

Computational Models - Lecture 4¹

Handout Mode

Iftach Haitner.

Tel Aviv University.

November 21, 2016

¹Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University.

Talk Outline

- ▶ **Context Free** Grammars/Languages (CFG/CFL)
- ▶ (begin) Algorithmic issues for CFL

- ▶ Sipser's book, [2.1](#)

Short overview

So far we saw

- ▶ finite automata,
- ▶ regular languages,
- ▶ regular expressions,
- ▶ Myhill-Nerode theorem
- ▶ pumping lemma for **regular languages**.

We now introduce stronger machines and languages with more expressive power:

- ▶ pushdown automata,
- ▶ context-free languages,
- ▶ context-free grammars,
- ▶ pumping lemma for context-free languages.

Context Free Grammars (CFG)

An example of a context free grammar, G_1 :

- ▶ $A \rightarrow 0A1$
- ▶ $A \rightarrow B$
- ▶ $B \rightarrow \#$

Terminology:

- ▶ Each line is a **substitution rule** or **production**.
- ▶ Each rule has the form: **symbol** \rightarrow **string**.
The left-hand symbol is a **variable** (usually upper-case).
- ▶ A **string** consists of **variables** and **terminals**.
- ▶ One variable is the **start variable** (lhs of top rule). In this case, it is A .

Rules for generating strings

- ▶ Write down the start variable.
- ▶ Pick a variable written down in current string and a derivation that starts with that variable.
- ▶ Replace that variable with right-hand side of that derivation.
- ▶ Repeat until no variables remain.
- ▶ Return final string (concatenation of terminals).

Process is inherently **non deterministic**.

Example

Grammar G_1 :

- ▶ $A \rightarrow 0A1$
- ▶ $A \rightarrow B$
- ▶ $B \rightarrow \#$

Derivation with G_1 :

$A \rightarrow 0A1$
 $\rightarrow 00A11$
 $\rightarrow 000A111$
 $\rightarrow 000B111$
 $\rightarrow 000\#111$

Question 1

What strings can be generated in this way from the grammar G_1 ?

Answer: Exactly those of the form $0^n\#1^n$ ($n \geq 0$).

Context-Free Languages (CFL)

The language generated in this way is called the **language of the grammar**.

For example, $\mathcal{L}(G_1) = \{0^n \# 1^n : n \geq 0\}$.

Any language generated by a context-free grammar is called a **context-free language**.

A useful abbreviation

Rules with same variable on left hand side

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

are written as:

$$A \rightarrow 0A1 \mid B$$

English-like sentences

A grammar G_2 to describe a few English sentences:

< SENTENCE > → < NP >< VERB >
 < NP > → < ARTICLE >< NOUN >
 < NOUN > → boy | girl | flower
 < ARTICLE > → a | the
 < VERB > → touches | likes | sees

A specific derivation in G_2 :

< SENTENCE > → < NP >< VERB >
 → < ARTICLE >< NOUN >< VERB >
 → a < NOUN >< VERB >
 → a boy < VERB >
 → a boy sees

More strings generated by G_2 : a flower sees, the girl touches

English-like sentences, cont.

- < SENTENCE > → < NP >< VERB >
- < ARTICLE >< NOUN >< VERB >
- a < NOUN >< VERB >
- a boy < VERB >
- a boy sees

Formal definition

A **context-free grammar** is a 4-tuple (V, Σ, R, S) , where

- ▶ V is a finite set of **variables**
- ▶ Σ is a finite set of **terminals** $(V \cap \Sigma = \emptyset)$
- ▶ R is a finite set of **rules** of the form $A \rightarrow x$, where $A \in V$ and $x \in (V \cup \Sigma)^*$.
- ▶ $S \in V$ is the **start symbol**.
- ▶ Let $u, v \in (V \cup \Sigma)^*$. If $A \rightarrow w \in R$, then uAv **yields** uwv , denoted $uAv \rightarrow uwv$.
- ▶ $u \xrightarrow{*} v$ if $u = v$, or $u \rightarrow u_1 \rightarrow \dots \rightarrow u_k \rightarrow v$ for some sequence u_1, u_2, \dots, u_k

Note that if $A \xrightarrow{*} xBy$ and $B \xrightarrow{*} z$, then $A \xrightarrow{*} xzy$.

Definition 2

The **language of the grammar** G , denoted $\mathcal{L}(G)$, is $\{w \in \Sigma^* : S \xrightarrow{*} w\}$

where $\xrightarrow{*}$ is determined by G .

Example 1

$G_3 = (\{S\}, \{a, b\}, R, S)$.

R (Rules): $S \rightarrow aSb \mid SS \mid \epsilon$

Some words in the language: $aabb$, $aababb$.

Question 3

What is this language?

Hint: Think of parentheses: i.e., a is "(" and b is ")". $(())$, $((()))$

Using larger alphabet (i.e., more terminals), $([]())$, represent well formed programs with many kinds of nested loops, "if then/else" statements.

Example 2

$$G_4 = (\{S\}, \{a, b\}, R, S).$$

R (Rules): $S \rightarrow aSa \mid bSb \mid \epsilon$

Some words in the language: *abba, aabaabaa*.

Question 4

What is this language?

$\mathcal{L}(G_4) = \{ww^R : w \in \{a, b\}^*\}$ (almost but not quite the set of palindromes)

Proving $\mathcal{L}(G_4) = \mathcal{L} = \{ww^R : w \in \{a, b\}^*\}$

What do we need to show?

1. If $z = ww^R$ then z is generated by G_4 .
2. If z is generated by G_4 then $z = ww^R$.

Proving $ww^R \in \mathcal{L}(G_4)$. Proof by induction on the length of z .

- ▶ Base: $|z| = 0$. Since $S \rightarrow \varepsilon = z$, $z \in \mathcal{L}(G_4)$.
- ▶ Inductive step. Let $z = ww^R$ be a word of size $2k$.
 - ▶ Let $w = \sigma w'$ (hence $|w'| = k - 1$)
 - ▶ Hence, $z = \sigma w' (w')^R \sigma$
 - ▶ By i.h. $S \xrightarrow{*} w' (w')^R$
 - ▶ Hence, $S \rightarrow \sigma S \sigma \xrightarrow{*} \sigma w' (w')^R \sigma = z$

Proving $z \in \mathcal{L}(G_4) \implies z \in \{ww^R : w \in \{a,b\}^*\}$

Proof by induction on the number of derivations steps used to derive z from S .

- ▶ Single derivation. Only possible derivation is $S \rightarrow \varepsilon$, and thus $z = \varepsilon \in \mathcal{L}$.
- ▶ Inductive step. Assume $S \xrightarrow{*} z$ in k derivations steps.
 - ▶ First derivation is $S \rightarrow \sigma S \sigma$ for $\sigma \in \{a,b\}$
 - ▶ Hence $z = \sigma z' \sigma$, and z' is derived from S using $k - 1$ steps
 - ▶ By i.h., $z' = w'(w')^R \in \mathcal{L}$
 - ▶ Hence, $z = \sigma w'(w')^R \sigma = w(w)^R$ for $w = \sigma w'$ is in \mathcal{L}

Example 3

$$G_5 = (\{S, A, B\}, \{a, b\}, R, S)$$

R (Rules):

$$S \rightarrow aB \mid bA \mid \varepsilon$$

$$A \rightarrow a \mid aS \mid bAA$$

$$B \rightarrow b \mid bS \mid aBB$$

Some words in the language: *aababb*, *baabba*.

Question 5

What is this language?

$$\mathcal{L}(G_5) = \{w \in \{a, b\}^* : \#_a(w) = \#_b(w)\}$$

$\#_x(w)$ — number of occurrences of x in w

Proving $\mathcal{L}(G_5) = \mathcal{L} = \{w \in \{a, b\}^* : \#_a(w) = \#_b(w)\}$

What do we need to show?

1. $w \in \mathcal{L}(G_5) \implies w \in \mathcal{L}$.
2. $w \in \mathcal{L} \implies w \in \mathcal{L}(G_5)$.

Claim 6

If $S \xrightarrow{*} w$, then $\#_a(w) + \#_A(w) = \#_b(w) + \#_B(w)$.

Proof: DIY

Claim 7

For $w \in \{a, b\}^*$ let $k = k(w) = \#_a(w) - \#_b(w)$. Then:

1. If $k = 0$, then $S \xrightarrow{*} wS$
2. If $k > 0$, then $S \xrightarrow{*} wB^k$
3. If $k < 0$, then $S \xrightarrow{*} wA^{|k|}$

Are we done? How to prove the claim?

Proving Claim 7

For $w \in \{a, b\}^*$ let $k = k(w) = \#_a(w) - \#_b(w)$. Then:

1. If $k = 0$, then $S \xrightarrow{*} wS$
2. If $k > 0$, then $S \xrightarrow{*} wB^k$
3. If $k < 0$, then $S \xrightarrow{*} wA^{|k|}$

Proof by induction on $|w|$:

- ▶ Basis: $w = \epsilon$ then $k(w) = 0$. Since $S \xrightarrow{*} S$, then $S \xrightarrow{*} \epsilon S$
- ▶ Induction step: for $w \in \{a, b\}^n$, write $w = w'\sigma$ with $|w'| = n - 1$

Assume for concreteness (other cases proved analogously)

1. $\sigma = a$
2. $k' = k(w') = (\#_a(w') - \#_b(w')) = k - 1 > 0$

By i.h., $S \xrightarrow{*} w'B^{k'}$

Hence, $S \xrightarrow{*} w'B^{k'} = w'BB^{k'-1} \xrightarrow{*} w'aBBB^{k'-1} = wB^{k'+1} = wB^k$

Section 1

Parse Trees

Parse trees

Definition 8 (parse tree)

A labeled tree T is a **parse tree** of CFG $G = (V, \Sigma, R, S)$, if

- ▶ Inner nodes labels by elements of V .
- ▶ Leaves are labeled by elements of $V \cup \Sigma \cup \{\varepsilon\}$.
- ▶ If n_1, \dots, n_k are the labels of the direct descendants (from left to right) of a node labeled A , then $(A \rightarrow n_1, \dots, n_k) \in R$

The **yield** of T , are the labels of all its leaves ordered written from left to right.

Theorem 9

Let $G = (V, \Sigma, R, S)$ be a CFG, let $A \in V$ and let $x \in (\Sigma \cup V)^*$. Then $A \xrightarrow{*} x$ iff G has a parse tree root labeled by A , that yields x .

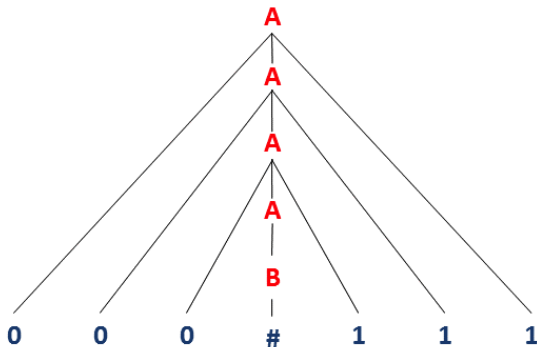
Proof? DIY

Might be that **several** derivation sequences (i.e., $S \rightarrow \dots \rightarrow w$) show that $w \in \mathcal{L}(G)$, but only a **single** tree rooted by S yields w (i.e., trees are “indifferent” to derivation order).

Example 1

$A \rightarrow 0A1|B$

$B \rightarrow \#$



The (unique) derivation sequence from A to $000\#111$:

$A \rightarrow 0A1 \rightarrow 00A11 \rightarrow 000A111 \rightarrow 000B111 \rightarrow 000\#111$

Example 2

< SENTENCE >

< NP >

< NOUN >

< ARTICLE >

< VERB >

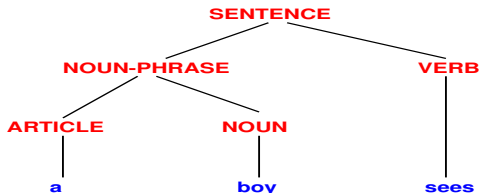
→ < NP >< VERB >

→ < ARTICLE >< NOUN >

→ boy | girl | flower

→ a | the

→ touches | likes | sees



Two derivation sequences from < SENTENCE > to *aboysees*:

- ▶ < SENTENCE > → < NP >< VERB > → < ARTICLE >< NOUN >< VERB > → *a* < NOUN >< VERB > → *aboy* < VERB > → *aboysees*
- ▶ < SENTENCE > → < NP >< VERB > → < ARTICLE >< NOUN >< VERB > → < ARTICLE >< NOUN > *sees* → < ARTICLE > *boyses* → *aboysees*

Section 2

Designing Context-Free Grammars

Designing CFGs

No recipe in general, but few rules-of-thumb

- ▶ If CFG is the **union** of several CFGs, rename variables (**not terminals**) so they are disjoint, and add new rule $S \rightarrow S_1 \mid S_2 \mid \dots \mid S_j$.
- ▶ For languages (like $\{0^n \# 1^n : n \geq 0\}$), with **linked** substrings, a rule of form $R \rightarrow uRv$ is helpful to force desired relation between substrings.
- ▶ For a regular language, grammar “follows” a DFA for the language (see next frame).

CFG for regular languages

Given a DFA : $M = (Q, \Sigma, \delta, q_0, F)$

CFG G for $\mathcal{L}(M)$:

1. Let R_0 be the starting variable
2. Add rule $R_i \rightarrow aR_j$, for any $q_i, q_j \in Q$ and $a \in \Sigma$ with $\delta(q_i, a) = q_j$
3. Add rule $R_i \rightarrow \varepsilon$, for any $q_i \in F$

Claim 10

$$\mathcal{L}(G) = \mathcal{L}(M)$$

Proof?

Claim 11

$$R_0 \xrightarrow{*} wR_j \text{ iff } \widehat{\delta}_M(q_0, w) = q_j.$$

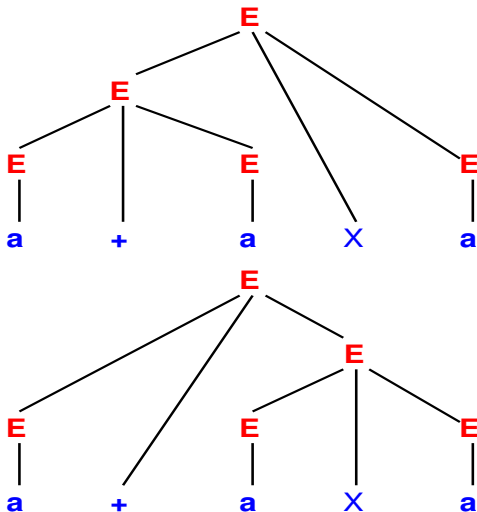
Proof? Next class we'll see alternative proof via "Push-Down Automata"

Section 3

Ambiguity in Context-Free Languages

Ambiguity in CFLs

Grammar $G: E \rightarrow E + E \mid E \times E \mid (E) \mid a$



Is this a problem?

Ambiguity in CFLs cont.

Consider the grammar $G' = (V, \Sigma, R, E)$, where

▶ $V = \{E, T, F\}$

▶ $\Sigma = \{a, +, \times, (,)\}$

▶ Rules:

$$\begin{array}{lcl} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T \times F \mid F \\ F & \rightarrow & (E) \mid a \end{array}$$

Claim 12

$$\mathcal{L}(G') = \mathcal{L}(G)$$

Proof Induction on derivation length.

But G' is not ambiguous. Proof (see Hopcroft, for a similar grammar)

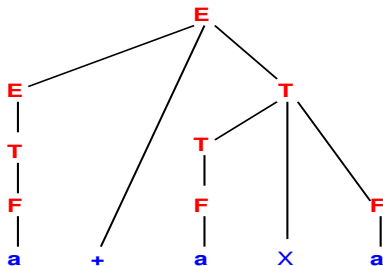
Parsing tree of G' for $a + a \times a$

G' :

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T \times F \mid F$$

$$F \rightarrow (E) \mid a$$



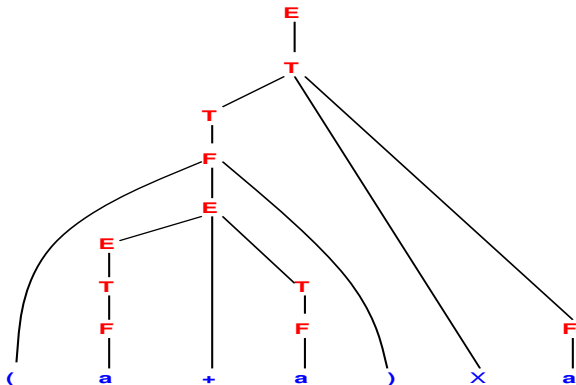
Parsing tree of G' for $(a + a) \times a$

G' :

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T \times F \mid F$$

$$F \rightarrow (E) \mid a$$



Ambiguity

Definition 13

A string w is derived **ambiguously** from grammar G , if w has two or more **different** parse trees that generate it from G . A CFG is **ambiguous**, if it ambiguously derives a string.

- ▶ Ambiguity is usually not only a syntactic notion but also **semantic**, implying multiple meanings for the same string. Think of $a + a \times a$ from last grammar.
- ▶ It is **sometime** possible to **eliminate** ambiguity by finding a different context free grammar generating the same language. This is true for the **arithmetic expressions** grammar.
- ▶ Some languages are **inherently** ambiguous.
- ▶ Example: $\mathcal{L} = \{a^n b^n c^m d^m : n, m \geq 1\} \cup \{a^n b^m c^m d^n : n, m \geq 1\}$
- ▶ \mathcal{L} is a CFL. Proof?
- ▶ \mathcal{L} is inherently ambiguous. Proof? see Hopcroft

Checking membership in CFLs

Challenge

Given a CFG G and a string w , decide whether $w \in \mathcal{L}(G)$?

Initial Idea: Design an algorithm that tries **all derivations**.

Problem: If G does **not** generate w , we'll never stop.

Possible solution: Use special grammars that are:

- ▶ just as expressive!
- ▶ better for checking membership.