Attribute-Based Precedence Relations for Free-Form Grammars

Angelo Borsotti[1], Stefano Crespi Reghizzi[1,2] and Matteo Pradella[1,2]

[1]*DEIB, Politecnico di Milano,*
via Ponzio 34/5, Milano, Italy
[2]*CNR - IEIIT*

Abstract

We extend the well-known and efficient parsing algorithm of operator-precedence (OP) grammars to unrestricted context-free grammar rules by means of newly defined precedence relations between nodes of syntax trees. Such relations are defined over pairs of attributes of Knuth's synthesized type that can be computed by a bottom-up parser that operates locally and in any direction (or in parallel) on the input. An algorithm for computing, given a grammar, the attributes and their relations is presented and supported by a tool. The new family of PR grammars is characterized by the absence of conflicts between precedence relations. The corresponding family of PR languages strictly include the OP family. A new normal form of PR grammars permits to prove the closure of PR languages under union. Future developments towards Boolean closures and application to model checking are mentioned.

Keywords

Context-free Grammars, Operator Precedence Languages, Precedence Relations, Tree Languages

1. Introduction

Within formal language theory, the classical topic of grammars and syntactic methods has recently seen advances motivated by new applications and by mathematical interest. We mention parsing algorithms that exploit parallel hardware architectures [1, 2], and formal specifications of non-finite-state systems [3] to be used in model checking. Concerning mathematical developments, we mention recent research applying logic and algebraic methods to context-free languages [4, 5, 6, 7]. Such studies deal with the operator-precedence (OP) subfamily of deterministic context-free languages [8], that was historically motivated by their fast and compact parsing algorithm and by the capacity to define artificial languages featuring operators layered by precedence. In later years, research on OP has never stopped, but has focused on different aspects: grammar inference from examples of sentences [9], Boolean closure properties [10], inclusion of visibly pushdown languages [11], parallel parsing [1, 2], ω-languages for formal specifications [12], non-counting properties [13], logical characterization [12], syntactic congruences [14].

Other efforts have attempted to increase the expressive power of OP grammars while keeping the operator-form of their rules (e.g., [15, 16]), or to remove the inconvenience of the operator form (e.g., Wirth-Weber [17]). Our contribution addresses the latter problem and differs from other old [17] and recent [18] proposals in that it strives to preserve most of the important formal properties of OP languages.

We define a novel concept of precedence relation (PR) that applies also to grammars not constrained by the operator-form. The new PRs differ from the known ones in their domain, which is neither the terminal alphabet of OP, nor the total grammar alphabet of the Wirth-Weber simple-precedence grammars, which by the way lack the closure and local parsability properties of OP. The domain of the new *attribute precedence relations* (APR) is a finite set of *attributes,* so called because they can be

computed bottom-up by a local parser as the semantic attributes of Knuth's attributed grammars [19]. Four binary relations, called topological, between tree nodes that are adjacent on an antichain, are introduced: *sibling* is self-explanatory, *take/yield* are similar to the OP relations of the same name, while *facing* is a new one that holds between two nodes placed at greater distance from their least common ancestor. In a tree decorated with the node attributes, the four topological relations induce corresponding relations on the attribute domain. The PR grammars are those that do not have conflicts in attribute relations. The PR language family strictly includes the OP family, of which it preserves some essential properties: we prove local parsability, closure under reversal and union. Other algebraic properties of OP languages (closure under concatenation and complement) are likely to hold but remain to be proved. To experiment with PR grammars, the algorithm that computes the attributes and their PRs has been implemented in a tool. We have found that many syntactic constructs found in typical deterministic languages satisfy the PR property, also when the grammar rules violate the operator form. The parsing algorithm for PR differs from the classic one for OP in that the arguments of precedence relations are not given in the input string, but are computed as synthesized attributes by the parser. The size of PR matrix, though larger than the one of the OP matrix, remains practical and should not jeopardize parsing speed.

After Sect. 2 on preliminaries, the main Sect. 3 contains the topological PRs, the definition of attributes and their PRs. Then the algorithm that computes and checks PRs for a grammar is presented. It involves a new normal form for attributed grammars. The proof of closure under union ends Sect. 3. Sect. 4 proves that each OP grammar is also PR, and that some PR grammars and languages are not OP. References to related work have been placed in all sections where pertinent.

2. Preliminaries

The alphabet is denoted by Σ and the empty string by ε. We say two letters x, y are *adjacent* in a string z, if xy is a substring of z.

We assume familiarity with the basic theories of CF languages. A grammar is denoted by $G = (\Sigma, V_N, P, S)$ where the terminal and non-terminal alphabets are respectively Σ and V_N and the total alphabet is $V = \Sigma \cup V_N$. The set P contains rules of the form $A \to \alpha$, $\alpha \in V^+$, $A \in V_N$. The set of axioms is $S \subseteq V_N$. We need some cases: *copy* rule $\alpha \in V_N$; *terminal* rule $\alpha \in \Sigma^+$; *operator form* rule if α does not contain two adjacent nonterminals. The reserved terminal $\#$ is used to delimit a string. We respectively denote by $L(G) \subseteq \Sigma^+$ and $T(G)$ the language and the set of syntax trees defined by G.

Tree languages. We also use concepts and notation from the basic theory of regular tree languages (e.g., in [20]). A *tree domain* is a finite set of the form $\{\varepsilon, 1, 2, \ldots, 11, 12, \ldots\}$.[1] In a tree t, each node is identified by a unique element of the tree domain dom_t, for brevity here called a *node-identifier nid*. The root nid is ε, the root children have nid's $1, 2, \ldots, r$, where r is the rank of the root, etc. Each node has a symbolic *label* on the tree alphabet $\Sigma \cup \Gamma$, where the tree leaves have labels in Σ, and the alphabet Γ is used for labeling the internal nodes. An *unlabeled* tree has $\Sigma = \Gamma = \{\bullet\}$, a *skeletal* tree has $\Gamma = \{\bullet\}$. The *frontier* of a tree is the sequence of its leaves, that respects the order in which they appear in a preorder visit of t, and is denoted by $F(t) \in \Sigma^+$.

We often assume for simplicity that the trees are *chain-free*, i.e., a node has either rank > 1 or rank 1 when its child label is in Σ.

We use the following relations on a tree domain: *parent* denoted $\dot{p}$, *sibling* denoted $\dot{s}$, and ancestor/descendant.

Given a tree, an *antichain* is a maximal lexically ordered sequence of nodes (nids) that are pairwise unrelated by the ancestor-descendant relation. Notice that if n' and n'' are adjacent in an antichain, they are either (i) adjacent siblings or (ii) descendants of adjacent siblings, such that n' is a rightmost descendant and n'' is a leftmost descendant; in case (ii) n' and n'' are cousins not necessarily of the same generation. E.g. some antichains of tree t in Fig. 1 are: $F(t)$; $11, 12, 13, 2, 3$; $1, 2, 3$; ε; on the other hand $12, 13, 2, 3$ is not an antichain since it is not maximal. The set of antichains of a tree is

---

[1] We assume to have rank at most 9, otherwise a separator such as $\cdot$ would be needed.

partially ordered by the reduction operation that substitutes the parent nid for the sequence of all its children nid's.

For a syntax tree, the label of an antichain is a string over $\Sigma \cup V_N$ that derives from an axiom, i.e., a sentential form.

3. Precedence relations

The traditional order of presentation of operator precedence as a relation between the terminal symbols of a grammar is not possible here because in a free form grammar some rules may not contain terminals. However, the concept of precedence can be reformulated for certain objects more hidden than terminals, to be called *attributes*. Such attributes need to be bottom-up and locally computable by a parser that starts from the attributes of the input terminals, so that the precedence relations between attributes make parsing deterministic. A natural question is whether nonterminal symbols can be used as attributes in the previous sense, since other well-known precedence-oriented grammar models (chiefly Wirth-Weber [21] and followers [22]) define precedence on terminals and nonterminals. The answer is negative since Wirth-Weber parsers necessarily operate from left to right and do not have the local parsing property; the corresponding language family loses the closure properties of OP languages, e.g., is not closed under union [23].

First, we introduce four topological precedence relations (facing, sibling, taking, yielding) on the nodes of (unlabeled) skeletal trees in analogy with the classical genealogical relations between nodes (parent, sibling, cousin, etc.). Then we define the attributes as bottom-up computable values on the nodes of skeletal trees whose frontier is labeled on Σ. For any tree in $T(G)$ and for any two tree nodes their topological precedence relation naturally maps on their attributes. A grammar has the PR property if between the same pair of attributes at most one of the four relations holds. The finite set of attributes found in the trees of grammar G and their APRs are computed by an algorithm.[2] The algorithm constructs a grammar in a new normal form, called *attributed*. An informal presentation of the parsing algorithm and its local parsability property comes after. The closure under union for the PR languages having non-conflictual APRs concludes the section.

Topological relations on unlabeled trees   For a tree t, with parent relation $\dot{p}$, we define four sets of binary relations for any pair of nodes n', n'' that are adjacent in an antichain.

Definition 1 (Precedence relations on tree domain). *Four topological precedence relation types (PR), named* yield precedence, take precedence, facing *and* sibling *are defined on the tree domain* dom_t. *Let nodes* n' *and* n'' *be adjacent in an antichain of tree* t *and let* n *be their nearest common ancestor. Notice that the adjacency* (n', n'') *implies that* n' *is met before* n'' *during a left-to-right, depth-first visit of the tree nodes.*

yield	$(n', n'') \in$ Yield, *better written in infix notation as* $n' \dot{y} n''$, *if* $n \dot{p} n'$ *(i.e.,* n' *is child of* n*) and* n'' *is a leftmost descendant but not a child of* n.
take	$(n', n'') \in$ Take, *or* $n' \dot{t} n''$, *if* n' *is a rightmost descendant but not a child of* n *and* $n \dot{p} n''$.
facing	$(n', n'') \in$ Face, *or* $n' \dot{f} n''$ *if* n' *and* n'' *are resp. a rightmost and a leftmost descendant − but not children of* n.
sibling	$(n', n'') \in$ Sibl, *or* $n' \dot{s} n''$ *if* n' *and* n'' *are children of* n.

The set of PR types is denoted by $\dot{R} = \{\dot{f}, \dot{s}, \dot{t}, \dot{y}\}$ *and* $\dot{r} \in \dot{R}$ *stands for a generic PR type.*
The graph with nodes dom_t *and with the arcs, labeled on* $\dot{R}$, *that represent the parent relation* $\dot{p}$ *(already in* t*) and the precedence relations* $\dot{R}$, *is called a* parse graph; *it is a* supergraph *of the tree graph* t, *denoted* super(t).

---

[2]The algorithm is incorporated in the interactive tool (avaible at https://pradella.faculty.polimi.it/prgrweb.html) which aids testing of the PR property, otherwise painful by hand.

Clearly, on a given tree the relations in $\dot{R}$ are mutually exclusive, i.e., disjoint, and for any pair of (antichain) adjacent nodes n' and n'' exactly one $n \dot{r} n'$ exists. Notice also that in the traditional representation of syntax trees the sibling arcs are understood and not displayed.

The antichains are:
11 12 13 211 22 3,
1 211 22 3,
1 21 22 3,   1 2 3,
11 12 13 21 22 3,
11 12 13 2 3
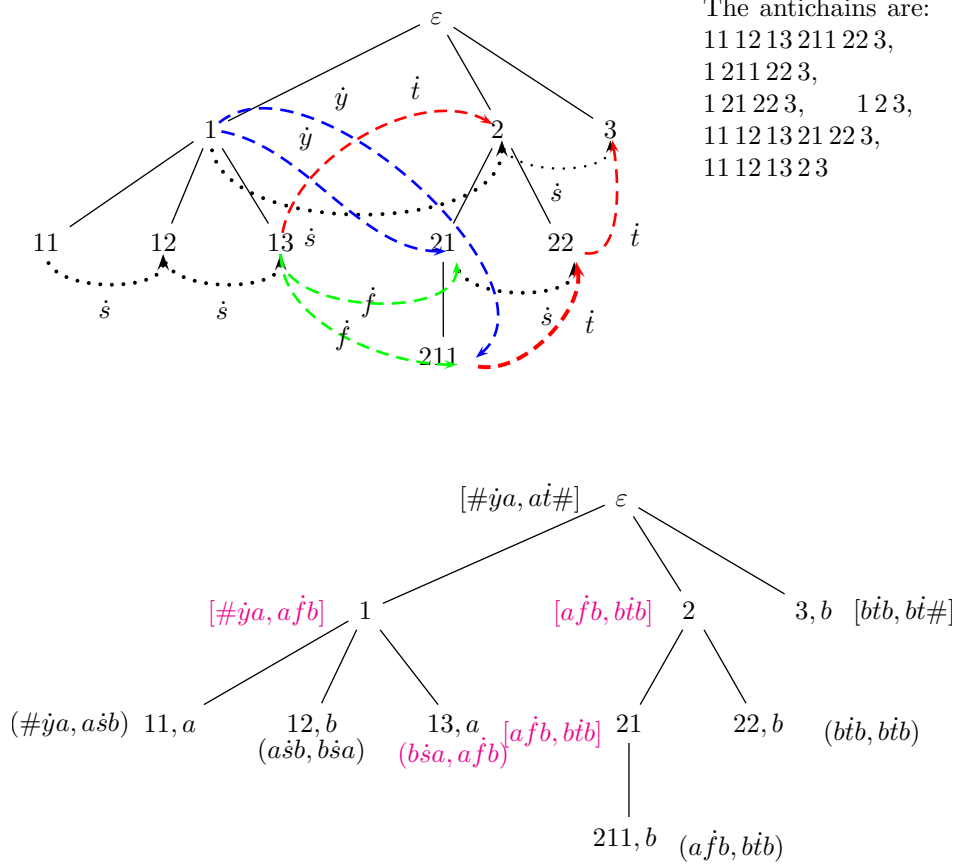
Figure 1: Top. The graph *super*(t) representing the $\dot{R}$ relations on the tree domain t. Notice that the antichains are the maximal left-to-right paths over the dashed/dotted arcs representing PR's. Bottom. The attributes for the skeletal tree with frontier *ababbb*. Internal and leaf attributes are resp. bracketed and parenthesized. The attributes of nodes 1, 2, 13 and 21 (evidenced in magenta) are involved in conflicts.

Example 1 (Topological PRs). *Fig. 1, top, shows a tree (solid arcs representing $\dot{p}$) and its supergraph (dashed arcs for $\dot{t}$, $\dot{y}$ and dotted ones for $\dot{s}$). The maximal paths over $\dot{R}$ arcs (excluding $\dot{p}$) are the antichains. In antichain 1 21 22 3 the arcs between the adjacent pairs (1, 21), (21, 22), and (22, 3) have the respective type $\dot{y}$, $\dot{s}$, and $\dot{t}$.*

3.1. Node attributes and their precedence relations

In a labeled tree domain, the topological PRs naturally induce four corresponding relations on node labels. Our main contribution is the definition of a novel domain of node labels, called *attributes* such that their precedence relations will permit to extend the desirable properties of OP grammars to a broader family of CF grammars and languages.

To define node attributes on a tree we introduce a function that can be computed bottom-up by a local parser.[3] Intuitively, a *local* (bottom-up deterministic) parser has the property that its decision to enter a shift or a reduce move only depends on the current input letter and on a finite neighborhood thereof. This enables the parser to use precedence relations between attributes to locate the next handle

---

[3]Ours are *synthesized semantic attributes* in Knuth's [19] sense, since a node attribute is computed on a tree as a function of the values of children attributes.

inside the current antichain of the tree under construction. Upon handle reduction, the parser computes the attribute of the parent node, and a new antichain replaces the current one. Such a process can be done in any order, not necessarily left-to-right.

We stress that not any locally computable function would qualify for the intended use, because some essential requirements must me met. The attribute should

1. permit to parse any OP grammar,

2. enable local parsability for non-operator-form grammars of common syntax constructs,

3. preserve all or most of the closure properties of OP languages;

4. moreover, the cost of computing attributes should not penalize parsing speed.

The following definition combines simplicity with empirical adequacy; it is the simplest one among several others we examined and discarded.

The *attribute* of a node n has the form $\langle \lambda, \rho, \textit{flag} \rangle$ where λ and ρ are precedence relations, respectively for the left- and right-*borders* (defined below) of the node. The *flag* component is in $\{\textit{leaf}, \textit{internal}\}$, specifying whether the rank of n is null or not. For readability, we will write (λ, ρ) for $\langle \lambda, \rho, \textit{leaf} \rangle$, and $[\lambda, \rho]$ for $\langle \lambda, \rho, \textit{internal} \rangle$. We exclude from the next definition any tree containing a linear chain, in agreement with the practice of OP grammar parsing.

Definition 2 (Attributes of tree nodes). *Let t be a chain-free tree with frontier $F(t)$ over Σ. For a leaf node n, let $pred(n)$ and $suc(n)$ be the (adjacent) leaf node resp. preceding and following n in the string $F(t)$. If n is the first or last symbol of $F(t)$, resp. let $pred(n) = \#$ and $suc(n) = \#$.*
Denote by $pred(n) \, \dot{r}_1 \, n$ and by $n \, \dot{r}_2 \, suc(n)$, with $\dot{r}_1, \dot{r}_2 \in \dot{R}$, the topological PR (see Def. 1) between the nodes $pred(n)$, n and between n, $suc(n)$, respectively. Conventionally, for $a \in \Sigma$, assume $\# \, \dot{y} \, a$ and $a \, \dot{t} \, \#$.
The recursive definition of the attribute α of a node n follows.

$$\alpha(n) = \begin{cases} (pred(n) \, \dot{r}_1 \, n \,, \; n \, \dot{r}_2 \, suc(n)), & \textit{if } n \textit{ is a leaf} \\ [x, y'], & \textit{if } n \textit{ has rank } \ell > 0, \textit{ where} \\ & \alpha(n1) = \langle x, y, f \rangle, \\ & \alpha(n\ell) = \langle x', y', f' \rangle \end{cases} \tag{1}$$

A tree where each node carries its attribute is called attributed. *The set of attributes of a tree t and a tree language T are denoted resp. by $A(t)$ and $A(T)$, or $A(G)$ if $T = T(G)$.*
The set $A(T)$ conflicts iff it contains two attributes α, α' such that $\alpha = \langle a \, \dot{r}_1 \, b \,, \; b \, \dot{r}_2 \, c, f \rangle$, $\alpha' = \langle a \, \dot{r}'_1 \, b \,, \; b \, \dot{r}'_2 \, c, f' \rangle$, and $\dot{r}_1 \neq \dot{r}'_1$, or $\dot{r}_2 \neq \dot{r}'_2$.

Given a tree with frontier over Σ, we compute the attributes starting with the leaf attributes, then proceeding in any topological order with respect to the child-parent relation. Fig. 1, bottom, shows the attributes for a skeletal tree with frontier $ababa$.

Since the left and right borders of a node are the same as respectively the left border of the first child and the right border of the last child, if node n has rank 1 and its child $n1$ is internal, their attributes are identical, $\alpha(n) = \alpha(n1)$, so that the two values are redundant and some spurious conflicts arise when their PRs are computed. To exclude them, Def. 2 is restricted to chain-free trees.

We formalize the precedence relations between the attributes of adjacent nodes in a tree.

Definition 3 (Precedence relations on node attributes). *Let t be a tree and consider the parse graph $super(t)$. Let n' and n'' be nodes in dom_t and let the topological relation $n' \, \dot{r} \, n''$ hold. Then the attribute pair $(\alpha(n'), \alpha(n''))$ has the attribute precedence relation (APR) $\dot{r}$, i.e., $\alpha(n') \, \dot{r} \, \alpha(n'')$.*
We call $\text{APR}(t)$ the set of PR among attribute pairs of the nodes of a tree t. Moreover, given a set T of trees, we define $\text{APR}(T) = \bigcup_{t \in T} \text{APR}(t)$.
The set $\text{APR}(T)$ conflicts iff it contains two relations $\alpha \, \dot{r} \, \alpha'$ and $\alpha \, \dot{r}' \, \alpha'$ with $\dot{r} \neq \dot{r}'$.
A set T of attributed trees has the precedence relation property (PRP) if no conflict occurs in $\text{APR}(T)$. A grammar G has the PRP if the set $A(G)$ has the PRP; we say that G is a PR grammar.

Remarks. The two entities involved in a conflict may occur in the same or in distinct trees of language T. Two types of conflicts are defined: conflict (1) holds between leaf attributes involving the same triple of terminals, while conflict (2) concerns two APRs between the same attribute pair α, α'. Notice that conflicts of type (2) may involve leaf and internal attributes.

Example 2 (Precedence relations on attributes). *The following* APR*s for the attribute set $A(t)$ of Fig. 1, bottom, are easily found by inspecting the super-graph in Fig. 1, top. We omit the relations involving the delimiter, e.g., $\# \, \dot{y} \, [\# \dot{y} a, a \dot{f} b]$.*

	1	11	12	13	2 = 21	211	22	3
$1[\# \dot{y} a, a \dot{f} b]$					$\dot{s}, \dot{y}$	$\dot{y}$		
$11(\# \dot{y} a, a \dot{s} b)$			$\dot{s}$					
$12(a \dot{s} b, b \dot{s} a)$				$\dot{s}$				
$13(b \dot{s} a, a \dot{f} b)$					$\dot{t}, \dot{f}$	$\dot{f}$		
$2 = 21[a \dot{f} b, b \dot{t} b]$							$\dot{s}$	$\dot{s}$
$211(a \dot{f} b, b \dot{t} b)$							$\dot{t}$	
$22(b \dot{t} b, b \dot{t} b)$								$\dot{t}$
$3(b \dot{t} b, b \dot{t} \#)$								

Set PR *has two conflicts of type (2) (evidenced in magenta) between the attribute of node 1 vs the attribute of nodes 2, 21 and between the attribute of node 13 vs 2, 21.*

Example 3 (PR on grammars.). *The grammar $G_1 : \{S \to aSaba \mid c\}$ has the PRP. Its attributes and their precedence relations* APR*s (explained later) follow.*
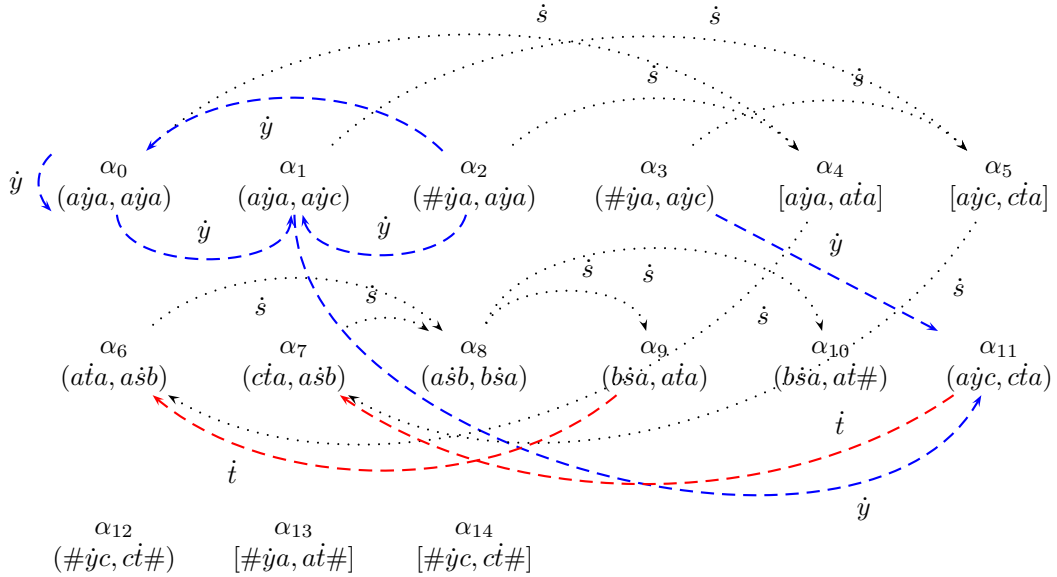
We anticipate that $L(G_1)$ is not an OP language.

Computation of the attributes and PRs for a grammar   We outline the algorithm implemented in our interactive tool that computes the attributes of a grammar and their PRs. Preliminarily, we observe that it would be unpractical to compute attributes and PRs on a tree by tree basis, since the number of syntax trees in a language is often not finite and it is not immediate to set a bound on the number of trees needed. Let $G = (\Sigma, V_N, P, S)$ and denote by $A(G)$ the set of attributes of any node in any tree in $T(G)$.

We first compute for each nonterminal $X \in V_N$ the borders of the terminal strings derived from X.

$$border(X) = \{(b, b) \mid X \to b \in P\} \cup \{(b, c) \mid X \xrightarrow{+} bxc \text{ where } x \in \Sigma^*\}.$$

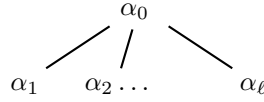The algorithm computes a grammar $\widetilde{G} = (\widetilde{\Sigma}, \widetilde{V}_N, \widetilde{P}, \widetilde{S})$, called *attributed normal form* ANF, essentially equivalent to G, whose terminal and nonterminal symbols are resp. in $\Sigma \times A(G)$ and in $V_N \times A(G)$. In the former case, the attributes pertain to leaves, in the latter case they pertain to internal nodes. The algorithm computes $A(G)$ along with the rules of $\widetilde{G}$, starting from the axioms. From the ANF rules, the APRs are then computed.

Initialization. All components of $\widetilde{G}$ are set to $\emptyset$.

(1) For each axiom $X \in S$, for each pair $(b, c) \in border(X)$
$\widetilde{V}_N := \widetilde{V}_N \cup \{\langle X, \alpha \rangle\}$ where $\alpha = \langle \#\dot{y}b, c\dot{t}\#, internal \rangle$

(2) For each nonterminal $\langle X_0, \alpha_0 \rangle \in \widetilde{V}_N$, for each rule $X_0 \to X_1 \ldots X_\ell \in P$

Consider the tree, where $\alpha_i = \left\langle \overbrace{a_i \dot{r}_i b_i}^{\lambda_i}, \overbrace{c_i \dot{r}'_i d_i}^{\rho_i}, flag_i \right\rangle$.

$$
\begin{array}{c}
\alpha_0 \\
\diagup \;\; \big| \;\; \diagdown \\
\alpha_1 \quad \alpha_2 \ldots \qquad \alpha_\ell
\end{array}
$$

Compute all ℓ-tuples $\alpha_1, \alpha_2, \ldots, \alpha_\ell$ of attribute values such that the following conditions hold.

1. $\lambda_1 = \lambda_0$, $\rho_\ell = \rho_0$;
2. $\forall 1 \le i \le \ell, (a_i, d_i) \in border(X_i)$;
3. $\forall 1 \le i < \ell, \lambda_{i+1} = \rho_i$;
4. $\forall 1 \le i < \ell$, the relation type is defined by the table:
   $flag_i = leaf \wedge flag_{i+1} = leaf: \qquad \dot{r}'_i = \dot{s}$
   $flag_i = leaf \wedge flag_{i+1} = internal: \qquad \dot{r}'_i = \dot{y}$
   $flag_i = internal \wedge flag_{i+1} = leaf: \qquad \dot{r}'_i = \dot{t}$
   $flag_i = internal \wedge flag_{i+1} = internal: \quad \dot{r}'_i = \dot{f}$

For each such ℓ-tuple add the corresponding nonterminal and rule to the ANF grammar:

$$
\begin{aligned}
\widetilde{V}_N &:= \widetilde{V}_N \cup \{\langle X_1, \alpha_1 \rangle, \ldots, \langle X_\ell, \alpha_\ell \rangle\} \\
\widetilde{P} &:= \widetilde{P} \cup \{\langle X_0, \alpha_0 \rangle \to \langle X_1, \alpha_1 \rangle \ldots \langle X_\ell, \alpha_\ell \rangle\}.
\end{aligned}
$$

(3) If nothing new is produced by step (2) terminate, else return to step (2).

Notes. α_0 is the value that would be computed from $\alpha_1, \alpha_2, \ldots, \alpha_\ell$ by Def. 2, Eq. (1). For all i, $\dot{r}_{i+1} = \dot{r}'i$.
   The attributes $A(G)$ are the second components of the nonterminals $\widetilde{V}_N$.
   The relations $\mathrm{APR}(G)$ are computed as follows.
   For each pair of adjacent symbols W, Z in the right-hand side of a rule of $\widetilde{P}$ the following PRs exist between the attributes of a rightmost descendant W' of W and of a leftmost descendant Z' of Z.

condition		relation		condition		relation
$W' = W$ leaf,	$Z' = Z$ leaf	$\dot{s}$		$W' = W$ leaf,	$Z \xrightarrow{+} Z'x$	$\dot{y}$
$W \xrightarrow{+} yW'$,	$Z' = Z$	$\dot{t}$		$W \xrightarrow{+} yW'$,	$Z \xrightarrow{+} Z'x$	$\dot{f}$

The next property of ANF grammars immediately descends from the construction above.

Proposition 1. *Let $T(G)$ be a set of chain-free syntax trees. There is a one-to-one correspondence between the ANF trees $T(\widetilde{G})$ and the trees $T(G)$ such that for each pair of corresponding trees $\widetilde{t}$ and t, for each pair $\widetilde{n}, n$ of corresponding nodes in $\widetilde{t}$ and t, the labels are resp. $\langle C, \alpha(n) \rangle$ and C where $C \in V$.*

Therefore $L(G)$ is identical to the projection on Σ of $L(\widetilde{G})$.

Remark. Proposition 1 can be made to hold also when a tree in $T(G)$ contains a chain: it suffices to collapse its (internal) nodes into a single node and to adjust accordingly the correspondence between nonterminal names.

Example 4 (ANF grammar). *For the equivalent variant $G_2 : \{S \to ASaba \mid c, \ A \to a\}$ of grammar $G_1 : \{S \to aSaba \mid c\}$ in Ex. 3, we show the ANF as output by the tool:*

$$
\begin{aligned}
0 &: S[\#\dot{y}a, a\dot{t}\#] \to && A[\#\dot{y}a, a\dot{f}a] \; S[a\dot{f}a, a\dot{t}a] \; a(a\dot{t}a, a\dot{s}b) \; b(a\dot{s}b, b\dot{s}a) \; a(b\dot{s}a, a\dot{t}\#) \\
1 &: S[\#\dot{y}a, a\dot{t}\#] \to && A[\#\dot{y}a, a\dot{f}c] \; S[a\dot{f}c, c\dot{t}a] \; a(c\dot{t}a, a\dot{s}b) \; b(a\dot{s}b, b\dot{s}a) \; a(b\dot{s}a, a\dot{t}\#) \\
2 &: S[\#\dot{y}c, c\dot{t}\#] \to && c(\#\dot{y}c, c\dot{t}\#) \\
3 &: A[\#\dot{y}a, a\dot{f}a] \to && a(\#\dot{y}a, a\dot{f}a) \\
4 &: S[a\dot{f}a, a\dot{t}a] \to && A[a\dot{f}a, a\dot{f}a] \; S[a\dot{f}a, a\dot{t}a] \; a(a\dot{t}a, a\dot{s}b) \; b(a\dot{s}b, b\dot{s}a) \; a(b\dot{s}a, a\dot{t}a) \\
5 &: S[a\dot{f}a, a\dot{t}a] \to && A[a\dot{f}a, a\dot{f}c] \; S[a\dot{f}c, c\dot{t}a] \; a(c\dot{t}a, a\dot{s}b) \; b(a\dot{s}b, b\dot{s}a) \; a(b\dot{s}a, a\dot{t}a) \\
6 &: A[\#\dot{y}a, a\dot{f}c] \to && a(\#\dot{y}a, a\dot{f}c) \\
7 &: S[a\dot{f}c, c\dot{t}a] \to && c(a\dot{f}c, c\dot{t}a) \\
8 &: A[a\dot{f}a, a\dot{f}a] \to && a(a\dot{f}a, a\dot{f}a) \\
9 &: A[a\dot{f}a, a\dot{f}c] \to && a(a\dot{f}a, a\dot{f}c)
\end{aligned}
$$

The attributed tree of $acaba$ is below:
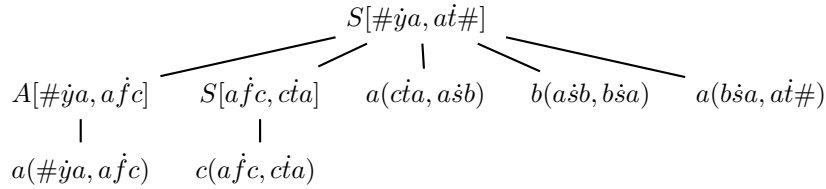
Precedence-parsing algorithm The parsing algorithm for PR grammars is analogous to the one for OP grammars described in [8] and in textbooks, see e.g. Alg. 4.5 of [24]. It is a shift-reduce algorithm purely driven by the APRs and constructs a skeletal parse tree with nodes labeled by attributes. Notice that the nonterminals do not influence parsing and are not pushed on the parsing tape. Following tradition, we describe the algorithm as a left-to-right one that constructs the tree in the reversed order of a rightmost derivation, but in reality it may work in any order, also mixing leftwards and rightwards moves; it can also operate in parallel if the input is sectioned into substrings, each one processed by an independent parser [1].

Such a parser accepts also some strings not in $L(G)$ if their APRs comply with $\mathrm{PR}(G)$. Far from being a defect, this property is fundamental for the algebraic theory of OP languages [10, 11, 12] and its application to model-checking of formal specification beyond the finite-state domain [5, 7]. To obtain the actual syntax tree of grammar G, a second algorithm tries to match the attributed skeletal parse tree against the grammar rules. If it does not succeed the string is rejected, otherwise a syntax tree is returned. The second algorithm does not differ from the classic case of OP grammars and we omit it.

The parsing algorithm differs from the classic OP one in two related aspects: (i) the grammar rules may exhibit adjacent nonterminals, and (ii) precedence relations involve attributes instead of terminals. We informally proceed to describe it.

The initial parser tape (which in left-to-right mode is a pushdown stack) is the input $x = \#b_1 \dots b_n\#$. For the current input letter b_i, the attribute $\alpha(b_i)$ is found in the APR table by look-up of the triple $b_{i-1}b_ib_{i+1}$, and it is pushed on the stack. If relation $\alpha(b_i) \dot{r} \alpha(b_{i+1})$ is $\dot{s}$ or $\dot{y}$ attribute $\alpha(b_{i+1})$ is pushed on the stack together with $\dot{r}$ (shift move). If $\dot{r} = \dot{f}$ or $\dot{r} = \dot{t}$ the parser executes a reduction; else $\dot{r}$ is undefined and the parser enters an error mode. In a reduction move, the stack is scanned backwards while relations $\dot{s}$ occur, until a relation $\dot{y}$ is found. The (finite) tape section in between $\dot{y}$ and $\dot{f}$ or $\dot{t}$ is the handle. The handle is replaced by a single element, namely the attribute value (computed as in Eq. (1)) of the handle parent. Then the relations between the two attributes, one preceding the other

following the parent attribute, determine the next parser move. The parser accepts if it completely scans the input and reduces to a single node, i.e. the root of the attributed skeletal tree.

Remark on local parsability. The earliest formal definition of *local parsability* (LP) we know is in [25] and concerns a restricted family of regular languages called code-words. For context-free languages, LP is a characteristic property of the OP family; it is preserved by some families that include OP and still use operator form rules, see in particular [15]. On the other hand, LP is lost by the more general family of Wirth-Weber *simple precedence* WWSP languages [21] that does not rely on the operator form, but extend the definition of PR to the total alphabet (terminal and nonterminal); the WWSP parser operates necessarily from left to right starting from the beginning of the input.

Our PR parser has the local parsing property. The local parsability property is important for efficiently implementing data-parallelism in parsers. The techniques for OP parser parallelization in [1] should be also applicable to the PR parser.

Language recognized by parser vs language defined by grammar   Since the set of strings recognized by the PR parser only depends on the PRs of G, we denoted it by $L(\mathrm{PR}(G))$. Clearly $L(G) \subseteq L(\mathrm{PR}(G))$. Interestingly, $L(\mathrm{PR}(G))$ is included in the language defined by a PR grammar, called *free*, (the free PR grammars are the counterpart of the free OP grammars [10]) which is obtained from the ANF grammar $\widetilde{G}$ by a straightforward transformation.

The *free grammar* associated to $\widetilde{G}$ is $\widehat{G} = (\Sigma, \widehat{V}_N, \widehat{P}, \widehat{S})$ where $\widehat{V}_N = A(G)$ and $\widehat{S} = \{\alpha \in A(G) \mid \langle A, \alpha \rangle \in \widetilde{S}\}$.

To define the rules of the free grammar, we need a projection. The projection $h : \widetilde{V}_N \cup \widetilde{\Sigma} \to \widehat{V}_N \cup \Sigma$ is defined as $h(\langle X, \alpha \rangle) = \alpha$, $X \in V_N$, i.e., the nonterminals that have identical attributes are coalesced, and $h(\langle b, \alpha \rangle) = b$, $b \in \Sigma$, i.e., it erases the (leaf) attributes of type 1 thus making the terminal alphabet identical to Σ. The rule set is $\widehat{P} = h(\widetilde{P})$.

Example 5 (Free grammar vs ANF grammar). *The grammar* $G = \{S \to aA,\ A \to aB,\ B \to a\}$ *has the ANF below.*

ANF grammar $\widetilde{G}$	free grammar $\widehat{G}$
$S[\#\dot{y}a, a\dot{t}\#] \to a(\#\dot{y}a, a\dot{y}a)\, A[a\dot{y}a, a\dot{t}\#]$	$[\#\dot{y}a, a\dot{t}\#] \to a\,[a\dot{y}a, a\dot{t}\#]$
$A[a\dot{y}a, a\dot{t}\#] \to a(a\dot{y}a, a\dot{y}a)\, B[a\dot{y}a, a\dot{t}\#]$	$[a\dot{y}a, a\dot{t}\#] \to a\,[a\dot{y}a, a\dot{t}\#]$
$B[a\dot{y}a, a\dot{t}\#] \to a(a\dot{y}a, a\dot{t}\#)$	$[a\dot{y}a, a\dot{t}\#] \to a$

Notice that $L(G) = a^3$ *and* $L(\widehat{G}) = a^+ a$.

The next statement is immediate.

Proposition 2. *The language inclusion relations hold:* $L(G) \subseteq L(\mathrm{PR}(G)) \subseteq L(\widehat{G})$.

In fact, the $\mathrm{APR}(G)$ used by the parser at each reduction step are included into, but not necessarily saturate the APRs of the free grammar. Therefore the parser may reject some strings in $L(\widehat{G})$.

Union   It is natural to consider as compatible two PR grammars such that their APRs do not clash, in the sense that their union does not contain a conflict. For compatible grammars closure under union is immediate.

Theorem 1 (Closure under union). *Let* G' *and* G'' *be PR grammars such that the set* $\mathrm{PR}(G') \cup \mathrm{PR}(G'')$ *is conflict-free. The language* $L(G') \cup L(G'')$ *has the PR property.*

Proof. Assume for simplicity and without loss of generality that the grammars are free from copy rules. Let $V'_N \cap V''_N = \emptyset$, and define G as $(\Sigma, V'_N \cup V''_N, P' \cup P'', S' \cup S'')$ so that $L(G) = L(G') \cup L(G'')$. Since $\mathrm{PR}(G) = \mathrm{PR}(G') \cup \mathrm{PR}(G'')$ is conflict-free, grammar G is PR. $\square$

We observe that grammar G may be ambiguous in a mild way that does not prevent deterministic parsing. More precisely a string $x \in L(G)$ may have two structurally identical syntax trees that only differ in the nonterminal labels. In that case the parser will compute only one syntax tree with two labels per internal node.

4. Comparison with Operator-Precedence languages

Our objective was to define a novel type of precedence relation and a corresponding family of languages that include the OPL yet do not require the operator form of grammars. We show that every OP grammar has the PRP.

We recall that on operator-precedence grammars three precedence relations, for brevity OPR, are defined over $(\Sigma \cup \#)^2$: $a \lessdot b$, $a \gtrdot b$ and $a \doteq b$, resp. named yield, take, equal in precedence. Notice that such relations are defined between two terminals that are adjacent, or also separated by a nonterminal, in a sentential form (antichain). Although conceptually similar, our APR definitions differ in three ways: they are defined on attributes, they include the facing relation, and are only defined between two nodes that are adjacent in antichains. This requires a more analytical comparison of the effect of the no-conflict condition for OPR and APR, in the next lemma.

Lemma 1 (Operator-precedence grammars vs PR grammars). *Let G be an operator precedence grammar. Then G meets the PR property.*

Proof. Assuming the relations $\mathrm{OPR}(G) \subseteq (\Sigma \cup \#)^2$ to be conflict-free, we prove that the relations $\mathrm{APR}(G)$ are conflict-free. First, the obvious constraints imposed on APRs by the operator form are: neither facing relations nor relations between two internal tree nodes are possible. The remaining cases are: (1) relations $\dot{y}$, $\dot{t}$, $\dot{s}$ between two (adjacent) leaves, and (2) relations between a leaf and an internal node or conversely. For cases (1) and (2) we prove that a conflict in $\mathrm{APR}(G)$ implies a conflict in $\mathrm{OPR}(G)$.

Let the conflicting pairs be m', m'' and n', n'' with $\alpha(m') = \alpha(n'), \alpha(m'') = \alpha(n'')$.

1. Leaf-Leaf. From definitions, a terminal-terminal PR implies in OPR the corresponding relation through the correspondence: $\dot{y} \mapsto \lessdot, \dot{t} \mapsto \gtrdot, \dot{s} \mapsto \doteq$. We prove just the case for the conflict $m' \dot{y} m''$, $n' \dot{s} n''$ schematized below, the others being similar.
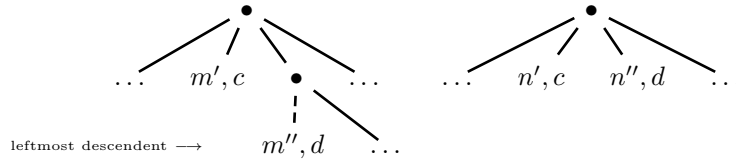
From $m' \dot{y} m''$, the attributes of the first pair have the form $\alpha(m') = (\ldots c, c \dot{y} d)$ and $\alpha(m'') = (c \dot{y} d, d \ldots)$. From $n' \dot{s} n''$, the attributes of the second pair have the form $\alpha(n') = (\ldots c, c \dot{s} d)$ and $\alpha(n'') = (c \dot{s} d, d \ldots)$, whence the OPG conflict $c \lessdot d$, $c \doteq d$.

2. Leaf-Internal (Internal-Leaf is similar and omitted). By symmetry it suffices to prove the case for the two pairs $m' \dot{y} m''$ and $n' \dot{s} n''$, resp. schematized in tree t, and in the trees t_1 and t_2.
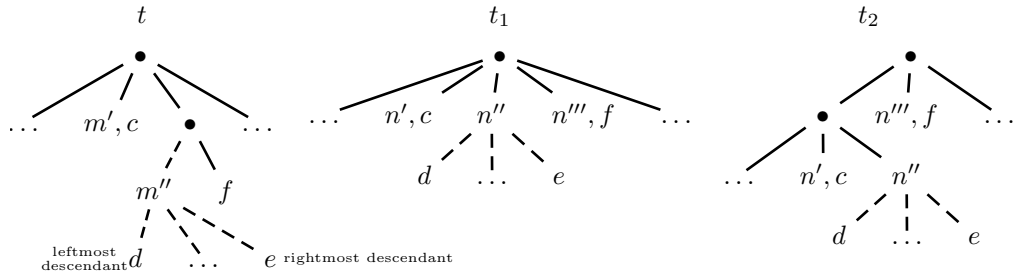
From $m' \dot{y} m''$ the attributes in tree t have the forms: $\alpha(m') = (\ldots c, c \dot{y} d)$ and $\alpha(m'') = [c \dot{y} d, e \dot{t} f]$. The corresponding OPRs are $c \lessdot d$, $e \gtrdot f$. In addition, relation $c \lessdot f$ holds since the nonterminal label of m'' is "transparent", by convention in OP. From $n' \dot{s} n''$, the attributes forms in trees t_1 and t_2 are $\alpha(n') = (\ldots c, c \dot{y} d)$ and $\alpha(n'') = [c \dot{y} d, e \dot{t} f]$. The possible subcases for n''', schematized in tree t_1 and t_2, differ in the respective APRs: $\alpha(n'') \dot{s} \alpha(n''')$ and $\alpha(n'') \dot{t} \alpha(n''')$. They imply respectively the OPRs $c \doteq f$ and $c \gtrdot f$, conflicting in both cases with the previous OPR $c \lessdot f$ of tree t. $\qquad\square$

Theorem 2. *The family of OP grammars is strictly included in the family of operator form grammars that have the PR property.*

Proof. The weak inclusion follows from Lemma 1. The strict inclusion is witnessed by the PR grammar $S \to X \mid Y,\ X \to Xs \mid s,\ Y \to AsY \mid As,\ A \to a$ that has the OP conflict $s \lessdot s, s \gtrdot s$. $\qquad\square$

Lemma 2. *The PR language $L = \{a^n c(aba)^n \mid n \geq 0\}$ of Ex. 3 is not defined by any OP grammar.*

Proof. From the pumping lemma for CF languages, any long enough sentence in L can be written as $uv^n xy^n z,\ n \geq 1$, where u, x, z are finite strings, $v = a$, and y is a cyclic permutation of aba. Therefore any (operator-form grammar) for L must produce a derivation of the form $Z \xrightarrow{+} aZy$ where the cases for y are: aba, aab, baa. This can be done in finitely many ways, each one can be proved to lead to OP conflicts. We just consider some cases, omitting the others, which are similar.

1. $y = aba$ and rule $S \to aSaba$. Conflicts: $a \lessdot a, a \gtrdot a, a \doteq a$.

2. $y = aab$ and rule $X \to aXaab$. Conflicts: $a \lessdot a, a \doteq a$. Similarly for case $y = baa$.

3. $y = aba$ and rules $S \to aXba,\ X \to Sa$, i.e., $S \xrightarrow{2} aSaba$. Conflicts: $a \doteq b, a \gtrdot b$.

$\qquad\square$

Theorem 3. *The family of OP languages is strictly included in the family of PR languages.*

At last, we discuss a drawback of OP rules that we have been able to cure using PR grammars. Consider a set of operators that are syntactically equivalent, e.g., in arithmetic expressions, the *multiply* and *divide*. In OP rules we cannot factor out both operators into a nonterminal $div\_mult \to * \mid /$, without creating a non-operator rule such as $F \to T\,div\_mult\,F$. We have experimentally found that such a transformation of typical OP grammars preserves the PRP. Notice that factoring transforms an OP grammar into the well-known generalized Chomsky normal form (GCNF), where only two types of rules are present: $B \to c$ with $c \in \Sigma$, and $B \to y$ with $y \in V_N^+$. In the GCNF, the only PR between terminals is $\dot{f}$. The question whether for any OP the corresponding GCNF has the PRP is open.

5. Conclusion

The idea of (sinthesized) attributes as the support of precedence relations is original and quite general, and its realization in Def. 2 is a satisfactory instantiation we arrived at after experimenting with other more complex definitions on a set of typical but small grammar examples. We intend to extend the experimentation to representative collections of grammars for artificial languages, to confirm the attribute definition or to improve it.

Several natural and interesting theoretical questions are ahead. The Boolean closure of the family of PR-compatible languages, if proved, would open the way to developing logical definitions of PR languages, in view of application to model checking of formal specifications.

Concerning syntactic properties, it would be interesting to extend PR definitions when regular expressions are permitted in grammar rules (as done for OP in [26]). Another range of open questions is which grammar transformations (e.g., into Greibach normal form) preserve the PR property.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] A. Barenghi, S. Crespi-Reghizzi, D. Mandrioli, F. Panella, M. Pradella, Parallel parsing made practical, Sci. Comput. Program. 112 (2015) 195–226. URL: https://doi.org/10.1016/j.scico.2015.09.002. doi:10.1016/J.SCICO.2015.09.002.

[2] L. Li, K. Taura, Associative operator precedence parsing: A method to increase data parsing parallelism, in: Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region, HPCAsia '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 75–87. URL: https://doi.org/10.1145/3578178.3578233. doi:10.1145/3578178.3578233.

[3] D. Jackson, Abstract model checking of infinite specifications, in: M. Naftalin, T. Denvir, M. Bertran (Eds.), FME '94: Industrial Benefit of Formal Methods, Springer Berlin Heidelberg, Berlin, Heidelberg, 1994, pp. 519–531.

[4] J. Esparza, D. Hansel, P. Rossmanith, S. Schwoon, Efficient Algorithms for Model Checking Pushdown Systems, in: Computer Aided Verification, volume 1855 of *LNCS*, 2000, pp. 232–247.

[5] M. Chiari, D. Mandrioli, M. Pradella, Operator precedence temporal logic and model checking, Theoretical Computer Science (2020). URL: https://doi.org/10.1016/j.tcs.2020.08.034. doi:10.1016/j.tcs.2020.08.034.

[6] M. Chiari, D. Mandrioli, M. Pradella, A first-order complete temporal logic for structured context-free languages, Logical Methods in Computer Science 18 (2022). URL: https://arxiv.org/abs/2105.10740.

[7] M. Chiari, D. Mandrioli, F. Pontiggia, M. Pradella, A model checker for operator precedence languages, ACM Trans. Program. Lang. Syst. 45 (2023) 19:1–19:66. URL: https://doi.org/10.1145/3608443. doi:10.1145/3608443.

[8] R. W. Floyd, Syntactic Analysis and Operator Precedence, J. ACM 10 (1963) 316–333.

[9] S. Crespi-Reghizzi, Reduction of enumeration in grammar acquisition, in: D. C. Cooper (Ed.), Proceedings of the 2nd International Joint Conference on Artificial Intelligence. London, UK, September 1-3, 1971, William Kaufmann, 1971, pp. 546–552. URL: http://ijcai.org/Proceedings/71/Papers/049.pdf.

[10] S. Crespi Reghizzi, D. Mandrioli, D. F. Martin, Algebraic Properties of Operator Precedence Languages, Information and Control 37 (1978) 115–133.

[11] S. Crespi Reghizzi, D. Mandrioli, Operator Precedence and the Visibly Pushdown Property, J. Comput. Syst. Sci. 78 (2012) 1837–1867. doi:10.1016/j.jcss.2011.12.006.

[12] V. Lonati, D. Mandrioli, F. Panella, M. Pradella, Operator precedence languages: Their automata-theoretic and logic characterization, SIAM J. Comput. 44 (2015) 1026–1088. doi:10.1137/140978818.

[13] D. Mandrioli, M. Pradella, S. Crespi Reghizzi, Aperiodicity, star-freeness, and first-order definability of structured context-free languages, Logical Methods in Computer Science 19 (2023). doi:10.46298/lmcs-19(4:12)2023.

[14] T. A. Henzinger, P. Kebis, N. Mazzocchi, N. E. Saraç, Regular methods for operator precedence languages, in: K. Etessami, U. Feige, G. Puppis (Eds.), 50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany, volume 261 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, pp. 129:1–129:20. doi:10.4230/LIPIcs.ICALP.2023.129.

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

[16] S. Crespi Reghizzi, M. Pradella, Beyond operator-precedence grammars and languages, Journal of Computer and System Sciences 113 (2020) 18–41. doi:10.1016/j.jcss.2020.04.006.

[17] N. Wirth, H. Weber, EULER: a generalization of ALGOL and it formal definition: Part 1, Commun. of the ACM 9 (1966) 13–25.

[18] L. Lizcano, E. Angulo, J. Márquez, Precedence table construction algorithm for CFGs regardless of being OPGs, Algorithms 17 (2024). URL: https://www.mdpi.com/1999-4893/17/8/345. doi:10.

3390/a17080345.

[19] D. E. Knuth, Semantics of context-free languages, Math. Syst. Theory 2 (1968) 127–145. doi:10.1007/BF01692511.

[20] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, M. Tommasi, Tree automata techniques and applications, https://hal.inria.fr/hal-03367725, 2008. URL: http://tata.gforge.inria.fr.

[21] N. Wirth, H. Weber, EULER: a generalization of ALGOL and it formal definition: Part 1, Commun. ACM 9 (1966) 13–25. URL: https://doi.org/10.1145/365153.365162. doi:10.1145/365153.365162.

[22] D. Grune, Parsing Techniques: A Practical Guide, 2nd ed., Springer Publishing Company, Incorporated, 2010.

[23] M. J. Fischer, Some properties of precedence languages, in: STOC '69: Proc. first annual ACM Symp. on Theory of Computing, ACM, New York, NY, USA, 1969, pp. 181–190.

[24] A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman, Compilers: principles, techniques, and tools, second ed., Pearson/Addison Wesley, 2007.

[25] R. McNaughton, S. Papert, Counter-free Automata, MIT Press, Cambridge, USA, 1971.

[26] M. Chiari, D. Mandrioli, M. Pradella, Cyclic operator precedence grammars for improved parallel parsing, in: DLT'24, volume 14791 of *LNCS*, Springer, 2024, pp. 98–113. doi:10.1007/978-3-031-66159-4_8.