
Context-Free Grammars & Context-Free Languages

Martin Fränze

Informatics and Mathematical Modelling
The Technical University of Denmark

02140 Languages and Parsing, MF, Fall 2004 – p.1/21

What you'll learn

1. Context-free grammars:

- Structure
- Interpretation (multiple equivalent ones)
 - Recursive inference
 - Reductions
 - Derivations & standard derivations
 - Parse trees
- Proof of equivalence

2. The relation between regular and context-free languages:

- context-free languages are richer (strictly)

02140 Languages and Parsing, MF, Fall 2004 – p.2/21

Regular languages are too restrictive

The examples concerning the pumping lemma for regular languages show that e.g.

- palindromes (“madamimadam”),
 - $\{0^n 1^n\}$,
 - the set of arithmetic expressions with *correctly paired brackets*
- are not regular.

Need some (limited) counting mechanism!

How to construct palindromes

All palindromes over $\{0, 1\}$ can be constructed by the following rules:

1. ε is a palindrome. $P \rightarrow \varepsilon$
2. 0 is a palindrome. $P \rightarrow 0$
3. 1 is a palindrome. $P \rightarrow 1$
4. If w is a palindrome then so is $0w0$. $P \rightarrow 0P0$
5. If w is a palindrome then so is $1w1$. $P \rightarrow 1P1$

The reddish things are called “production rules”.

Context-free grammars

A **context-free grammar (CFG)** consists of

1. a finite set V of *variables* (a.k.a. *nonterminals* or *nonterminal symbols*), each of which represents a language over
2. a finite set T of *terminals* (or *terminal symbols*), which provides the alphabet of the language defined by the grammar,
3. a finite set P of *productions* (a.k.a. *rules*), that provide the recursive definitions of the languages associated to the elements of V ,
4. a designated *start symbol* (a.k.a. *axiom*) $S \in V$, that declares which of the $|V|$ languages defined by the grammar is the one we are actually interested in.

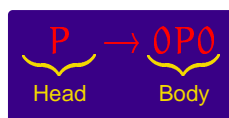
Thus, a CFG G is a 4-tuple $G = (V, T, P, S)$ formed of its variables V , terminals T , productions P , and start symbol S .

02140 Languages and Parsing, MF, Fall 2004 – p.5/21

Productions

A **production** consists of

1. A variable $A \in V$ that is (partially) defined by the production, called the *head of the production*,
2. the *production symbol* \rightarrow (or sometimes $::=$),
3. a string α over $V \cup T$, called the *body of the production*.
The body represents *one possible* way of building strings belonging to the language of the head variable A : We obtain a word of the language of A by replacing any variable A' occurring in α by a word (over T) known to be in the language of A' .



02140 Languages and Parsing, MF, Fall 2004 – p.6/21

Recursive inference

A word w belongs to the language of a CFG $G = (V, T, P, S)$ iff it can be *inferred to belong to the language of S* by the following **recursive inference** rules:

1. If P contains a production $A \rightarrow v$ with $v \in T^*$ then we can infer that v is in the language of A **(base inference)**
2. If we have inferred $w_1, \dots, w_n \in T^*$ to be in the languages of A_1 to A_n and P contains a production $A \rightarrow v_1 A_1 v_2 A_2 \dots v_n A_n v_{n+1}$ with $v_i \in T^*$ then we can infer that $v_1 w_1 v_2 w_2 \dots v_n w_n v_{n+1}$ is in the language of A **(recursive inference)**

Notation:

- The **language** $L(G)$ of a CFG $G = (V, T, P, S)$ is $\{w \in T^* \mid w \text{ can be inferred to be in } G\}$.
- For $A \in V$, $L_G(A)$ is $L(G')$ with $G' \stackrel{\text{def}}{=} (V, T, P, A)$.

02140 Languages and Parsing, MF, Fall 2004 – p.7/21

Reductions

A word $w \in T^*$ belongs to the language of a CFG $G = (V, T, P, S)$ iff it can be *reduced to S in a finite number of steps* by the following **reduction** rule:

- If P contains a production $B \rightarrow \beta$ (with $\beta \in (V \cup T)^*$) then $\alpha\beta\gamma \in (V \cup T)^*$ can be reduced to $\alpha B\gamma$.

Notation:

- If $\alpha \in (V \cup T)^*$ can be reduced to S in a finite number of steps then α is called a **sentential form of G** .
- A sentential form w with $w \in T^*$ is called a **sentence of G** .

02140 Languages and Parsing, MF, Fall 2004 – p.8/21

Derivations

A word $w \in T^*$ belongs to the language of a CFG $G = (V, T, P, S)$ iff it can be *derived from S in a finite number of steps* by the following **derivation** rule:

- If P contains a production $B \rightarrow \beta$ then $\alpha B \gamma \in (V \cup T)^*$ gives rise to the derivation $\alpha \beta \gamma$.

Notation:

- If β is a derivation of α (wrt. G) then we write $\alpha \Rightarrow_G \beta$.
- If β can be derived from α in a finite number (maybe 0) of steps, then we write $\alpha \Rightarrow_G^* \beta$.
- If $S \Rightarrow_G \alpha$ then α is called a **sentential form of G** .
- A sentential form w with $w \in T^*$ is called a **sentence of G** .

Leftmost derivations

A word $w \in T^*$ belongs to the language of a CFG $G = (V, T, P, S)$ iff it can be *derived through leftmost derivation from S in a finite number of steps* by the following **leftmost derivation** rule:

- If P contains a production $B \rightarrow \beta$ and $\alpha \in T^*$ then $\alpha B \gamma \in (V \cup T)^*$ gives rise to the *leftmost* derivation $\alpha \beta \gamma$.

Notation:

- If β is a leftmost derivation of α (wrt. G) then we write $\alpha \xRightarrow{lm} \beta$.
- If $S \xRightarrow{lm} \alpha$ then α is called a **left sentential form of G** .

Rightmost derivations

A word $w \in T^*$ belongs to the language of a CFG $G = (V, T, P, S)$ iff it can be *derived through rightmost derivation from S in a finite number of steps* by the following **rightmost derivation** rule:

- If P contains a production $B \rightarrow \beta$ and $\gamma \in T^*$ then $\alpha B \gamma \in (V \cup T)^*$ gives rise to the *rightmost derivation* $\alpha \beta \gamma$.

Notation:

- If β is a rightmost derivation of α (wrt. G) then we write $\alpha \xRightarrow{rm} \beta$.
- If $S \xRightarrow{rm} \alpha$ then α is called a **right sentential form of G** .

Parse trees

A **parse tree** for a CFG $G = (V, T, P, S)$ is a tree satisfying

1. each **leaf** is labeled by either a variable $A \in V$, a terminal $t \in T$, or by ε ;
2. each **interior node** is labeled by a variable $A \in V$;
3. if a node is labeled A and has **sons** X_1 to X_n (from left to right) then $A \rightarrow X_1 \dots X_n \in P$, where all $X_i \neq \varepsilon$ if $n > 1$.

Yields of parse trees

The **yield of a parse tree** is the word $\alpha \in (V \cup T)^*$ obtained by concatenating all leaves of the parse tree from left to right.

A word $w \in T^*$ belongs to the language of a CFG $G = (V, T, P, S)$ iff w is the **yield of a parse tree for G with root S** .

Why termed context-free?

The context-free grammars are termed “context-free” because reductions/derivations don’t take account of the context:

$$\beta \Rightarrow_G \delta \text{ iff } \alpha\beta\gamma \Rightarrow_G \alpha\delta\gamma$$

Context-free languages

Def: The context-free languages are exactly those languages that can be defined by a context-free grammar.

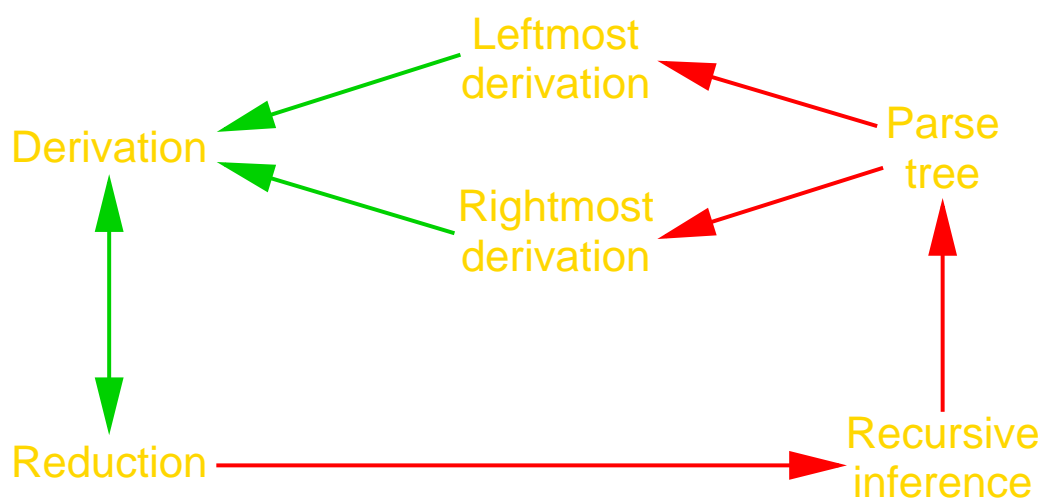
Qu.: What is “the language” defined by a CFG, given that we have multiple mechanisms for using CFGs, namely recursive inference, reductions, (restricted) derivations, parse trees?

Thm: Given a CFG G , the languages defined by G via

1. recursive inference,
2. reductions,
3. derivations,
4. leftmost derivations,
5. rightmost derivations,
6. yields of parse trees coincide.

02140 Languages and Parsing, MF, Fall 2004 – p.15/21

Structure of proof



02140 Languages and Parsing, MF, Fall 2004 – p.16/21

Formal statements: Parse tree to leftm. der.

Thm: Let $G = (V, T, P, S)$ be a CFG, $A \in V$, and $w \in T^*$.

If there is a parse tree with root A and yield w then $A \xRightarrow{*}_G w$.

N.B.: Above statement holds for $w \in T^*$. It is in general not true for arbitrary yields of parse trees.

Prf: We perform an induction on the height of the parse tree.

Formal statements: Rec. inf. to parse tree

Thm: Let $G = (V, T, P, S)$ be a CFG, $A \in V$, and $w \in T^*$.

If there is a recursive inference showing that w is in the language of A then there is a parse tree with root A and yield w .

Prf: We perform an induction on the number of steps in the recursive inference.

Formal statements: Reductions to rec. inf.

Thm: Let $G = (V, T, P, S)$ be a CFG, $A \in V$, and $w \in T^*$.

If there is a reduction sequence reducing w to A (i.e., $w \rightsquigarrow_G^* A$) then there is a recursive inference that determines w to be in the language of A .

N.B.: Above statement applies only to $w \in T^*$. It can't be generalized because the recursive inference procedure starts its recursion with productions that have terminal body, i.e. can infer properties of terminal strings only.

Prf: We perform an induction on the length of the reduction sequence.

Regular implies context-free

Thm: If L is regular then L is context-free (= definable by a CFG).

Prf: (Sketch) We map REs to CFGs: Given RE E , we take a variable $[F]$ for each subexpression F of E and generate productions $P_{[F]}$ for $[F]$ such that $L_G([F]) = L(F)$.

$$\begin{aligned}
 \emptyset &\rightsquigarrow \text{no production} \\
 \varepsilon &\rightsquigarrow [\varepsilon] \rightarrow \varepsilon \\
 a &\rightsquigarrow [a] \rightarrow a \\
 F_1 + F_2 &\rightsquigarrow \begin{cases} [F_1 + F_2] \rightarrow [F_1] \\ [F_1 + F_2] \rightarrow [F_2] \end{cases} \\
 F_1 F_2 &\rightsquigarrow [F_1 F_2] \rightarrow [F_1][F_2] \\
 F^* &\rightsquigarrow \begin{cases} [F^*] \rightarrow \varepsilon \\ [F^*] \rightarrow [F][F^*] \end{cases}
 \end{aligned}$$

Take the CFG $G = (\underbrace{\{[F] \mid F \text{ subexp. of } E\}}_V, \underbrace{\Sigma_E}_T, \underbrace{\bigcup P_{[F]}}_P, \underbrace{[E]}_S)$.

Context-free does not imply regular

As the palindromes are context-free, yet not regular, there is a context-free language that is not regular (in fact, there are infinitely many).

Hence:

Thm: The context-free languages are a strict superset of the regular languages.