

# Intro to Analysis of Algorithms

## Computational Foundations

### Section 9.4

### Chapter 9

Michael Soltys

CSU Channel Islands

[Ed: 4th, last updated: October 2, 2025]

# Context-free languages

A *context-free grammar (CFG)* is  $G = (V, T, P, S)$  — Variables, Terminals, Productions, Start variable

Ex.  $P \longrightarrow \varepsilon|0|1|0P0|1P1$ .

Ex.  $G = (\{E, I\}, T, P, E)$  where  $T = \{+, *, (, ), a, b, 0, 1\}$  and  $P$  is the following set of productions:

$$E \longrightarrow I|E + E|E * E|(E)$$

$$I \longrightarrow a|b|Ia|Ib|I0|I1$$

If  $\alpha A \beta \in (V \cup T)^*$ ,  $A \in V$ , and  $A \longrightarrow \gamma$  is a production, then  $\alpha A \beta \Rightarrow \alpha \gamma \beta$ . We use  $\stackrel{*}{\Rightarrow}$  to denote 0 or more steps.

$$L(G) = \{w \in T^* | S \stackrel{*}{\Rightarrow} w\}$$

**Lemma:**  $L(\{\{P\}, \{0, 1\}, \{P \rightarrow \varepsilon | 0|1|0P0|1P1\}, P)$  is the set of palindromes over  $\{0, 1\}$ .

**Proof:** Suppose  $w$  is a palindrome; show by induction on  $|w|$  that  $P \xRightarrow{*} w$ .

BS:  $|w| \leq 1$ , so  $w = \varepsilon, 0, 1$ , so use  $P \rightarrow \varepsilon, 0, 1$ .

IS: For  $|w| \geq 2$ ,  $w = 0x0, 1x1$ , and by IH  $P \xRightarrow{*} x$ .

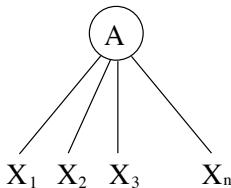
Suppose that  $P \xRightarrow{*} w$ ; show by induction on the number of steps in the derivation that  $w = w^R$ .

BS: Derivation has 1 step.

IS:  $P \Rightarrow 0P0 \xRightarrow{*} 0x0 = w$  (or with 1 instead of 0).

If  $S \xRightarrow{*} \alpha$ , then  $\alpha \in (V \cup T)^*$ , and  $\alpha$  is called a *sentential form*.  $L(G)$  is the set of those sentential forms which are in  $T^*$ .

Given  $G = (V, T, P, S)$ , the *parse tree* for  $(G, w)$  is a tree with  $S$  at the root, the symbols of  $w$  are the leaves (left to right), and each interior node is of the form:



whenever we have a rule  $A \longrightarrow X_1 X_2 X_3 \dots X_n$

Derivation: head  $\longrightarrow$  body

Recursive Inference: body  $\longrightarrow$  head

The following five are all equivalent:

1. Recursive Inference
2. Derivation
3. Left-most derivation
4. Right-most derivation
5. Yield of a parse tree.

## Ambiguity of Grammars

$$E \Rightarrow E + E \Rightarrow E + E * E$$

$$E \Rightarrow E * E \Rightarrow E + E * E$$

Two different parse trees! Different meaning.

A grammar is ambiguous if there exists a string  $w$  with two different parse trees.

A *Pushdown Automaton (PDA)* is an  $\varepsilon$ -NFA with a stack.

Two (equivalent) versions: (i) accept by final state, (ii) accept by empty stack.

PDAs describe CFLs.

The PDA pushes and pops symbols on the stack; the stack is assumed to be as big as necessary.

Ex. What is a simple PDA for  $\{ww^R \mid w \in \{0,1\}^*\}$  ?



Formal definition of a PDA:

$$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

$Q$  finite set of states

$\Sigma$  finite input alphabet

$\Gamma$  finite stack alphabet,  $\Sigma \subseteq \Gamma$

$$\delta(q, a, X) = \{(p_1, \gamma_1), \dots, (p_n, \gamma_n)\}$$

if  $\gamma = \varepsilon$ , then the stack is popped, if  $\gamma = X$ , then the stack is unchanged, if  $\gamma = YZ$  then  $X$  is replaced  $Z$ , and  $Y$  is pushed onto the stack

$q_0$  initial state

$Z_0$  start symbol

$F$  accepting states

A **configuration** is a tuple  $(q, w, \gamma)$ : state, remaining input, contents of the stack

If  $(p, \alpha) \in \delta(q, a, X)$ , then  $(q, aw, X\beta) \rightarrow (p, w, \alpha\beta)$

**Theorem:** If  $(q, x, \alpha) \rightarrow^* (p, y, \beta)$ , then  
 $(q, xw, \alpha\gamma) \rightarrow^* (p, yw, \beta\gamma)$

Acceptance by final state:

$$L(P) = \{w \mid (q_0, w, Z_0) \rightarrow^* (q, \varepsilon, \alpha), q \in F\}$$

Acceptance by empty stack:  $L(P) = \{w \mid (q_0, w, Z_0) \rightarrow^* (q, \varepsilon, \varepsilon)\}$

**Theorem:**  $L$  is accepted by PDA by final state iff it is accepted by PDA by empty stack.

**Proof:** When  $Z_0$  is popped, enter an accepting state. For the other direction, when an accepting state is entered, pop all the stack.

**Theorem:** CFGs and PDAs are equivalent.

**Proof:** From Grammar to PDA: A left sentential form is  $\underbrace{x}_{\in T^*} \overbrace{A\alpha}^{\text{tail}}$

The tail appears on the stack, and  $x$  is the prefix of the input that has been consumed so far.

Total input is  $w = xy$ , and hopefully  $A\alpha \xRightarrow{*} y$ .

Suppose PDA is in  $(q, y, A\alpha)$ . It guesses  $A \rightarrow \beta$ , and enters  $(q, y, \beta\alpha)$ .

The initial segment of  $\beta$ , if it has any terminal symbols, they are compared against the input and removed, until the first variable of  $\beta$  is exposed on top of the stack.

Accept by empty stack.

Ex. Consider  $P \rightarrow \varepsilon|0|1|0P0|1P1$

The PDA has transitions:

$$\delta(q_0, \varepsilon, Z_0) = \{(q, PZ_0)\}$$

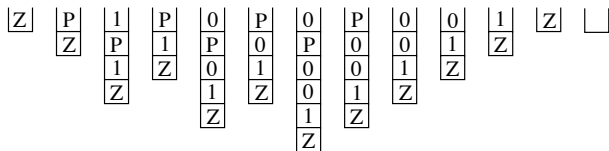
$$\delta(q, \varepsilon, P) = \{(q, 0P0), (q, 0), (q, \varepsilon), (q, 1P1), (q, 1)\}$$

$$\delta(q, 0, 0) = \delta(q, 1, 1) = \{(q, \varepsilon)\}$$

$$\delta(q, 0, 1) = \delta(q, 1, 0) = \emptyset$$

$$\delta(q, \varepsilon, Z_0) = (q, \varepsilon)$$

Consider:  $P \Rightarrow 1P1 \Rightarrow 10P01 \Rightarrow 100P001 \Rightarrow 100001$



From PDA to grammar:

Idea: “net popping” of one symbol of the stack, while consuming some input.

Variables:  $A_{[pXq]}$ , for  $p, q \in Q$ ,  $X \in \Gamma$ .

$A_{[pXq]} \xRightarrow{*} w$  iff  $w$  takes PDA from state  $p$  to state  $q$ , and pops  $X$  off the stack.

Productions: for all  $p$ ,  $S \longrightarrow A_{[q_0 Z_0 p]}$ , and whenever we have:

$$(r, Y_1 Y_2 \dots Y_k) \in \delta(q, a, X)$$

$A_{[qXr_k]} \longrightarrow a A_{[rY_1 r_1]} A_{[r_1 Y_2 r_2]} \dots A_{[r_{k-1} Y_k r_k]}$   
where  $a \in \Sigma \cup \{\varepsilon\}$ ,  $r_1, r_2, \dots, r_k \in Q$  are all possible lists of states.

If  $(r, \varepsilon) \in \delta(q, a, X)$ , then we have  $A_{[qXr]} \longrightarrow a$ .

**Claim:**  $A_{[qXp]} \xRightarrow{*} w \iff (q, w, X) \rightarrow^* (p, \varepsilon, \varepsilon)$ .

A PDA is deterministic if  $|\delta(q, a, X)| \leq 1$ , and the second condition is that if for some  $a \in \Sigma$   $|\delta(q, a, X)| = 1$ , then  $|\delta(q, \varepsilon, X)| = 0$ .

**Theorem:** If  $L$  is regular, then  $L = L(P)$  for some deterministic PDA  $P$ .

**Proof:** ignore the stack.

DPDAs that accept by final state are **not** equivalent to DPDAs that accept by empty stack.

$L$  has the *prefix property* if there exists a pair  $(x, y)$ ,  $x, y \in L$ , such that  $y = xz$  for some  $z$ .

Ex.  $\{0\}^*$  has the prefix property.

**Theorem:**  $L$  is accepted by a DPDA by empty stack  $\iff L$  is accepted by a DPDA by final state **and**  $L$  does not have the prefix property.

**Theorem:** If  $L$  is accepted by a DPDA, then  $L$  is unambiguous.

Eliminating useless symbols from CFG:

$X \in V \cup T$  is *useful* if there exists a derivation such that  
 $S \xRightarrow{*} \alpha X \beta \xRightarrow{*} w \in T^*$

$X$  is *generating* if  $X \xRightarrow{*} w \in T^*$

$X$  is *reachable* if there exists a derivation  $S \xRightarrow{*} \alpha X \beta$

A symbol is useful if it is generating and reachable.

Generating symbols: Every symbol in  $T$  is generating, and if  
 $A \rightarrow \alpha$  is a production, and every symbol in  $\alpha$  is generating (or  
 $\alpha = \varepsilon$ ) then  $A$  is also generating.

Reachable symbols:  $S$  is reachable, and if  $A$  is reachable, and  
 $A \rightarrow \alpha$  is a production, then every symbol in  $\alpha$  is reachable.



If  $L$  has a CFG, then  $L - \{\varepsilon\}$  has a CFG without productions of the form  $A \rightarrow \varepsilon$

A variable is *nullable* if  $A \xRightarrow{*} \varepsilon$

To compute nullable variables: if  $A \rightarrow \varepsilon$  is a production, then  $A$  is nullable, if  $B \rightarrow C_1 C_2 \dots C_k$  is a production and all the  $C_i$ 's are nullable, then so is  $B$ .

Once we have all the nullable variables, we eliminate  $\varepsilon$ -productions as follows: eliminate all  $A \rightarrow \varepsilon$ .

If  $A \rightarrow X_1 X_2 \dots X_k$  is a production, and  $m \leq k$  of the  $X_i$ 's are nullable, then add the  $2^m$  versions of the rule the the nullable variables present/absent (if  $m = k$ , do not add the case where they are *all* absent).

Eliminating unit productions:  $A \longrightarrow B$

If  $A \xRightarrow{*} B$ , then  $(A, B)$  is a unit pair.

Find all unit pairs:  $(A, A)$  is a unit pair, and if  $(A, B)$  is a unit pair, and  $B \longrightarrow C$  is a production, then  $(A, C)$  is a unit pair.

To eliminate unit productions: compute all unit pairs, and if  $(A, B)$  is a unit pair and  $B \longrightarrow \alpha$  is a non-unit production, add the production  $A \longrightarrow \alpha$ . Throw out all the unit productions.

A CFG is in *Chomsky Normal Form* if all the rules are of the form  $A \longrightarrow BC$  and  $A \longrightarrow a$ .

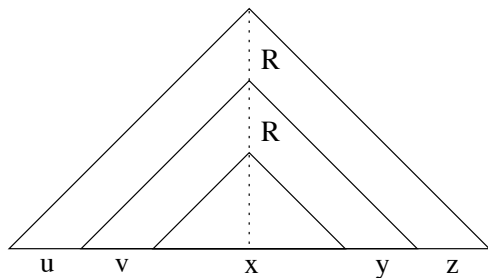
**Theorem:** Every CFL without  $\varepsilon$  has a CFG in CNF.

**Proof:** Eliminate  $\varepsilon$ -productions, unit productions, useless symbols. Arrange all bodies of length  $\geq 2$  to consist of only variables (by introducing new variables), and finally break bodies of length  $\geq 3$  into a cascade of productions, each with a body of length exactly 2.

**Pumping Lemma for CFLs:** There exists a  $p$  so that any  $s$ ,  $|s| \geq p$ , can be written as  $s = uvxyz$ , and:

1.  $uv^i xy^i z$  is in the language, for all  $i \geq 0$ ,
2.  $|vy| > 0$ ,
3.  $|vxy| \leq p$

Proof:



Ex. The lang  $\{0^n 1^n 2^n | n \geq 1\}$  is not CF.

So CFL are not closed under intersection:  $L_1 = \{0^n 1^n 2^i | n, i \geq 1\}$  and  $L_2 = \{0^i 1^n 2^n | n, i \geq 1\}$  are CF, but  $L_1 \cap L_2 = \{0^n 1^n 2^n | n \geq 1\}$  is not.

**Theorem:** If  $L$  is a CFL, and  $R$  is a regular language, then  $L \cap R$  is a CFL.

$L = \{ww : w \in \{0,1\}^*\}$  is not CF, but  $L^c$  is CF. So CFLs are not close under complementation either.

We design a CFG for  $L^c$ . First note that no odd strings are of the form  $ww$ , so the first rule should be:

$$\begin{aligned} S &\longrightarrow O|E \\ O &\longrightarrow a|b|aaO|abO|baO|bbO \end{aligned}$$

here  $O$  generates all the odd strings.

$E$  generates even length strings not of the form  $ww$ , i.e., all strings of the form:

$$X = | \_ \_ \_ \_ 0 \_ \_ | \_ \_ \_ \_ 1 \_ \_ | \quad \text{or} \quad Y = | \_ \_ \_ \_ 1 \_ \_ | \_ \_ \_ \_ 0 \_ \_ |$$

We need the rule:

$$E \longrightarrow X|Y$$

and now

$$\begin{array}{ll} X \longrightarrow PQ & Y \longrightarrow VW \\ P \longrightarrow RPR & V \longrightarrow SVS \\ P \longrightarrow a & V \longrightarrow b \\ Q \longrightarrow RQR & W \longrightarrow SWS \\ Q \longrightarrow b & W \longrightarrow a \\ R \longrightarrow a|b & S \longrightarrow a|b \end{array}$$

Ex.

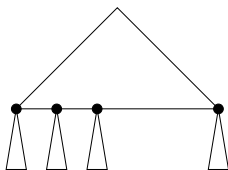
$$\begin{aligned} X &\Rightarrow PQ \Rightarrow RPRQ \Rightarrow RRPRRQ \Rightarrow RRRPRRRQ \Rightarrow RRRRPRRRRQ \\ &\Rightarrow RRRRRPRRRRRQ \Rightarrow RRRRRaRRRRRQ \Rightarrow RRRRRaRRRRRRQR \\ &\Rightarrow RRRRRaRRRRRRRRQRR \Rightarrow RRRRRaRRRRRRRRbRR \end{aligned}$$

and now the R's can be replaced at will by a's and b's.

CFL are closed under substitution: for every  $a \in \Sigma$  we choose  $L_a$ , which we call  $s(a)$ . For any  $w \in \Sigma^*$ ,  $s(w)$  is the language of  $x_1x_2 \dots x_n$ ,  $x_i \in s(a_i)$ .

**Theorem:** If  $L$  is a CFL, and  $s(a)$  is a CFL  $\forall a \in \Sigma$ , then  $s(L) = \cup_{w \in L} s(w)$  is also CF.

**Proof:**



CFL are closed under union, concatenation,  $*$  and  $+$ , homomorphism (just define  $s(a) = \{h(a)\}$ , so  $h(L) = s(L)$ ), and reversal (just replace each  $A \rightarrow \alpha$  by  $A \rightarrow \alpha^R$ ).

We can test for emptiness: just check whether  $S$  is generating.  
Test for membership: use CNF of the CYK algorithm (more efficient).

However, there are many **undecidable** properties of CFL:

1. Is a given CFG  $G$  ambiguous?
2. Is a given CFL inherently ambiguous?
3. Is the intersection of two CFL empty?
4. Given  $G_1, G_2$ , is  $L(G_1) = L(G_2)$ ?
5. Is a given CFL everything?



CYK<sup>1</sup> alg: Given  $G$  in CNF, and  $w = a_1 a_2 \dots a_n$ , build an  $n \times n$  table.  $w \in L(G)$  if  $S \in (1, n)$ .  $(X \in (i, j) \iff X \xRightarrow{*} a_i a_{i+1} \dots a_j.)$

Let  $V = \{X_1, X_2, \dots, X_m\}$ . Initialize  $T$  as follows:

for  $(i = 1; i \leq n; i++)$

for  $(j = 1; j \leq m; j++)$  Put  $X_j$  in  $(i, i)$  iff  $\exists X_j \rightarrow a_i$

Then, for  $i < j$ :

for  $(k = i; k < j; k++)$

if  $(\exists X_p \in (i, k) \ \& \ X_q \in (k+1, j) \ \& \ X_r \rightarrow X_p X_q)$

Put  $X_r$  in  $(i, j)$

x	(2,2)	(2,3)	(2,4)	(2,5)
x	x			(3,5)
x	x	x		(4,5)
x	x	x	x	(5,5)

---

<sup>1</sup>Cocke-Kasami-Younger

*Context-sensitive grammars (CSG)* have rules of the form:

$$\alpha \rightarrow \beta$$

where  $\alpha, \beta \in (T \cup V)^*$  and  $|\alpha| \leq |\beta|$ . A language is *context sensitive* if it has a CSG.

**Fact:** It turns out that  $CSL = NTIME(n)$

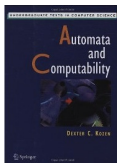
A *rewriting system* (also called a *Semi-Thue system*) is a grammar where there are no restrictions;  $\alpha \rightarrow \beta$  for arbitrary  $\alpha, \beta \in (V \cup T)^*$ .

**Fact:** It turns out that a rewriting system corresponds to the most general model of computation; i.e., a language has a rewriting system iff it is “computable.”

Enter Turing machines ...

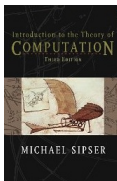
**Chomsky-Schutzenberger Theorem:** If  $L$  is a CFL, then there exists a regular language  $R$ , an  $n$ , and a homomorphism  $h$ , such that  $L = h(\text{PAREN}_n \cap R)$ .

**Parikh's Theorem:** If  $\Sigma = \{a_1, a_2, \dots, a_n\}$ , the signature of a string  $x \in \Sigma^*$  is  $(\#a_1(x), \#a_2(x), \dots, \#a_n(x))$ , i.e., the number of occurrences of each symbol, in a fixed order. The signature of a language is defined by extension; regular and CFLs have the same signatures.



## Automata and Computability

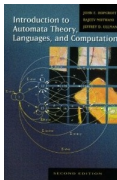
Dexter Kozen



## Intro to the theory of Computation

Third edition

Michael Sipser



## Intro to automata theory, languages and computation

Second edition

John Hopcroft, Rajeev Motwani, Jeffrey Ullman

*There is now a 3rd edition!*