# Free Grammars and Languages

Violetta Lonati[2], Dino Mandrioli[1], Federica Panella[1], Matteo Pradella[1]

[1] DEIB - Politecnico di Milano, via Ponzio 34/5, Milano, Italy
{dino.mandrioli, federica.panella, matteo.pradella}@polimi.it
[2] DI - Università degli Studi di Milano, via Comelico 39/41, Milano, Italy
lonati@di.unimi.it

Floyd's Operator Precedence languages (OPLs) were originally introduced to support deterministic parsing of programming languages [11]; then, interest in them decayed for several decades, probably due to the advent of more expressive grammars, such as LR ones which also allow for efficient deterministic parsing.

Recently, however, we renewed our interest in this family of languages on the basis of two major properties thereof: 1) their "locality property", i.e., the fact that partial strings can be parsed independently of the context in which they occur within a whole string; this enables more effective parallel and incremental parsing techniques than for other deterministic languages [3,4]; and 2) the fact that, to the best of our knowledge, OPLs are the largest family closed w.r.t. Boolean operations, concatenation, Kleene * and other classical operations [7]; furthermore they are recognized by a peculiar automata family [12] and are characterized in terms of classical monadic second order (MSO) logic [13]. This latter property entitles OPLs as a best candidate for extending the application of powerful verification techniques such as model-checking far beyond the original class of regular languages and even other recent families such as Visibly Pushdown Languages (VPLs) [2] which are strictly contained within OPLs too.

In this paper we introduce (more precisely, resume) a subclass of OPLs, namely free languages (FrLs) which were defined in [9,8] with the main motivation of grammar inference. FrLs constitute a kind of algebra within the structure defined by any given operator precedence matrix (OPM) [8]. Besides briefly outlining a "specialized" version of automata explicitly tailored for FrLs, we offer a new logic characterization in terms of first-order logic, as opposed to the traditional but more complex one in terms of MSO logic. FrLs however, lose some closure properties and have some "distinguishing" limits in terms of generative power, which nevertheless covers various relevant and heterogeneous cases: thus, they better lend themselves to act as "basic skeleton" for describing the structural part of a language to be complemented either with a suitable intersection with "regular control" or with additional restrictions formalized in terms of first-order properties conjuncted with the original ones associated with a grammar (FrG), somewhat in the spirit of the classical Chomsky-Schutzenberger characterization of CF languages.

## Basic definitions

The reader may find more details on OPGs in [7,8,11]. Let $\Sigma$ be an alphabet and $\varepsilon$ the empty string. Let $G = (N, \Sigma, P, S)$ be a *context-free* (CF) grammar, where $N$ is the nonterminal alphabet, $P$ the rule (or production) set, and $S$ the axiom. A rule is in *operator form* if its right hand side (r.h.s.) has no adjacent nonterminals; an *operator grammar*

(OG) contains only such rules. The following naming convention will be adopted, unless otherwise specified: lowercase Latin letters $a, b, \dots$ denote terminal characters; uppercase Latin letters $A, B, \dots$ denote nonterminal characters; letters $r, s, t, u, v, \dots$ denote terminal strings; and Greek letters $\alpha, \dots, \omega$ denote strings over $\Sigma \cup N$. The strings may be empty, unless stated otherwise.

For an OG $G$ and a nonterminal $A$, the *left and right terminal sets* are

$$\mathcal{L}_G(A) = \{a \in \Sigma \mid A \overset{*}{\Rightarrow} Ba\alpha\} \qquad \mathcal{R}_G(A) = \{a \in \Sigma \mid A \overset{*}{\Rightarrow} \alpha aB\}$$

where $B \in N \cup \{\varepsilon\}$ and $\overset{*}{\Rightarrow}$ denotes the derivation relation.
The following binary operator precedence (OP) relations are defined:

$$\begin{aligned} \text{equal in precedence: } & a \doteq b \iff \exists A \to \alpha aBb\beta, B \in N \cup \{\varepsilon\} \\ \text{takes precedence: } & a > b \iff \exists A \to \alpha Db\beta, D \in N \text{ and } a \in \mathcal{R}_G(D) \\ \text{yields precedence: } & a < b \iff \exists A \to \alpha aD\beta, D \in N \text{ and } b \in \mathcal{L}_G(D) \end{aligned}$$

The *operator precedence matrix* (OPM) $M = OPM(G)$ is a $|\Sigma| \times |\Sigma|$ array that associates with any ordered pair $(a, b)$ the set $M_{ab}$ of OP relations holding between $a$ and $b$. An OG $G$ is an *operator precedence* or *Floyd grammar* (*OPG*) if, and only if, $M = OPM(G)$ is a *conflict-free* matrix, i.e., $\forall a, b, |M_{ab}| \leq 1$.

A conflict-free OPM $M$ on an alphabet $\Sigma$ assigns a structure to strings in $\Sigma^*$, i.e., a string can be uniquely associated with a tree.

If $M_{a,b} = \circ$, where $\circ \in \{<, \doteq, >\}$, we write $a \circ b$. For $u, v \in \Sigma^*$ we write $u \circ v$ if $u = xa$ and $v = by$ with $a \circ b$. $M$ is *complete* if $M_{a,b}$ is defined for every $a$ and $b$ in $\Sigma$. Moreover in the following we assume that $M$ is $\doteq$*-acyclic*, which means that $c_1 \doteq c_2 \doteq \dots \doteq c_k \doteq c_1$ does not hold for any $c_1, c_2, \dots c_k \in \Sigma, k \geq 1$. See [8,7,15] for a discussion on this hypothesis.

A *simple chain* is a string $c_0 c_1 c_2 \dots c_\ell c_{\ell+1}$, written as ${}^{c_0}[c_1 c_2 \dots c_\ell]^{c_{\ell+1}}$, such that: $c_0, c_{\ell+1} \in \Sigma \cup \{\#\}$, $c_i \in \Sigma$ for every $i = 1, 2, \dots \ell$, $M_{c_0 c_{\ell+1}} \neq \emptyset$, and $c_0 < c_1 \doteq c_2 \dots c_{\ell-1} \doteq c_\ell > c_{\ell+1}$ (by convention the special symbol # yields precedence to all elements of $\Sigma$ and all elements of $\Sigma$ take precedence over it). A *composed chain* is a string $c_0 s_0 c_1 s_1 c_2 \dots c_\ell s_\ell c_{\ell+1}$, where ${}^{c_0}[c_1 c_2 \dots c_\ell]^{c_{\ell+1}}$ is a simple chain, and $s_i \in \Sigma^*$ is the empty string or is such that ${}^{c_i}[s_i]^{c_{i+1}}$ is a chain (simple or composed), for every $i = 0, 1, \dots, \ell$. Such a composed chain will be written as ${}^{c_0}[s_0 c_1 s_1 c_2 \dots c_\ell s_\ell]^{c_{\ell+1}}$. A string $s \in \Sigma^*$ is *compatible* with the OPM $M$ if ${}^{\#}[s]^{\#}$ is a chain.

Let $s$ be any word $\in \Sigma^*$. For $0 \leq x < y \leq |s| + 1$, we say that $(x, y)$ is a *chain boundary* iff there exists a sub-string of $\#s\#$ which is a chain ${}^{a}[r]^{b}$, such that $a$ is in position $x$ and $b$ is in position $y$. In general if $(x, y)$ is a chain boundary, then $y > x + 1$, and a position $x$ may be in such a relation with more than one position and vice versa. Moreover, if $s$ is compatible with $M$, then $(0, |s| + 1)$ is a chain boundary.

**Definition 1.** Free Grammar and Language
*First, the definition of left and right terminal sets of a grammar G is extended from N to $(\Sigma \cup N)^*$ in the following natural way:*

$$\mathcal{L}_G(\alpha) = \begin{cases} \textit{if } \alpha = a\beta, a \in \Sigma, \beta \in (\Sigma \cup N)^* \textit{ then } \{a\}, \\ \textit{else if } \alpha = Aa\beta, A \in N \textit{ then } \mathcal{L}_G(A) \cup \{a\} \end{cases}$$

*$\mathcal{R}_G(\alpha)$ is defined symmetrically.*

*Let $G$ be an OPG in the usual form such that: the axiom $S$ does not occur in the r.h.s. of any rule, no empty rule exists except possibly $S \to \varepsilon$, the other rules having $S$ as l.h.s are renaming, and no other renaming rules exist.*
*$G$ is a* free grammar *(FrG) iff a) for every production $A \to \alpha$, with $A \neq S$, $\mathcal{L}_G(A) = \mathcal{L}_G(\alpha)$ and $\mathcal{R}_G(A) = \mathcal{R}_G(\alpha)$ and b) for every nonterminals $A$, $B$, $\mathcal{L}_G(A) = \mathcal{L}_G(B)$ and $\mathcal{R}_G(A) = \mathcal{R}_G(B)$ implies $A = B$. A language generated by a FrG is a* free language *(FrL).*

Notice that, by definition, a free grammar is invertible (i.e., no two rules have identical r.h.s). Also, being $N \setminus \{S\}$ isomorphic to $\mathcal{P}(\Sigma) \times \mathcal{P}(\Sigma)$, it is customary to use $\mathcal{P}(\Sigma) \times \mathcal{P}(\Sigma) \cup \{S\}$ as the nonterminal alphabet of a free grammar. Given an OPM $M$, the *maxgrammar* associated with $M$ is the free grammar that contains all productions that are compatible with $M$. Thus, the set of free grammars with a given OPM is a lattice whose top element is the maxgrammar associated with the matrix [8].[3]

## First properties and examples

In this section we investigate the generative power of free grammars by means of some examples and comparisons with other classes of languages.

*Example 1.* The FrG depicted in Figure 1 generates unparenthesized arithmetic expressions with the usual precedences of $*$ w.r.t. $+$, which are not VPLs. This grammar is obtained from the maxgrammar by taking only those nonterminals that have $a$ in both left and right sets. By this way we guarantee that all strings generated by the grammar begin and end with an $a$. We did not include the copy rules $S \to A$, for each nonterminal $A \in \mathcal{P}(\Sigma) \times \mathcal{P}(\Sigma)$, for brevity.

| | $a$ | $+$ | $*$ |
|---|---|---|---|
| $a$ | | $\gtrdot$ | $\gtrdot$ |
| $+$ | $\lessdot$ | $\gtrdot$ | $\lessdot$ |
| $*$ | $\doteq$ | | |

$$\begin{aligned}
\langle\{a\},\{a\}\rangle &\to a\\
\langle\{+,a\},\{+,a\}\rangle &\to \langle\{+,a\},\{+,a\}\rangle \ + \ \langle\{*,a\},\{a\}\rangle\\
\langle\{+,*,a\},\{+,a\}\rangle &\to \langle\{+,*,a\},\{+,a\}\rangle \ + \ \langle\{*,a\},\{a\}\rangle\\
\langle\{*,a\},\{a\}\rangle &\to \langle\{a\},\{a\}\rangle \ * \ a\\
\langle\{*,a\},\{a\}\rangle &\to \langle\{*,a\},\{a\}\rangle \ * \ a\\
\langle\{+,a\},\{+,a\}\rangle &\to \langle\{a\},\{a\}\rangle \ + \ \langle\{*,a\},\{a\}\rangle\\
\langle\{+,*,a\},\{+,a\}\rangle &\to \langle\{+,*,a\},\{+,a\}\rangle \ + \ \langle\{a\},\{a\}\rangle\\
\langle\{+,*,a\},\{+,a\}\rangle &\to \langle\{*,a\},\{a\}\rangle \ + \ \langle\{a\},\{a\}\rangle\\
\langle\{+,a\},\{+,a\}\rangle &\to \langle\{a\},\{a\}\rangle \ + \ \langle\{a\},\{a\}\rangle\\
\langle\{+,*,a\},\{+,a\}\rangle &\to \langle\{*,a\},\{a\}\rangle \ + \ \langle\{*,a\},\{a\}\rangle\\
\langle\{+,a\},\{+,a\}\rangle &\to \langle\{+,a\},\{+,a\}\rangle \ + \ \langle\{a\},\{a\}\rangle
\end{aligned}$$

**Fig. 1.** A FrG for unparenthesized arithmetic expressions (right), and its OPM (left).

Extending the above grammar to generate also parenthesized arithmetic strings is an easy exercise by observing that all that we need are rules of the type $\langle\{(\!|\},\{|\!)\}\rangle \to (\!|X|\!)$, where $(\!|$ and $|\!)$ denote left and right parentheses, $X$ represents anyone of previous nonterminals, and the new nonterminal can occur in turn wherever $\langle\{a\},\{a\}\rangle$ occurs in the

[3] In [8] it is also shown that each free grammar is the top element of a Boolean algebra and that the whole family of OPLs compatible with a given OPM is itself a Boolean algebra whose top element is the language generated by the maxgrammar.

above grammar. Free grammars are also adequate to generate various other types of languages, which model sequences of operations handled in a classical LIFO policy which can be interrupted by high(er) priority interrupts, sequences of operations that manage documents and their updating, etc. On the other hand FrLs are noncounting [6]; thus, since regular languages can be counting, FrLs are incomparable with regular languages and VPLs.

The above example also illustrates a typical feature of FrGs: they are not intended to be built by hand; being driven by the powerset of $\Sigma$, both $N$ and $P$ suffer from combinatorial explosion. However, according to their original motivation to support grammar inference, they are well suited to be built by some automatic device.[4] Furthermore, their typical canonical form makes also easy the application of the classical minimization procedure that extends to structure grammars the minimization of finite state machines [14].

We envisage two major approaches to build FrGs generating a desired language: a bottom up one abstracts away from a given sample of language sentences as in traditional grammar inference (in this case it exploits the distinguishing property of FrGs that they can be inferred in the limit on the basis of a positive sample only [9]); a top-down one, instead, starts from the maxgrammar and "prunes" nonterminals and productions that would generate undesired sentences: this technique has been applied to build the grammar of the above examples. We will see that FrGs can be useful even when the language to define exceeds the limits of their generative power.

The next result concerns FrLs closure properties.

**Theorem 1.** *FrLs (with a fixed OPM) are closed w.r.t. intersection but, unlike general OPLs and VPLs, not w.r.t. complement, union, concatenation, and Kleene's *.*

FrLs can be associated with a natural and simple class of automata accepting them: intuitively, a free automaton (FrA) shifts all input characters onto the stack, then, as soon as a full r.h.s (of the FrG) is on top of the stack, and is recognized with the help of the OPM, it replaces the r.h.s. with the unique corresponding l.h.s., if any; otherwise the string is rejected. Without going into formal details we consider FrAs as "stateless" since they simply must push symbols onto the stack and verify whether and when a r.h.s. on its top is ready to be reduced.

## First-order characterization of free languages

Unlike general OPLs, VPLs and many other language families that require a typical MSO logic characterization [2,13], FrLs can be characterized by means of a first-order logic (FOL). The key idea to achieve such a simplified formalization derives from the "stateless nature" of FrLs. In fact, MSO formulas characterizing "normal" languages are quantified w.r.t. sets of positions corresponding to the states entered by the accepting automaton, such states being a qualifying feature of each automaton. In the case of FrLs, instead there are no states and the stack alphabet is fixed a priori (it is essentially $\Sigma \cup \mathcal{P}(\Sigma) \times \mathcal{P}(\Sigma)$); thus, each position of the string can be associated with elements that

[4] In fact the grammar of the example and a few others have been produced by the prototype tool available at *http://home.deib.polimi.it/pradella*.

belong to a finite set of possible choices: this allows to avoid quantification w.r.t. to sets of positions. Next, we illustrate the key points of the construction.

**Theorem 2.** *Let G be a FrG: then a FO formula $\psi_G(m)$ can be effectively built such that $w \in L(G)$ iff $w \models \psi_G(|w|)$.*

As customary, first-order variables are interpreted over positions of the string; $a(x)$ is true iff the character in position $x$ is $a$; the other logical symbols have the usual meaning. Moreover, we introduce the predicate $\curvearrowright$: $x \curvearrowright y$ is true iff $(x, y)$ is a chain boundary. Figure 2a illustrates the intuitive meaning of the relation: $x \curvearrowright y$ means that $x + 1$ and $y - 1$, respectively, are the positions of the leftmost and rightmost leaves of the subtree with root labeled as $H$ in the figure.

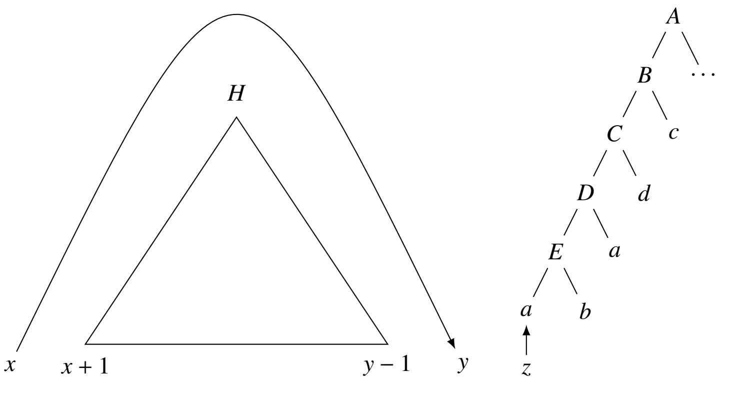

**Fig. 2.**

W.r.t. MSO syntax on the one side we drop second-order variables and, on the other side, for every subset $S \subseteq \Sigma$ we add monadic predicates $L\_S(x)$ and $R\_S(x)$: intuitively, $L\_S(x)$ will hold iff, in the leftmost path of a syntax tree going from the leaf at position $x$ towards the root there exists a (nonterminal) node whose $L$-set contains exactly the elements in $S$; for instance, in the case of Figure 2b $L\_\{a\}(z), L\_\{a, d\}(z), L\_\{a, c, d\}(z)$ hold. Then, the construction provides axioms such that, if $x \curvearrowright y$, then $L\_L(H)(x + 1)$ and $R\_R(H)(y - 1)$ hold. For instance, for all terminal rules of the type $\rho = \langle\{c_1\}, \{c_k\}\rangle \to c_1 c_2 \ldots c_k$ we build the axioms

$$\varphi_1^\rho := \forall x \left( \begin{array}{c} x \curvearrowright x + k + 1 \wedge \\ c_1(x + 1) \wedge c_2(x + 2) \wedge \ldots \wedge c_k(x + k) \\ \Rightarrow \\ L\_\{c_1\}(x + 1) \wedge R\_\{c_k\}(x + k) \end{array} \right)$$

and

$$\varphi_2^\rho := \forall x \left( \begin{array}{c} x \curvearrowright x + k + 1 \wedge L\_\{c_1\}(x + 1) \wedge R\_\{c_k\}(x + k) \\ \Rightarrow \\ c_1(x + 1) \wedge c_2(x + 2) \wedge \ldots \wedge c_k(x + k) \end{array} \right)$$

We build similar axioms $\varphi_3^\rho$ and $\varphi_4^\rho$ for all rules of the type $\rho = \langle L, R\rangle \to \langle L_0, R_0\rangle c_1 \langle L_1, R_1\rangle c_2 \langle L_2, R_2\rangle c_3 \ldots \langle L_k, R_k\rangle c_k \langle L_{k+1}, R_{k+1}\rangle$, with $\langle L_i, R_i\rangle \in \mathcal{P}(\Sigma) \times \mathcal{P}(\Sigma) \cup \{\varepsilon\}$, $0 \leq i \leq$

$k + 1$, and $L = L_0 \cup \{c_1\}$, $R = R_{k+1} \cup \{c_k\}$ (when $\langle L_i, R_i\rangle = \{\varepsilon\}$, $L_i$ and $R_i$ are empty) and we require also that $\bigwedge_{\langle L,R\rangle} \varphi_{\langle L,R\rangle}$ holds, where $\varphi_{\langle L,R\rangle} := \bigvee_{\rho=\langle L,R\rangle\to\alpha} \varphi_1^\rho \wedge \varphi_2^\rho \wedge \varphi_3^\rho \wedge \varphi_4^\rho$.

The converse of Theorem 2 does not hold: in fact, by means of FO formulas we can define also counting languages.

**Conclusions**

Recently, the old-fashioned OPGs somewhat surprisingly generated renewed interest and potential application in the context of novel technologies such as parallel compilation and model-checking. In our long term path aiming at exploiting their properties we resumed free grammars, which were originally introduced to support automatic grammar inference. The main result of this paper, i.e., the characterization of FrLs in terms of FO logic, as opposed to the more general and traditional MSO one, could represent a first step towards extending to OPLs and their subfamilies other classic results on various types of logic characterization (FO, tree logic [1], LTL,. . . ) of various language families (star-free regular languages [10,5], VPLs,. . . ).

## References

1. L. Afanasiev, P. Blackburn, I. Dimitriou, B. Gaiffe, E. Goris, M.J. Marx, and M. de Rijke. PDL for ordered trees. *Journal of Applied Non-Classical Logic*, 15(2):115–135, 2005.
2. R. Alur and P. Madhusudan. Adding nesting structure to words. *Journ. ACM*, 56(3), 2009.
3. A. Barenghi, S. Crespi Reghizzi, D. Mandrioli, and M. Pradella. Parallel parsing of operator precedence grammars. *Information Processing Letters*, 2013. DOI:10.1016/j.ipl.2013.01.008.
4. A. Barenghi, E. Viviani, S. Crespi Reghizzi, D. Mandrioli, and M. Pradella. PAPAGENO: a parallel parser generator for operator precedence grammars. In *5th International Conference on Software Language Engineering (SLE)*, 2012.
5. C Choffrut, A Malcher, C Mereghetti, and B Palano. First-order logics: some characterizations and closure properties. *Acta Inf.*, 49(4):225–248, 2012.
6. S. Crespi Reghizzi and D. Mandrioli. A class of grammar generating non-counting languages. *Inf. Process. Lett.*, 7(1):24–26, 1978.
7. S. Crespi Reghizzi and D. Mandrioli. Operator Precedence and the Visibly Pushdown Property. *Journal of Computer and System Science*, 78(6):1837–1867, 2012.
8. S. Crespi Reghizzi, D. Mandrioli, and D. F. Martin. Algebraic Properties of Operator Precedence Languages. *Information and Control*, 37(2):115–133, May 1978.
9. S. Crespi Reghizzi, M. A. Melkanoff, and L. Lichten. The Use of Grammatical Inference for Designing Programming Languages. *Commununications of the ACM*, 16(2):83–90, 1973.
10. V. Diekert and P. Gastin. First-order definable languages. *Logic and Automata: History and Perspectives*, 2:261, 2008.
11. R. W. Floyd. Syntactic Analysis and Operator Precedence. *Journ. ACM*, 10(3), 1963.
12. V. Lonati, D. Mandrioli, and M. Pradella. Precedence Automata and Languages. In *CSR*, volume 6651 of *LNCS*, pages 291–304. 2011.
13. V. Lonati, D. Mandrioli, and M. Pradella. Logic Characterization of Invisibly Structured Languages: the Case of Floyd Languages. In *SOFSEM*, volume 7741 of *LNCS*, pages 307–318. Springer, 2013.
14. R. McNaughton. Parenthesis Grammars. *Journ. ACM*, 14(3):490–500, 1967.
15. F. Panella, M. Pradella, V. Lonati, and D. Mandrioli. Operator precedence $\omega$-languages. *CoRR*, abs/1301.2476, 2013. http://arxiv.org/abs/1301.2476.