

Grammars and Context Free Languages

H. Geuvers and A. Kissinger

Institute for Computing and Information Sciences
Radboud University Nijmegen

Version: fall 2015

Outline

Grammars

Regular Grammars

Parsing (extra material)

Pumping Lemma for context-free languages (extra material)

Generating words

Example of a **grammar** G :

$$\begin{array}{lcl} S & \rightarrow & O \mid E \\ E & \rightarrow & \lambda \mid aEa \mid bEb \\ O & \rightarrow & a \mid b \mid aOa \mid bOb \end{array}$$

Productions, always start with S

$$S \Rightarrow E \Rightarrow aEa \Rightarrow abEba \Rightarrow abba$$

$$S \Rightarrow E \Rightarrow bEb \Rightarrow baEab \Rightarrow babEbab \Rightarrow babaEabab \Rightarrow babaabab$$

$$S \Rightarrow O \Rightarrow bOb \Rightarrow bab$$

$$S \Rightarrow O \Rightarrow bOb \Rightarrow baOab \Rightarrow babab$$

We can generate exactly the set of words w with $w = w^R$
(w is a palindrome)

Context-free grammar

Let Σ be a finite alphabet.

Definition A **context-free grammar** (CFG) $G = \langle V, S, P \rangle$ over Σ consists of

- V , a set of **non-terminal symbols**,
- $S \in V$, a **start symbol**,
- P , a set of **production rules** of the form

$$X \rightarrow w,$$

where $X \in V$ and $w \in (V \cup \Sigma)^*$.

Notation:

$$E \rightarrow \lambda \mid aEa \mid bEb$$

is shorthand for three rules:

$$E \rightarrow \lambda, \quad E \rightarrow aEa, \quad E \rightarrow bEb.$$

Context-free language

Using G , a language is generated using a relation $\Rightarrow$ ('produces') defined as follows (where u, v, w are arbitrary elements of $(\Sigma \cup V)^*$)

$$\begin{aligned} X \rightarrow w \quad \text{implies} \quad uXv &\Rightarrow uwv \\ u \Rightarrow v, v \Rightarrow w \quad \text{implies} \quad u &\Rightarrow w \end{aligned}$$

Definition The language generated by G , is

$$\mathcal{L}(G) = \{w \in \Sigma^* \mid S \Rightarrow w\}.$$

A language L is context-free if $L = \mathcal{L}(G)$ for some context-free grammar G .

Non-terminal symbols have a meaning

When we write down a grammar, each non-terminal symbol has a **meaning**.

This helps in **explaining** the grammar and in **proving** that it satisfies a property.

Back to our first example grammar G :

$$S \rightarrow O \mid E$$

$$E \rightarrow \lambda \mid aEa \mid bEb$$

$$O \rightarrow a \mid b \mid aOa \mid bOb$$

- Words produced by E are **palindromes of even length**,
- Words produced by O are **palindromes of odd length**,
- Words produced by S are palindromes of even or odd length.
- So: $\mathcal{L}(G) \subseteq \{w \in \{a, b\}^* \mid w = w^R\}$.

Leftmost derivations

There can be many ways to produce a word $w \in \mathcal{L}(G)$.

$$G: \boxed{S \rightarrow aSB \mid \lambda, \quad B \rightarrow b}$$

Derivations of $aabb$:

$$d_1 : S \Rightarrow aSB \Rightarrow aaSBB \Rightarrow aaBB \Rightarrow aabB \Rightarrow aabb$$

$$d_2 : S \Rightarrow aSB \Rightarrow aSb \Rightarrow aaSBb \Rightarrow aaSbb \Rightarrow aabb$$

$$d_3 : S \Rightarrow aSB \Rightarrow aSb \Rightarrow aaSBb \Rightarrow aaBb \Rightarrow aabb$$

Derivation d_1 is leftmost, d_2 is rightmost, d_3 is neither.

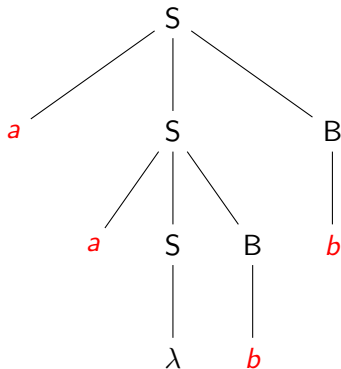
Definition A derivation is **leftmost** if in each step a rule is applied to the leftmost non-terminal.

Lemma For all grammars G and words w , $w \in \mathcal{L}(G)$ iff there is a leftmost derivation of w .

Parse trees / Derivation trees

G: $S \rightarrow aSB \mid \lambda, \quad B \rightarrow b$

A **parse tree** (or **derivation tree**) for $abbb$ is:



Ambiguity

Definition A context-free grammar G is **unambiguous** if for each $w \in \mathcal{L}(G)$ there exists a unique leftmost derivation of w in G . Otherwise G is **ambiguous**.

Equivalently: A context-free grammar is ambiguous if there is a word with two different parse trees.

Example “dangling else”

S	$\rightarrow$	print E
		if E then S else S
		if E then S
E	$\rightarrow$	$a \mid b \mid \dots$

Some context-free languages

Is there a grammar G_2 such that

$$\mathcal{L}(G_2) = \{a^n b^n \mid n \geq 0\}?$$

$$G_2: \boxed{S \rightarrow \lambda \mid aSb}$$

All possible derivations

$$\begin{array}{ccccccc}
 S & \Rightarrow & aSb & \Rightarrow & aaSbb & \Rightarrow & \dots \Rightarrow a^n Sb^n \Rightarrow \dots \\
 \Downarrow & & \Downarrow & & \Downarrow & & \Downarrow \\
 \lambda & & ab & & aabb & & \dots \Rightarrow a^n b^n
 \end{array}$$

What is a grammar G_3 such that

$$\mathcal{L}(G_3) = \{a^n b^n \mid n > 0\}?$$

$$G_3: \boxed{S \rightarrow ab \mid aSb}$$

More context-free languages

Let $\Sigma = \{a, b, c\}$

Claim: $L = \{a^n b^m c^{2n+1} \mid m, n \geq 0\}$ is context-free.

Let us first show that $L' = \{a^n c^{2n+1} \mid m, n \geq 0\}$ is context-free.

For L' use

$S \rightarrow c \mid aScc$

For L use

$S \rightarrow Bc \mid aScc$
$B \rightarrow \lambda \mid bB$

Fact $\{a^n b^n c^n \mid n \geq 0\}$ is *not* context-free.

This can be proved using the [Pumping Lemma for Context Free Languages](#)

Regular languages

Theorem Every regular language is context-free.

Proof: Let $M = \langle Q, q_0, \delta, F \rangle$ be an NFA (or DFA, or NFA_λ) that accepts $L \subseteq \Sigma^*$. Define a context-free grammar G_M as follows:

$V = Q$ non-terminals are states

$S = q_0$

$P = \{q \rightarrow aq' \mid q' \in \delta(q, a)\} \cup \{q \rightarrow \lambda \mid q \in F\}$

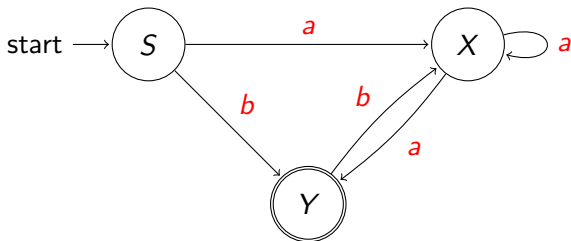
$\mathcal{L}(G_M) = L(M)$, since there is a 1-1 correspondence between computations and derivations:

$$M : q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n \in F$$

$$G_M : q_0 \Rightarrow a_1 q_1 \Rightarrow a_1 a_2 q_2 \Rightarrow \cdots \Rightarrow a_1 \cdots a_n q_n \Rightarrow a_1 \cdots a_n \lambda$$



Regular languages: example



Corresponding grammar:

S	$\rightarrow$	$aX \mid bY$
X	$\rightarrow$	$aX \mid aY$
Y	$\rightarrow$	$bX \mid \lambda$

Observe: The context free grammars we construct from an automaton are of a specific (simple) form.

Regular Grammars

Definition A **regular grammar** is a context-free grammar $G = \langle V, S, P \rangle$ in which all rules have the form

$$X \rightarrow aY \quad \text{or} \quad X \rightarrow \lambda$$

where $X, Y \in V$ and $a \in \Sigma$.

Example Consider the regular language $L := \mathcal{L}(a(aa + b)^*(bb)^*)$. A regular grammar for this language is found by first also allowing rules $X \rightarrow wY$ for $w \in \Sigma^*$, and then refining.

$$\begin{array}{ll} S & \rightarrow aX \mid aY \\ X & \rightarrow aaX \mid bX \mid aaY \mid bY \\ Y & \rightarrow bbY \mid \lambda \end{array}$$

$$\begin{array}{ll} S & \rightarrow aX \mid aY \\ X & \rightarrow aU \mid bX \mid bY \\ U & \rightarrow aX \mid aY \\ Y & \rightarrow bV \mid \lambda \\ V & \rightarrow bY \end{array}$$

Regular Languages and Regular Grammars

Theorem Let $L \subseteq \Sigma^*$.

L is regular $\iff L = \mathcal{L}(G)$ for some regular grammar G .

Proof:

For $\implies$: From an NFA accepting L , we define a regular grammar accepting L . (As shown before.)

For $\impliedby$: Given G , build an NFA $M_G = \langle Q, \delta, q_0, F \rangle$ as follows:

- State set $Q = V$, $q_0 = S$,
- $F = \{X \in V \mid (X \rightarrow \lambda) \in P\}$.
- $\delta(X, a) = \{Y \in V \mid (X \rightarrow a Y) \in P\}$

$\mathcal{L}(M_G) = L(G)$, since derivations correspond to computations:

$$G : S \Rightarrow a_1 X_1 \Rightarrow a_1 a_2 X_2 \Rightarrow \cdots \Rightarrow a_1 \cdots a_n X_n \Rightarrow a_1 \cdots a_n \lambda$$

$$M_G : S \xrightarrow{a_1} X_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} X_n \in F$$

Parsing

A **parser** is an algorithm that determines for given G and w whether $w \in \mathcal{L}(G)$?

In practice we also want the parse tree.

The well-known **Cocke-Younger-Kasami** (CYK) algorithm requires G to be in **Chomsky normal form**.

For simplicity we assume $\lambda \notin \mathcal{L}(G)$.

Definition A CFG G is in Chomsky normal form if all its productions are of the form:

$$X \rightarrow YZ \quad \text{or} \quad X \rightarrow a$$

where $X, Y, Z \in V$ and $a \in \Sigma$

Lemma Every CFG G (with $\lambda \notin \mathcal{L}(G)$) can be transformed into an equivalent CFG in Chomsky normal form.

Chomsky normal form and the CYK Algorithm

Example of a grammar in Chomsky normal form:

$S \rightarrow AT \mid AB$

$T \rightarrow SB$

$A \rightarrow a$

$B \rightarrow b$

Accepts $L := \{a^n b^n \mid n \geq 1\}$

Advantages of grammar in Chomsky normal form:

- A derivation of a word w is at most $2|w| - 1$ steps long.
- A parse tree is a **binary tree**,
- so a parse tree of a word w has maximum depth $|w|$.

Idea of CYK algorithm: For each substring u of input word w , compute the set of non-terminals that can produce u .

Pumping Lemma for context-free languages

Theorem Let L be a context-free language over Σ^*

Then there exists a number $k > 0$

such that every word $z \in L$ with $|z| > k$

can be written as $z = u_1 v_1 w v_2 u_2$ such that

- $|v_1 w v_2| \leq k$
- $|v_1| + |v_2| > 0$
- $u_1 v_1^i w v_2^i u_2 \in L$ for all $i \geq 0$

Proofsketch If there is a sufficiently large word $z \in L$ then in the derivation of z one has

$$\begin{aligned} S &\Rightarrow u_1 A u_2 \\ A &\Rightarrow v_1 A v_2 \quad \text{a grammar loop!} \\ A &\Rightarrow w \end{aligned}$$

So: $S \Rightarrow u_1 v_1 w v_2 u_2 = z$ and also

$$S \Rightarrow u_1 A u_2 \Rightarrow u_1 v_1 A v_2 u_2 \Rightarrow \cdots u_1 v_1^i A v_2^i u_2 \Rightarrow u_1 v_1^i w v_2^i u_2 \in L$$

Applications of the pumping lemma for context-free languages

- $L_1 := \{a^k b^k c^k \mid k \geq 0\}$ is not context-free because it violates the pumping lemma for CF languages.
- $L_2 := \{ww \mid w \in \{a, b\}^*\}$ is not context-free because it violates the pumping lemma for CF languages.
- NB: $L_3 := \{a^k b^k \mid k \geq 0\}$ is context-free hence satisfies the pumping lemma for CF languages.