \mathrm{Red}(\Phi)$. On the other hand, word $aacbb$ is not accepted because it is not the content of a tagged word that reduces to $\textcircled{\#}\odot\textcircled{\#}$, in particular, the reduction of handle $[a\odot c\odot b\odot b]$ in $\textcircled{\#}[a[a\odot c\odot b\odot b]\textcircled{\#}$ is not possible because there is not a tag s such that $as\# \in \mathrm{SLT}(\Phi)$.

First, we compare Red with REG, the family of *regular languages*.

Theorem 3.5. *The family of* Red *languages strictly includes the SLT family and is incomparable with the REG family.*

Proof. Inclusion SLT $\subseteq$ Red: The mapping $\widetilde{(\cdot)} : \Sigma^+ \to \Sigma^{\Box}$ is defined by $\widetilde{z} = z(1)\odot z(2)\odot\cdots\odot z(|z|)$, for any $z \in \Sigma^+$. Given a set F_j, $j \geq 2$ defining an SLT language over Σ, we define the set Φ_k, $k = 2j-1$: it contains, for every $u \in F_j \cap (\Sigma - \{\#\})^+$, the tagged word $\widetilde{u}$, and, for every $w = \#^{j_1}v\#^{j_2} \in F_j$, $j_1 + |v| + j_2 = j$, the tagged word $\widetilde{w} = (\#\odot)^{j_1-1}\#[v(1)\odot v(2)\odot\cdots\odot v(|v|)]\#(\odot\#)^{j_2-1}$.
We prove that $\mathrm{SLT}(F_j) \subseteq \mathrm{Red}(\Phi_k)$. Consider any $z \in \mathrm{SLT}(F_j)$, for simplicity assume $|z| \geq j$. Then $\widetilde{\#^{j-1}z\#^{j-1}} = \textcircled{\#}[z(1)\odot\cdots\odot z(|z|)]\textcircled{\#} \rightsquigarrow_{\Phi_k} \textcircled{\#}\odot\textcircled{\#}$. Since the converse inclusion $\mathrm{SLT}(F_j) \supseteq \mathrm{Red}(\Phi_k)$ is obvious, it follows that SLT $\subseteq$ Red.
The inclusion SLT $\subset$ Red is proved by using $L = a^*ba^* \cup a^+$ which is defined by $\Phi_3 = \{\#[b,a]b, \#\odot\#, \#[a,b]\#, a\odot a, b[a,a]\#\}$. But it is known that L is not locally testable.
The inclusion Red $\not\subseteq$ REG is proved by the Dyck languages. To prove REG $\not\subseteq$ Red, we consider $R = (aa)^+$. By Lemma 2.3, a Φ_k for R may only use one tag, and we first consider the case [(the case with] is

analogous). For any odd value $k \geq 3$, Φ_k has the form $\{\textcircled{\#}[a, \dots \#[a[\dots[a,a[\dots[a,\dots a[\dots[a]\textcircled{\#}, \dots\}$, therefore the handle is always $[a]$ and $\mathrm{Red}(\Phi_k)$ necessarily includes also words with an odd number of a's. The same conclusion holds in the $\odot$ case, since any handle has the form $[a \odot \dots \odot a]$. □

We prove that family Red is an infinite strict hierarchy.

Theorem 3.6. *For every $k \geq 5$, the language family* $\mathrm{Red}(k)$ *strictly includes* $\mathrm{Red}(k-2)$.

Proof. Consider the languages $L_{(h)} = (a^h b)^+$, $a \geq 1$. It is easy to verify that for $k = 2h+1$, $L_{(h)}$ is in $\mathrm{Red}(k)$. E.g., $L_{(2)} = \mathrm{Red}(\{\# \odot \#[a,\ b]\# \odot \#,\ b]a \odot a,\ a \odot b]a,\ a \odot a \odot b,\ \#[a \odot a,\ a \odot b]\#,\ \# \odot \# \odot \#\})$.

We prove that $L_{(h)}$ is not in $\mathrm{Red}(k-2)$. Assume by contradiction that $L_{(h)} = \mathrm{Red}(\Phi_{k-2})$, for some Φ_{k-2}, and consider $y \in L_{(h)}$ and $y' \in \overline{\mathrm{Red}}(\Phi_{k-2})$, such that $y = \sigma(y')$. The word y' contains a tagged sub-word $w = as_1 \dots as_{h-1}a$, $s_i \in \Delta$, and two cases are possible.

- $\exists 1 \leq i < j \leq h-1$ such that $s_i \neq s_j$. By Lemma 2.3, the set $\varphi_{k-2}(w)$ is conflictual.
- All s_i in w are identical. But this means that, if $a^h b \in \mathrm{Red}(\Phi_{k-2})$ (by hypothesis), then also $a^{h+1}b \in \mathrm{Red}(\Phi_{k-2})$, which is not in $L_{(h)}$. □

The Red family is not closed under the following operations, as proved by witnesses:

Intersection: $\{a^n b^n c^* \mid n > 0\} = \mathrm{Red}(\{b[c,\ c]\#,\ a \odot b,\ c \odot c,\ \# \odot \#,\ \#[a,\ b]\#,\ b]b,\ a[a\ \})$ and $\{a^* b^n c^n \mid n > 0\} = \mathrm{Red}(\{\#[b,\ a]b,\ c]\#,\ c]c,\ \# \odot \#,\ b[b,\ \#[a,\ a \odot a,\ b \odot c\})$, and their intersection is not context-free.

Set difference: $(a^* b a^* \cup a^+) - a^+$. The first language is in Red(3) (see the proof of Theorem 3.5) and requires $\textcircled{\#}[a$ and $a]\textcircled{\#}$ because words may begin and end with a. Since unlimited runs of a are possible, Lemma 2.3 imposes the same tag between any pair of a's, hence the resulting Red language necessarily contains a^+, a contradiction.

Concatenation: $a^* b = \mathrm{Red}(\{\#[b,\ a \odot b,\ \# \odot \#,\ \#[a,\ b]\#,\ a \odot a\})$ concatenated with a^+ is similar to the witness for set difference.

Intersection with regular set: $\{a,b\}^+ \cap (aa)^+$, see Theorem 3.5.

The language family of the next section contains Red and has better closure properties.

4 Generalization of operator-precedence languages

We introduce a new family of languages, called HOP(k), standing for *Higher-order Operator Precedence languages* of order $k \geq 3$. The HOP(k) condition is decidable for grammars; HOP(k) languages are deterministic, also in reverse. With respect to existing families, we start from the operator-precedence (OP) languages, defined by Floyd [11], and prove that they are the same as the new family HOP(3), when the grammar rules are in non-extended CF form. We prove the Boolean closure property for the family of HOP(k) languages having the same set Φ_k of tagged k-words. The top element in such family is the Red language (also known as max-language) defined by Φ_k.

Operator Precedence Grammars An Operator Precedence grammar[1] [11] is characterized by three OP relations over Σ^2, denoted by $\gtrdot$, $\doteq$, $\lessdot$, that are used to assign a structure to the words (see e.g. [13] for the classical parsing algorithm).

Example 4.1. Consider the following operator grammar (with rules specified for brevity by regular expressions) and its OP relations:

$$G_1 = \{S \to XbX \cup bX,\ X \to aa^*\} \qquad a \doteq a,\ a \gtrdot b,\ b \lessdot a.$$

[1]Floyd's definition uses CF grammars, but it is straightforward to extend it to ECF grammars.

By default, $\# \lessdot x$ and $x \gtrdot \#$, for any $x \in \Sigma$, and $\# \doteq \#$. A grammar is OP if at most one relation exists between any two terminal symbols. To parse word *aaaba*, the bottom-up OP parser executes the following reduction steps:

$$\# \lessdot a \doteq a \doteq a \gtrdot b \lessdot a \gtrdot \# \Longleftarrow_{G_1} \# \overset{X}{\lessdot} b \lessdot a \gtrdot \# \Longleftarrow_{G_1} \# \overset{X}{\lessdot} b \overset{X}{\gtrdot} \# \Longleftarrow_{G_1} \# \overset{S}{\doteq} \# \tag{2}$$

If we substitute the OP relation symbols with tags, the above OP relations are encoded by the tagged 3-words $\Phi_3 = \{a \odot a, a[b, b]a, \#[a, a]\#, \# \odot \#, \#[b, b]\#\}$. The OP property implies that Φ_3 is non-conflictual. Notice that each word in (2) (disregarding the #'s) belongs to $\overline{\text{Red}}(\Phi_3)$. Recalling Definition 3.1, we observe that $a \odot a \odot a]b[a \in \overline{\text{Red}}(\Phi_3)$, therefore $aaaba \in \text{Red}(\Phi_3)$. Moreover, $\text{Red}(\Phi_3) = a^*ba^* \cup a^+ \supset L(G_1)$.

Our generalization of OP grammars is based on the idea of using tagged k-words, with $k \geq 3$, for assigning a syntactic structure to words. The test for a grammar to be OP [11] is quite simple, but the wider contexts needed when $k > 3$, impose a more involved device for the no-conflict check. For that we define a grammar, called *tagged*, obtained from the original grammar by inserting tags into rules.

Definition 4.2 (Tagged grammar)**.** Let $G = (V_N, \Sigma, P, S)$ be a grammar. Define a language substitution $\rho : V \to \mathscr{P}\left(V \cup \Delta \cup (\Sigma \cup \Delta)^2\right)$ such that

$$\rho(a) = \{a,\ a\odot,\ a],\ [a\}, a \in \Sigma; \qquad \rho(X) = \{X\} \cup \Delta, X \in V_N.$$

Let R be the regular language defined by $R = (V_N \cup \{[\}) \cdot \Sigma \cdot ((V_N \cup \{\odot\}) \cdot \Sigma)^* \cdot (V_N \cup \{]\})$. We construct from G the *tagged grammar* associated to G, denoted by $\overline{G} = (V_N, \Sigma \cup \Delta, \overline{P}, S)$. For each rule $X \to R_X \in P$, $\overline{G}$ has the rule $X \to \overline{R}_X$ where $\overline{R}_X = \rho(R_X) \cap R$. □

The idea underlying $\overline{G}$'s definition is to insert tags into the rules of $\overline{P}$, so that G's structure becomes visible in the tagged words generated by $\overline{G}$. Tagged grammars are akin to the classical parenthesis grammars [19], yet their representation of nested structures is more parsimonious, since a single "[" tag (analogously a "]") can represent many open (resp. closed) parentheses. Notice that $\sigma(L(\overline{G})) \supseteq L(G)$, since tagged grammar rules exist, which replace a nonterminal with a tag. Such rules may generate words that, after deleting the tags, are not in $L(G)$. To illustrate, going back to G_1 of Example 4.1, grammar $\overline{G}_1$ has the rules $\{S \to (X \cup [)\, b\, (X \cup]), X \to [a\,(\odot a)^*]\}$ and generates the word $[b]$, while $\sigma([b]) \notin L(G_1)$. With the help of the tagged grammar $\overline{G}$, we can compute all the tagged k-words that may occur in parsing any valid word for grammar G (exemplified in (2)). Then, we can check whether they are conflictual or not. In the latter case, grammar G fulfills the next Definition 4.3 of HOP(k) grammar.
Returning to Example 4.1, the tagged 3-words $\varphi_3\left(\#L(\overline{G}_1)\#\right)$ coincide with the set Φ_3 encoding the precedence relations. As observed, since G_1 is an OP grammar, Φ_3 is nonconflictual, and G_1 is a HOP(3) grammar as well. The formalization follows.

Definition 4.3 (Higher-order Operator Precedence grammars)**.** Let $k \geq 3$ be an odd integer. A grammar G, having $\overline{G}$ as associated tagged grammar, is a *higher-order operator precedence grammar* of order k (in short HOP(k)) if

$$\nexists u, v \in \varphi_k\left(\textcircled{\#}L(\overline{G})\textcircled{\#}\right) \text{ such that } u \neq v \text{ and } \sigma(u) = \sigma(v) \tag{3}$$

This means that the set of all tagged k-words occurring in any sentence of $L(\overline{G})$ is nonconflictual. The union of the families HOP(k) for all values of k is denoted by HOP. The family of grammars HOP(k) having the same set Φ of tagged k-words is denoted by HOP(k, Φ). Identical notations denote the corresponding language families, when no confusion arises. □

The decidability of Condition (3) for a given grammar and a fixed value of k is obvious. With an abuse of terminology, we also say that $\varphi_k(L(\overline{G}))$ are the tagged k-words of grammar G.

Theorem 4.4. *The family OP of* operator precedence *languages coincides with the family HOP(3), and is properly included within the HOP family.*

Proof. The proof formalizes the already stated fact that OP relations are encoded by tagged 3-words. Let G be an ECF grammar. For all letters $a,b \in \Sigma$ we show that the following relations hold:

$$a \doteq b \iff a \odot b \in \varphi_3(L(\overline{G})), \quad a \lessdot b \iff a[b \in \varphi_3(L(\overline{G})), \quad a \gtrdot b \iff a]b \in \varphi_3(L(\overline{G})).$$

If $a \gtrdot b$, from the definition of OP grammar [11], it follows that there are a sentential word uXv with $i_1(v) = b$, and an X-grammatical word w such that $t_1(w) = a$ or $t_2(w) = aY$ with $Y \in V_N$. In both cases, for the tagged grammar $\overline{G}$ (see Definition 4.2), either the substitution $\rho(a) = a]$ or $\rho(Y) =]$ causes the 3-word $a]b$ to be in Φ_3, the tagged k-words of grammar G. (Notice that a wrong choice for the symbol returned by $\rho(Y)$, i.e. $[$ and $\odot$, is neutralized by the intersection with language R and does not show up in the tagged grammar.) Conversely, it is obvious that $a]b \in \Phi_3$ implies $a \gtrdot b$.
We omit the similar case $a \lessdot b$, and examine the case $a \doteq b$, which happens if there exists a rule containing in the right part as factor ab or aYb. The respective substitutions $\rho(a) = a\odot$ and $\rho(Y) = \odot$ produce the 3-word $a \odot b \in \Phi_3$. The converse is also immediate. It follows that, for every pair $a,b \in \Sigma$, grammar G violates the OP condition if, and only if, the HOP condition is false for $k = 3$.

On the other hand, we show a HOP language that is not an OP language. Let $L = \{a^n(baab)^n \mid n \geq 1\}$. For any grammar of L, by applying a pumping lemma, it is clear that the relations $a \lessdot a$ and $b \gtrdot b$ are unavoidable, i.e., the 3-words $a[a, b]b$ are necessarily present in Φ_3. But it can be checked that, no matter how we choose the other precedence relations, either there is a conflict or the language generated by the grammar fails to be L; this can be exhaustively proved by examining all possible non-conflictual choices of $\Phi_3 \subset \Sigma^{\square 3}$.
On the other hand, it is possible to check that the grammar $G_2 : S \to aSbaab \cup abaab$ is in HOP(7), and its tagged 7-words are

$$\Phi_7 = \left\{ \begin{array}{llllll} \#\odot\#[a\odot b, & a\odot a\odot b]\#, & \#[a[a\odot b, & a[a\odot b\odot a, & b]b\odot a\odot a, & a\odot b]b\odot a, \\ a\odot a\odot b]b, & a\odot b]\#\odot\#, & a[a[a[a, & b\odot a\odot a\odot b, & \#[a\odot b\odot a, & \#\odot\#\odot\#\odot\#, \\ a[a[a\odot b, & \#\odot\#\odot\#[a, & b]\#\odot\#\odot\#, & \#\odot\#[a[a, & \#[a[a[a, & a\odot b\odot a\odot a \end{array} \right\}.$$

□

It is known that OP grammars are structurally unambiguous, and that OP languages are CF deterministic and reverse-deterministic. Since such properties immediately follow from the bottom-up parser for OP grammars, which is easily extended to HOP(k) grammars without changing the essential operations, the same properties hold for any value of k.

Theorem 4.5. *Every HOP grammar is structurally unambiguous. The HOP language family is properly included within the deterministic and reverse-deterministic CF languages.*

Proof. We only need to prove the last statement. Let $L = \{a^nba^n \mid n \geq 1\}$, which is deterministic and reverse deterministic. For any grammar G of L, a straightforward application of the pumping lemma shows that, for any $k \geq 3$, $\varphi_k(L(\overline{G}))$ includes a word $as_1as_2\ldots a$ containing two distinct tags "[" and "]", therefore it also includes two conflictual k-words, because of the remark following Lemma 2.3. □

We show the significant connection between the HOP languages and the Red languages, which motivates their appellation of max-languages.

Max-grammars We prove by a fairly articulate construction that if L is a max-language, i.e. $L = \mathrm{Red}(\Phi_k)$, for some $\Phi_k \subseteq \Sigma^{\Box k}$, then L is generated by a grammar $G \in \mathrm{HOP}(k, \Phi_k)$. Moreover, L is the largest language in $\mathrm{HOP}(k, \Phi_k)$.

Preliminarily, we define, for an arbitrary SLT language over a generic alphabet Υ, a nondeterministic FA which is symmetrical w.r.t. the scanning direction; this property contrasts with the standard deterministic sliding-window device, e.g., in [5].

Definition 4.6 (symmetrical FA)**.** Let F_k be a k-word set. The *k-symmetrical automaton A* associated to F_k is obtained by trimming the FA $A_0 = (\Upsilon, Q, \delta, I, T)$ where:

- $Q = \{(\beta, \alpha) \in \Upsilon^{k-1} \times \Upsilon^{k-1} \mid \beta, \alpha \in f_{k-1}(F_k)\}$
- $(\beta, \alpha) \xrightarrow{a} (\beta', \alpha') \in \delta$ if, and only if, $\beta' = t_{k-1}(\beta a) \wedge \alpha = i_{k-1}(a \alpha')$
- $I = \{(t_{k-1}(\text{\#}), \alpha) \in Q\}$, $T = \{(\beta, i_{k-1}(\text{\#})) \in Q\}$. $\Box$

Intuitively, β and α represent the look-back and look-ahead $(k-1)$-words of state (β, α). See Figure 1 for illustration. Two relevant properties of the *symmetrical FA A* are:

1. $L(A) = \mathrm{SLT}(F_k)$, since, on each accepting path, the k-factors are by construction those of F_k.
2. The automaton A is unambiguous. Consider a word $x = uyv$, and assume by contradiction that there are two accepting paths in A, with state sequences $\pi_u \pi_y \pi_v$, $\pi_u \pi'_y \pi_v$ and the same label x (u and v could be ε). But, by construction of A, if $\pi_y = q_1 q_2 \ldots q_t$ and $\pi'_y = q'_1 q'_2 \ldots q'_t$, then $q_1 = (t_{k-1}(u), i_{k-1}(y))$ and also $q'_1 = (t_{k-1}(u), i_{k-1}(y))$. This holds for every subsequent step, hence $\pi'_y = \pi_y$, so the two paths must be identical.

Given $\Phi \subseteq \Sigma^{\Box k}$, we construct a grammar, denoted by $\overline{G}_\Phi$, that generates the max-language $\overline{\mathrm{Red}}(\Phi)$. The construction is also illustrated in Example 4.8.

Definition 4.7 (max-grammar construction)**.** Let $\Phi \subseteq \Sigma^{\Box k}$, and let $A_\Phi = (\Sigma \cup \Delta, Q, \delta, I, T)$ be the symmetrical automaton recognizing $\mathrm{SLT}(\Phi)$. The grammar $\overline{G}_\Phi = (\Sigma \cup \Delta, V_N, P, S)$ called *tagged max-grammar*, is obtained by reducing (in the sense of trimming the useless parts) the grammar constructed as follows.

- V_N is a subset of $Q \times Q$ such that $(q_1, q_2) \in V_N$ if $q_1 = (\beta_1, [\gamma_1)$ and $q_2 = (\gamma_2], \alpha_2)$, for some $\beta_1, \gamma_1, \gamma_2, \alpha_2$. Thus a nonterminal X is also identified by $(\beta_1, [\gamma_1, \gamma_2], \alpha_2)$.
- The axiom set is $S = I \times T$.
- Each rule $X \to R_X \in P$ is such that the right part is defined by the FA $M_X = (V \cup \Delta, Q_X, \delta_X, \{p_I\}, \{p_T\})$ where $Q_X \subseteq Q$, next specified.

  Let $X = (\beta_X, \alpha_X, \beta'_X, \alpha'_X)$. Then: $p_I = (\beta_X, \alpha_X)$, $p_T = (\beta'_X, \alpha'_X)$,
- The graph of the transition relation is $\delta_X = (\delta \cup \delta') - \delta''$, where

$$\delta' = \left\{ (\beta_1, \alpha_3) \xrightarrow{(\beta_1, \alpha_1, \beta_2, \alpha_2)} (\beta_3, \alpha_2) \;\middle|\; \begin{array}{l} (\beta_1, \alpha_1, \beta_2, \alpha_2) \in V_N, \\ (\beta_1, \alpha_3) \xrightarrow{\odot} (\beta_3, \alpha_2) \in \delta \vee \\ (\beta_1, \alpha_3) = p_I \wedge (\beta_1, \alpha_3) \xrightarrow{[} (\beta_3, \alpha_2) \in \delta \vee \\ (\beta_3, \alpha_2) = p_T \wedge (\beta_1, \alpha_3) \xrightarrow{]} (\beta_3, \alpha_2) \in \delta \end{array} \right\}$$

$$\delta'' = \begin{array}{l} \left\{ q' \xrightarrow{[} q'' \in \delta \;\middle|\; q' \neq p_I \right\} \cup \left\{ q' \xrightarrow{]} q'' \in \delta \;\middle|\; q'' \neq p_T \right\} \cup \\ \left\{ q' \xrightarrow{x} p_I \in \delta \right\} \cup \left\{ p_T \xrightarrow{x} q' \in \delta \right\}. \end{array}$$

Intuitively, δ' adds transitions with nonterminal labels between any two states already linked by a tag-labeled transition, which are "compatible" with the nonterminal name (i.e. with the same look-back and look-ahead). The transitions δ'' to be deleted are: those labeled by tags "[" or "]" that are not initial or final, and those reentering the initial or final states.

Define the *max-grammar* as $G_\Phi = (V_N, \Sigma, \{X \to \sigma(R_X) \mid X \to R_X \in P\}, S)$.

The *grammar graph* $\Gamma(\overline{G}_\Phi)$ of $\overline{G}_\Phi$ is a graph containing all the arcs and states of the symmetrical automaton A_Φ associated to Φ, together with all the arcs labelled by nonterminals, defined by the above construction of δ'. $\square$

The grammar graph synthetically represents all the rules of the max-grammar $\overline{G}_\Phi$ and will be used in the proof of the forthcoming lemma. Each rule right part is a subgraph starting with a label "[", ending with a label "]", and containing only terminals and $\odot$ tags; the rule left part is denoted by the pair of initial and final states of the subgraph.

Example 4.8. We show the construction of the max-grammar for the tagged 3-word set $\Phi = \{\# \odot \#,\ \#[a,\ b]\#,\ b]c,\ c \odot b,\ b \odot b,\ a \odot c,\ a[a\}$ of Example 3.4. Its symmetrical automaton A_Φ is reported in Figure 1 (i). The nonterminals are included in the set $\{(\odot\#, [a), ([a, [a)\} \times \{(b], c\odot), (b], \#\odot)\}$, but $(\odot\#, [a,b], c\odot)$ and $([a, [a,b], \#\odot)$ are unreachable, because they are neither axioms nor they are transition labels in the grammar graph. Thus only two nonterminals are left: the axiom $X = (\odot\#, [a,b], \#\odot)$ and $Y = ([a, [a,b], c\odot)$ which occurs on the transition from $([a, \odot c)$ to $(a\odot, c\odot)$. The two rules of the resulting grammar are $X \to [a(\odot \cup Y)c \odot (b\odot)^* b]$ and $Y \to [a(\odot \cup Y)c \odot (b\odot)^* b]$ and their automata are show in Figure 1 (ii) and (iii), respectively.

By construction, the rules of any tagged max-grammar $\overline{G}_\Phi$ have some properties worth noting:

1. For each rule $X \to M_X$, $R_X \subseteq (V_N \cup \{[\}) \cdot \Sigma \cdot ((V_N \cup \{\odot\}) \cdot \Sigma)^* \cdot (V_N \cup \{]\})$. This fact implies that $\overline{G}_\Phi$ and G_Φ are in operator form.
2. For each rule $X \to M_X$ in P, M_X is an unambiguous FA.

We prove that the languages defined by max-grammars and by reductions of Definition 3.1 coincide.

Lemma 4.9. *Let $\Phi \subseteq \Sigma^{\square k}$, and let $\overline{G}_\Phi$ and G_Φ be the max-grammars of Definition 4.7. Then $L(\overline{G}_\Phi) = \overline{\mathrm{Red}}(\Phi)$ and $L(G_\Phi) = \mathrm{Red}(\Phi)$.*

Proof. It suffices to consider $\overline{G}_\Phi$, since G_Φ has the same structure. We need also the symmetrical FA A_Φ, and the grammar graph $\Gamma(\overline{G}_\Phi)$. Notice that A_Φ and $\Gamma(\overline{G}_\Phi)$ have the same set of states, and that A_Φ is a sub-graph of $\Gamma(\overline{G}_\Phi)$, which only differs by the absence of nonterminally-labeled arcs.
We say that two words w and w' are equivalent on a sequence of states $\pi = q_1, q_2, \ldots, q_n$ (or *path equivalent*), written $w \equiv_\pi w'$, iff in $\Gamma(\overline{G}_\Phi)$ there exist two paths, both with the state sequence π, such that w and w' are their labels.
We start from a string $w^{(0)} \in \mathrm{SLT}(\Phi)$; we will show that, for some $m > 0$ and for some axiom $W \in S$:
$w^{(0)} \rightsquigarrow_\Phi w^{(1)} \rightsquigarrow_\Phi \ldots \rightsquigarrow_\Phi w^{(m)} = \textcircled{\#} \odot \textcircled{\#}$ iff $\tilde{w}^{(0)} \Longleftarrow_{\overline{G}_\Phi} \tilde{w}^{(1)} \Longleftarrow_{\overline{G}_\Phi} \cdots \Longleftarrow_{\overline{G}_\Phi} \tilde{w}^{(m)} = W$,
where $\tilde{w}^{(0)} = w^{(0)}$, and $\forall i$, $\exists \pi_i : \tilde{w}^{(i)} \equiv_{\pi_i} w^{(i)}$.
We prove the theorem by induction on the reduction steps.
Base case: Consider $\tilde{w}^{(0)}$: it is by definition $\tilde{w}^{(0)} = w^{(0)}$, hence $\exists \pi : \tilde{w}^{(0)} \equiv_\pi w^{(0)}$.
Induction case: First, we prove that $w^{(t)} \rightsquigarrow_\Phi w^{(t+1)}$ implies $\tilde{w}^{(t)} \Longleftarrow_{\overline{G}_\Phi} \tilde{w}^{(t+1)}$ with $\tilde{w}^{(t+1)} \equiv_{\pi_{t+1}} w^{(t+1)}$. To perform the reduction, we need a handle, let it be called x, such that $w^{(t)} = uxv \rightsquigarrow w^{(t+1)} = usv$, $s \in \Delta$. By induction hypothesis, we know that $\tilde{w}^{(t)} \equiv_{\pi_t} w^{(t)} = uxv$, therefore $\tilde{w}^{(t)} = \tilde{u}\tilde{x}\tilde{v}$ with $\tilde{u} \equiv_{\pi'_t} u$, $\tilde{x} \equiv_{\pi''_t} x$, and $\tilde{v} \equiv_{\pi'''_t} v$, with $\pi_t = \pi'_t \pi''_t \pi'''_t$. The equivalence $\tilde{x} \equiv_{\pi''_t} x$, with x handle, implies that there

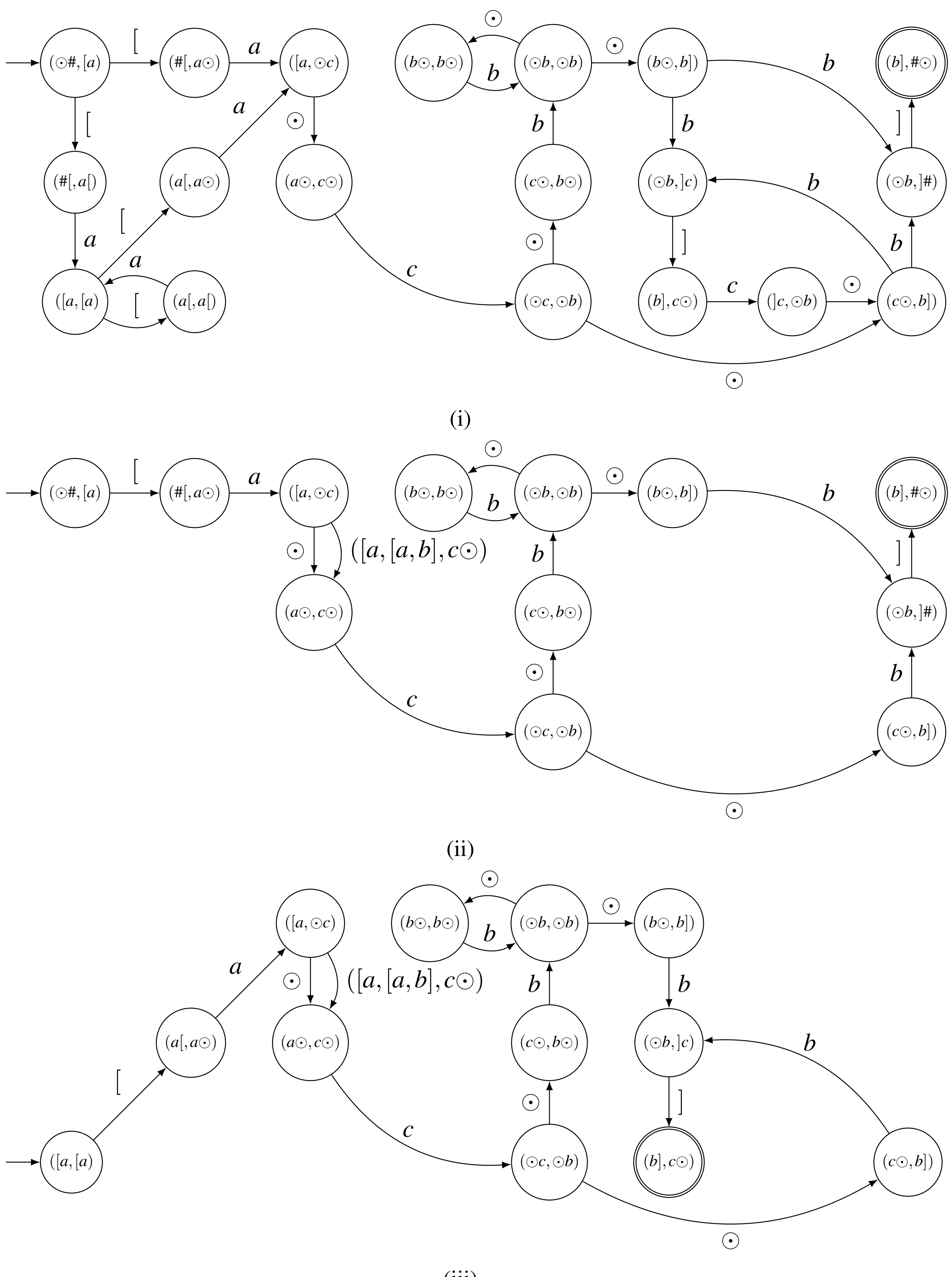

Figure 1: (i) Symmetrical FA A_Φ of Example 3.4. (ii) Automaton of the rule $X \to [a(\odot \cup Y)c\odot(b\odot)^*b]$ of Example 4.8. (iii) Automaton of the rule $Y \to [a(\odot \cup Y)c\odot(b\odot)^*b]$ of Example 4.8.

is a right part of a rule $X \to M_X \in P$, such that $\tilde{x} \in R_X$. Hence, $\tilde{w}^{(t)} \Longleftarrow_{\overline{G}_\Phi} \tilde{w}^{(t+1)} = \tilde{u}X\tilde{v}$ and $X = (t_{k-1}(\#u), i_{k-1}(xv\#), t_{k-1}(\#ux), i_{k-1}(v\#))$. The reduction relation implies that in A_Φ (and therefore also in $\Gamma(\overline{G}_\Phi)$) there is a path with states π_{t+1} and labels $w^{(t+1)}$: call π'_{t+1} the states of its prefix with label u, and π''_{t+1} those of its suffix with label v. Let us call q_u the last state of π'_{t+1} and q_v the first state of π''_{t+1}. By construction of $\overline{G}_\Phi$, in $\Gamma(\overline{G}_\Phi)$ there is a transition $q_u \xrightarrow{X} q_v$, while in A_Φ there is $q_u \xrightarrow{s} q_v$. From this it follows $\tilde{w}^{(t+1)} \equiv_{\pi'_{t+1}\pi''_{t+1}} w^{(t+1)}$.
We now prove that $\tilde{w}^{(t)} \Longleftarrow_{\overline{G}_\Phi} \tilde{w}^{(t+1)}$ implies $w^{(t)} \rightsquigarrow_\Phi w^{(t+1)}$, with $w^{(t+1)} \equiv_{\pi_{t+1}} \tilde{w}^{(t+1)}$. By definition of derivation, it is $\tilde{w}^{(t)} = \tilde{u}\tilde{x}\tilde{v} \Longleftarrow_{\overline{G}_\Phi} \tilde{w}^{(t+1)} = \tilde{u}X\tilde{v}$ for some $X \in V_N$. By induction hypothesis, we know that $\tilde{u}\tilde{x}\tilde{v} = \tilde{w}^{(t)} \equiv_{\pi_t} w^{(t)}$, hence $w^{(t)} = uxv$ with $\tilde{u} \equiv_{\pi'_t} u$, $\tilde{x} \equiv_{\pi''_t} x$, and $\tilde{v} \equiv_{\pi'''_t} v$, with $\pi_t = \pi'_t\pi''_t\pi'''_t$. From this it follows that $X = (t_{k-1}(\#u), i_{k-1}(xv\#), t_{k-1}(\#ux), i_{k-1}(v\#))$, and that x must be an handle. Therefore, $w^{(t)} = uxv \rightsquigarrow w^{(t+1)} = usv$, $s \in \Delta$, and in A_Φ (and in $\Gamma(\overline{G}_\Phi)$) there is a path with states π_{t+1} and labels $w^{(t+1)}$: call π'_{t+1} the states of its prefix with label u, and π''_{t+1} those of its suffix with label v. Let us call q_u the last state of π'_{t+1} and q_v the first state of π''_{t+1}. By construction of $\overline{G}_\Phi$, in $\Gamma(\overline{G}_\Phi)$ there is a transition (q_u, X, q_v), while in A_Φ there is (q_u, s, q_v). Hence $\tilde{w}^{(t+1)} \equiv_{\pi'_{t+1}\pi''_{t+1}} w^{(t+1)}$. □

Theorem 4.10. *Let G be any grammar in the family HOP(k,Φ) and $\overline{G}$ its tagged version. Let* $\mathrm{Red}(\Phi) = L(G_\Phi)$ *(respectively* $\overline{\mathrm{Red}}(\Phi) = L(\overline{G}_\Phi)$*) be the max-languages. The following inclusions hold:*

$$L(\overline{G}) \subseteq \overline{\mathrm{Red}}(\Phi), \quad L(G) \subseteq \mathrm{Red}(\Phi).$$

Proof. (Hint) Let $G = (V_N, \Sigma, P, S)$, $\overline{G} = (V_N, \Sigma \cup \Delta, \overline{P}, S)$, $G_\Phi = (V'_N, \Sigma, P', S')$ and $\overline{G}_\Phi = (V'_N, \Sigma \cup \Delta, \overline{P}', S')$. We prove that if, for $X \in S$, $X \overset{+}{\Longrightarrow}_{\overline{G}} w$ then, for some $Y' \in S'$, $Y' \overset{+}{\Longrightarrow}_{\overline{G}_\Phi} w$; we may assume both derivations are leftmost. If $X \overset{+}{\Longrightarrow}_{\overline{G}} uX_1v \Longrightarrow_{\overline{G}} uw_1v = w$ then w_1 is the leftmost handle in w, and by definition of max-grammar, there exists a derivation $uZ'v \Longrightarrow_{\overline{G}} uw_1v$ where Z' is the 4-tuple $(t_k(\#u), i_k(w_1v\#), t_k(\#uw_1), i_k(v\#))$.
Then, after the reduction or the derivation step, the position of the leftmost handle in uX_1v and in $uZ'v$ coincide, and we omit the simple inductive arguments that completes the proof.
Clearly, the two derivations of $\overline{G}$ and of $\overline{G}_\Phi$ have the same length and create isomorphic trees, which only differ in the nonterminal names. By applying the projection σ to both derivations, the inclusion $L(G) \subseteq \mathrm{Red}(\Phi)$ follows. □

Thus, for each set of tagged k-words Φ, the max-language $L(G_\Phi)$ includes every language in HOP(k,Φ), actually also any language in HOP(k,Φ'), where $\Phi' \subseteq \Phi$.

To prove the Boolean closure of HOP(k,Φ), we need the following lemma (the tedious proof is omitted) which extends Theorem 5 of Knuth [15] from CF to ECF grammars.

Lemma 4.11. *Let $G_{(),1}$ and $G_{(),2}$ be ECF parenthesis grammars. Then there exists an ECF parenthesis grammar $G_{()}$ such that $L(G_{()}) = L(G_{(),1}) - L(G_{(),2})$.*

Theorem 4.12. *For every k and $\Phi \subset \Sigma^{\square k}$, the language family HOP(k,Φ) is closed under union, intersection and under relative complement, i.e., $L_1 - L_2 \in$ HOP(k,Φ) if $L_1, L_2 \in$ HOP(k,Φ).*

Proof. Let $L_i = L(G_i)$ where $G_i = (V_{N_i}, \Sigma, P_i, S_i)$, for $i = 1,2$. We assume that the nonterminal names of the two grammars are disjoint.
Union. The grammar $G = (V_{N_1} \cup V_{N_2}, \Sigma, P_1 \cup P_2, S_1 \cup S_2)$ generates $L(G_1) \cup L(G_2)$ and is in HOP(k,Φ), since its set of tagged k grams is Φ.
Complement. Let $G_{(),i}$ be the parenthesis grammar of G_i, $i = 1,2$, and by Lemma 4.11 let $G_{()} = (V_N, \Sigma, P, S)$ be the parenthesis grammar such that $L(G_{()}) = L(G_{(),1}) - L(G_{(),2})$. Since G_1 and G_2 are

structurally unambiguous by Theorem 4.5, there exists a bijection between the sentences of L_i and $L(G_{(),i})$, $i = 1, 2$.
Define the grammar $G = (V_N, \Sigma, P, S)$ obtained from $G_{()}$ by erasing the parentheses from each rule right part. It is obvious that $L(G) = L_1 - L_2$ since, if x, y are sentences of $G_{()}$ and $\sigma(x) = \sigma(y)$, then $x = y$. It remains to prove that G has the HOP(k) property. Each sentence of $L(G)$ corresponds to one, and only one, sentence of $G_{()}$. Since $L(G) \subseteq L_1$, the tagged k-words of grammar G are a subset of the tagged k-words Φ of grammar G_1, which by hypothesis are not conflictual. The closure under intersection is obvious. $\square$

Combining Theorem 4.10 and Theorem 4.12, we have:

Corollary 4.13. *For every k and $\Phi \subset \Sigma^{\square k}$, the language family HOP(k, Φ) is a Boolean algebra having as top element the max-language* $\text{Red}(\Phi)$.

Our last result reaffirms for the HOP languages a useful property of OP languages.

Theorem 4.14. *For every k and $\Phi \in \Sigma^{\square k}$, the language family HOP(k, Φ) is closed under intersection with regular languages.*

Proof. (Hint) Let us consider a grammar $G_0 \in \text{HOP}(k, \Phi)$ and a regular language R_0. We first add tags to R_0 through the language substitution $\eta : \Sigma^2 \to \mathscr{P}(\Sigma \cdot \Delta \cdot \Sigma)$, such that $\eta(ab) = \{a\}\Delta\{b\}$. Consider the regular language $R_1 = \eta(R_0)$ and an FA $M_1 = (\Sigma \cup \Delta, Q_R, \delta_R, I_R, T_R)$ that recognizes R_1. Let A_Φ be the symmetrical automaton recognizing SLT(Φ). We apply the classic "product" construction for the language intersection of the two FA A_Φ and M_1; let the product machine be $(\Sigma \cup \Delta, Q, \delta, I, T)$. Note that a state of Q consists of three components (β_1, α_1, q_1): the look-back β_1 and look ahead α_1, where $\beta_1, \alpha_1 \in (\Sigma \cup \Delta)^{k-1}$ come from the states of A_Φ, while the state q_1 comes from Q_R.

By extending the construction presented in Definition 4.7, we proceed now to define the grammar $G_1 = (\Sigma \cup \Delta, V_N, P, S)$ for $\overline{\text{Red}}(\Phi) \cap R_1$ as follows.
– V_N is a subset of $Q \times Q$ such that $(\beta_1, [\gamma_1, q_1, \gamma_2], \alpha_2, q_2) \in V_N$, for some $\beta_1, \gamma_1, \gamma_2, \alpha_2$, q_1, q_2.
– $S \subseteq V_N$ and $X \in S$ if, and only if, $X = (\gamma_1 \odot \#, \alpha_1, q_1, \beta_1, \# \odot \gamma_2, q_2)$, for some $\gamma_1, \gamma_2, \alpha_1, \beta_1$, $q_1 \in I_R$, $q_2 \in F_R$.
– Each rule $X \to M_X \in P$ is such that the right part is an FA $M_X = (\Sigma \cup \Delta, Q_X, \delta_X, \{p_I\}, \{p_T\})$ where $Q_X \subseteq Q$. (For each X there exists only one M_X.) Let $X = (\beta_X, \alpha_X, q_X, \beta'_X, \alpha'_X, q'_X)$. Then: $p_I = (\beta_X, \alpha_X, q_X)$, $p_T = (\beta'_X, \alpha'_X, q'_X)$, $\delta_X = (\delta \cup \delta') - \delta''$,

$$\delta' = \left\{ (\beta_1, \alpha_3, q_1) \overset{(\beta_1,\alpha_1,q_1,\beta_2,\alpha_2,q_2)}{\longrightarrow} (\beta_3, \alpha_2, q_2) \;\middle|\; \begin{array}{l} (\beta_1, \alpha_1, q_1, \beta_2, \alpha_2, q_2) \in V_N, \\ (\beta_1, \alpha_3, q_1) \xrightarrow{\odot} (\beta_3, \alpha_2, q_2) \in \delta \,\vee \\ (\beta_1, \alpha_3, q_1) = p_I \wedge (\beta_1, \alpha_3, q_1) \xrightarrow{[} (\beta_3, \alpha_2, q_2) \in \delta \,\vee \\ (\beta_3, \alpha_2, q_2) = p_T \wedge (\beta_1, \alpha_3, q_1) \xrightarrow{]} (\beta_3, \alpha_2, q_2) \in \delta \end{array} \right\}$$

$$\delta'' = \begin{array}{l} \left\{ q' \xrightarrow{[} q'' \in \delta \;\middle|\; q' \neq p_I \right\} \cup \left\{ q' \xrightarrow{]} q'' \in \delta \;\middle|\; q'' \neq p_T \right\} \cup \\ \left\{ q' \xrightarrow{x} p_I \in \delta \right\} \cup \left\{ p_T \xrightarrow{x} q' \in \delta \right\}. \end{array}$$

It is easy to see that $L(G_1) = \overline{\text{Red}}(\Phi) \cap R_1$. If we remove tags by taking $G_2 = (V_N, \Sigma, \{X \to \sigma(R_X) \mid X \to R_X \in P\}, S)$, we see that $G_2 \in \text{HOP}(k, \Phi)$ by construction, and $L(G_2) = \text{Red}(\Phi) \cap R_0$. By Cor. 4.13, $L(G_0) \cap L(G_2) = L(G_0) \cap R_0$ is in HOP(k, Φ). $\square$

5 Related work and Conclusion

Earlier attempts have been made to generalize the *operator precedence* model and other similar grammar models. We discuss some relevant works and explain how they differ from the *higher-order operator precedence* model.

Floyd himself proposed the *bounded-context* grammars [12], which use left and right contexts of bounded length to localize the edges of the handle; unfortunately, the contexts contain also nonterminals and so lose the closure properties of OP languages as well as the possibility to do local parsing.

Chain-driven languages [7] are a recent extension of OP languages, which shares with HOP the idea of specifying the syntax structure by non-conflictual tags, but differs in technical ways we cannot describe here. The resulting family offers some significant gain in expressive capacity over OP, enjoys local parsability, but it has poor closure properties, and cannot be easily formulated for contexts larger than one terminal. Notice that the automata-theoretic approach presented in [7] can be naturally applied to HOP languages for proving their local parsability.

Since HOP extend the OP language family, which in turn include the input-driven (or VP) language [8] family, it is interesting to compare the HOP family with the recent extension of VP languages, recognized by *tinput-driven pushdown automata* (TDPDA) [16], which enjoy similar closure properties. The families HOP and TDPDA are incomparable: on one side, the language $\{a^n b a^n \mid n \geq 1\} \in \text{TDPDA} - \text{HOP}$, on the other side, TDPDA only recognize real-time languages, and thus fail the non-realtime language which is $\{a^m b^n c^n d^m \mid n,m \geq 1\} \cup \{a^m b^+ e d^m \mid m \geq 1\} \in \text{HOP}(3)$. Moreover the tinput parser is not suitable for local parsing, because it must operate from left to right, starting from the first character.

Recalling that OP grammars have been applied in early grammar inference studies, we mention two loosely related language classes motivated by grammar inference research, which strives to discover expressive grammar types having good learnability properties. Within the so-called distributional approach, several authors have introduced various grammar types based on a common idea: that the syntax class of a word v is determined by the left and right contexts of occurrence, the context lengths being finite integers k and ℓ. Two examples are: the (k,ℓ) *substitutable* CF languages [22] characterized by the implication $x_1 v y_1 u z_1, x_1 v y_2 u z_1, x_2 v y_1 u z_2 \in L$ implies $x_2 v y_2 u z_2 \in L$ where $|v| = k$ and $|u| = \ell$; and the related hierarchies of languages studied in [18]. A closer comparison of HOP and language classes motivated by grammar inference would be interesting.

Since HOP is a new language model, its properties have been only partially studied. Thus, it remains to be seen whether other known theoretical properties of OP languages (such as the closure under concatenation and star or the invariance with respect to the CF non-counting property [6]) continue to hold for HOP.

We finish by discussing the potential for applications. First, the enhanced generative capacity of higher degree HOP grammars in comparison to OP grammars may in principle ease the task of writing syntactic specifications, but, of course, this needs to be evaluated for realistic cases. We are confident that the practical parallel parsing algorithm in [3] can be extended from OP to HOP grammars.

To apply HOP to model-checking of infinite-state systems, the model has to be extended to ω-languages and logically characterized, as recently done for OP languages in [17].

Last, for grammar inference: we observe that it would be possible to define a partial order based on language inclusion, within each subfamily of HOP(k) languages closed under Boolean operation, i.e., structurally compatible. Such a partially ordered set of grammars and languages, having the max-grammar as top element, is already known [10, 9] for the OP case, and its lattice-theoretical properties have been exploited for inferring grammars using just positive information sequences [2]. The availability of the

k-ordered hierarchy may then enrich the learnable grammar space.

References

[1] R. Alur & P. Madhusudan (2009): *Adding nesting structure to words*. *JACM* 56(3).

[2] D. Angluin & C. H. Smith (1983): *Inductive Inference: Theory and Methods*. *ACM Comput. Surv.* 15(3), pp. 237–269, doi:10.1145/356914.356918.

[3] A. Barenghi, S. Crespi Reghizzi, D. Mandrioli, F. Panella & M. Pradella (2015): *Parallel parsing made practical*. *Sci. Comput. Program.* 112, pp. 195–226, doi:10.1016/j.scico.2015.09.002.

[4] B. von Braunmühl & R. Verbeek (1983): *Input-driven languages are recognized in log n space*. In: *Proc. of the Symp. on Fundamentals of Computation Theory, LNCS 158*, Springer, pp. 40–51.

[5] P. Caron (2000): *Families of locally testable languages*. *Theor. Comput. Sci.* 242(1-2), pp. 361–376, doi:10.1016/S0304-3975(98)00332-6.

[6] S. Crespi Reghizzi, G. Guida & D. Mandrioli (1981): *Operator Precedence Grammars and the Noncounting Property*. *SIAM J. Computing* 10, pp. 174—191.

[7] S. Crespi Reghizzi, V. Lonati, D. Mandrioli & M. Pradella (2017): *Toward a theory of input-driven locally parsable languages*. *Theor. Comput. Sci.* 658, pp. 105–121, doi:10.1016/j.tcs.2016.05.003.

[8] S. Crespi Reghizzi & D. Mandrioli (2012): *Operator Precedence and the Visibly Pushdown Property*. *JCSS* 78(6), pp. 1837–1867.

[9] S. Crespi Reghizzi, D. Mandrioli & D. F. Martin (1978): *Algebraic Properties of Operator Precedence Languages*. *Information and Control* 37(2), pp. 115–133.

[10] S. Crespi Reghizzi, M. A. Melkanoff & L. Lichten (1973): *The Use of Grammatical Inference for Designing Programming Languages*. *Commun. ACM* 16(2), pp. 83–90.

[11] R. W. Floyd (1963): *Syntactic Analysis and Operator Precedence*. *JACM* 10(3), pp. 316–333.

[12] R. W. Floyd (1964): *Bounded context syntactic analysis*. *Commun. ACM* 7(2), pp. 62–67.

[13] D. Grune & C. J. Jacobs (2008): *Parsing techniques: a practical guide*. Springer, New York.

[14] M. A. Harrison (1978): *Introduction to Formal Language Theory*. Addison Wesley.

[15] D. Knuth (1967): *A characterization of parenthesis languages*. *Information and Control* 11, pp. 269–289.

[16] M. Kutrib, A. Malcher & M. Wendlandt (2015): *Tinput-Driven Pushdown Automata*. In J. Durand-Lose & B. Nagy, editors: *Machines, Computations, and Universality - 7th International Conference, MCU 2015*, *LNCS* 9288, Springer, pp. 94–112, doi:10.1007/978-3-319-23111-2_7.

[17] V. Lonati, D. Mandrioli, F. Panella & M. Pradella (2015): *Operator Precedence Languages: Their Automata-Theoretic and Logic Characterization*. *SIAM J. Comput.* 44(4), pp. 1026–1088, doi:10.1137/140978818.

[18] F. M. Luque & G. G. Infante López (2010): *PAC-Learning Unambiguous* k, l-*NTS* $^{<=}$ *Languages*. In: *Grammatical Inference: Theoretical Results and Applications, 10th International Colloquium, ICGI 2010, Valencia, Spain, September 13-16, 2010. Proceedings*, pp. 122–134.

[19] R. McNaughton (1967): *Parenthesis Grammars*. *JACM* 14(3), pp. 490–500.

[20] R. McNaughton & S. Papert (1971): *Counter-free Automata*. MIT Press, Cambridge, USA.

[21] K. Mehlhorn (1980): *Pebbling Mountain Ranges and its Application of DCFL-Recognition*. In: *Automata, languages and programming (ICALP-80)*, *LNCS* 85, pp. 422–435.

[22] R. Yoshinaka (2008): *Identification in the Limit of k, l-Substitutable Context-Free Languages*. In A. Clark, F. Coste & L. Miclet, editors: *Grammatical Inference: Algorithms and Applications, 9th International Colloquium, ICGI 2008*, *LNCS* 5278, Springer, pp. 266–279, doi:10.1007/978-3-540-88009-7_21.