Theory of Computation Slides based on Michael Sipser's Textbook

Moses A. Boudourides[1]

Visiting Associate Professor of Computer Science
Haverford College

[1] Moses.Boudourides@cs.haverford.edu

Section 2.3

More on Context–Free Grammars and Pushdown Automata

March 15, 2022

Regular Grammars, I

Definition: Right–, Left–, Regular and Linear Grammars

Let $G = (V, \Sigma, S, P)$ a CFG.

- G is called **right–linear** if all productions are of one of the two forms
$$A \to xB,$$
$$A \to x,$$
where $A, B \in V$ and $x \in \Sigma^*$.
- G is called **left–linear** if all productions are of one of the two forms
$$A \to Bx,$$
$$A \to x.$$
- G is called **regular grammar** if it is either right–linear or left–linear.
- G is called **linear grammar** if at most one variable can ocur on the right side of any production, independently of its position.

Regular Grammars, II

Example

Consider the following CFGs:

$G_1 = (\{S\}, \{a, b\}, S, \{S \to abS \mid a\})$,
$G_2 = (\{S, S_1, S_2\}, \{a, b\}, S, \{S \to S_1ab, S_1 \to S_1ab \mid S_2, S_2 \to a\})$,
$G_3 = (\{S, A, B\}, \{a, b\}, S, \{S \to A, A \to aB \mid \varepsilon, B \to Ab\})$.

Then G_1 is right–linear, G_2 is left–linear, and G_3 is linear. Notice that G_1 generates the regular language $(ab)^*a$, G_2 generates the regular language $aab(ab)^*$, and G_3 generates the nonregular language $\{a^nb^n \mid n \geq 0\}$.

Theorem

If G is a right–linear (or left–linear) CFG, then $L(G)$ is a regular language.

Theorem

If L is a regular language over Σ, then there exists a right–linear (or left–linear) CFG $G = (V, \Sigma, S, P)$ such that $L = L(G)$.

Regular Grammars, III

Correspondence of Regular Grammar to Finite Automaton

Assume that G is a right–linear CFG with variables $V = \{V_0, V_1, \ldots\}, S = V_0$, and productions of the form $V_0 \to v_i V_i, V_i \to v_2 V_j, \ldots$ or $V_n \to v_k$.

- Each variable in V is considered to be a state with the start state being the start variable.
- Each production $V_i \to \sigma V_j$ (where $\sigma \in \Sigma$) creates the transition $\delta(V_i, \sigma) = V_j$.
- Each production $V_i \to \sigma_1 \sigma_2 \cdots \sigma_m V_j$, where $\sigma_1 \sigma_2 \cdots \sigma_m \in \Sigma^*, m \geq 2$, creates the additional states $P_1, P_2, \ldots, P_{m-1}$ (other than states V_i and V_j) and the transitions $\delta(V_i, \sigma_1) = P_1, \delta(P_1, \sigma_2) = P_2, \ldots, \delta(P_{m-1}, \sigma_m) = V_j$.
- Each production $V_i \to \sigma$ (where $\sigma \in \Sigma$) creates a final state F and the transition $\delta(V_i, \sigma) = F$.
- Each production $V_i \to \tau_1 \tau_2 \cdots \tau_m$, where $\tau_1 \tau_2 \cdots \tau_m \in \Sigma^*, |\tau_1 \tau_2 \cdots \tau_m| > 0$ (necessarily $m > 1$), creates the additional states $Q_1, Q_2, \ldots, Q_{m-1}$ (other than the state V_i), a final state F, and the transitions $\delta(V_i, \tau_1) = Q_1, \delta(Q_1, \tau_2) = Q_2, \ldots, \delta(Q_{m-1}, \tau_m) = F$.
- Each production $V_i \to \varepsilon$ creates a final state F and a transition $\delta(V_i, \varepsilon) = F$.

Regular Grammars, IV

Example

Construct a finite automaton that accepts the regular language generated by the right–linear CFG $(\{S, V\}, \{a, b\}, S, \{S \to aV, V \to abS \mid b\})$.

This is the finite automaton with states $Q = \{S, V, P, F\}$ and transitions converted by productions as follows:

Productions	States	Transitions
$S \to aV$	S, V	$\delta(S, a) = V$
$V \to abS$	S, P, V	$\delta(V, a) = P$ $\delta(P, b) = S$
$V \to b$	V, F	$\delta(V, b) = F$

Apparently, the regular language is $(aab)^* ab$.

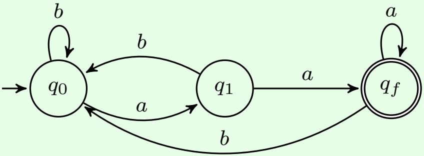
Correspondence of Finite Automaton to Regular Grammar

Assume that M is finite automaton over Σ with set of states Q, start state q_0 and (set of) final states F, and transitions given by function δ.

- Each state in Q other than final states is considered to be a variable with the start variable being the start state.
- Each transition $\delta(p, \sigma) = q$ creates the production $p \to \sigma q$.
- Each final state $q_f \in F$ creates the production $q_f \to \varepsilon$.

Example

Find the right–linear CFG corresponding to the language accepted by the following finite automaton:



The corresponding CFG has variables (q_0, q_1, q_f) (with q_0 the start variable) and productions converted by transitions as follows:

Transitions	States	Productions
$\delta(q_0, a) = q_1$	q_0, q_1	$q_0 \to aq_1$
$\delta(q_0, b) = q_0$	q_0	$q_0 \to bq_0$
$\delta(q_1, a) = q_f$	q_1, q_f	$q_1 \to aq_f$
$\delta(q_1, b) = q_0$	q_1, q_0	$q_1 \to bq_0$
$\delta(q_f, a) = q_f$	q_f	$q_f \to aq_f$
$\delta(q_f, b) = q_0$	q_f, q_0	$q_f \to bq_0$
Final state	q_f	$q_f \to \varepsilon$

Apparently, the regular language is $(a + b)^*aa$, i.e., strings ending in aa.

Simplification of Context–Free Grammars: Removing ε–Productions, I

Definition: ε–Productions and Nullable Variables

In a CFG, any production of the form

$$A \to \varepsilon$$

is called ε–**production** and any variable A, for which the derivation

$$A \Longrightarrow^* \varepsilon$$

is possible, is called **nullable**.

Theorem

Let $G = (V, \Sigma, S, P)$ a CFG containing ε–productions. Removing in P all ε–productions and adding new productions obtained by substituting ε for an ε–production wherever else the latter occurs in P, a CFG $\hat{G}$ is created such that $L(\hat{G}) = L(G) \setminus \{\varepsilon\}$.

Modifying ε–Productions

- If B is ε–production and $A \to BB$, then the latter production is modified as $A \to B$.
- If B, C are ε–productions and $A \to aBbCa$, then the latter production is modified as $A \to abCa \mid aBba$.

Removing ε–Productions, II

Example

Let G the CFG with ε–productions at the left column of the following table. The productions of the corresponding ε–productions–free CFG $\hat{G}$ are converted at the right column of the table:

Productions of G	Productions of $\hat{G}$
$S \to XY$	$S \to XY$
$X \to Zb$	$X \to Zb \mid b$
$Y \to bW$	$Y \to bW \mid b$
$Z \to AB$	$Z \to AB \mid A \mid B$
$W \to Z$	$W \to Z$
$A \to aA$	$A \to aA \mid a$
$A \to bA$	$A \to bA \mid b$
$A \to \varepsilon$	—
$B \to Ba$	$B \to Ba \mid a$
$B \to Bb$	$B \to Bb \mid b$
$B \to \varepsilon$	—

ε–productions: $A \to \varepsilon$ and $B \to \varepsilon$

Nullable variables $= \{A, B, Z, W\}$

Simplification of Context–Free Grammars: Removing Unit–Productions, I

Definition: Unit–Productions

Any production of a CFG $G = (V, \Sigma, S, P)$ of the form
$$A \to B,$$
where $A, B \in V$, is called a **unit–production**. Moreover, for any variable $C \in V$, we define the **unit set** of C as

$\mathrm{Unit}(C) = \{D \in V \mid C \implies^* D \text{ only through unit–productions}\}$.

Notice that, by default, $C \in \mathrm{Unit}(C)$.

Theorem

Let $G = (V, \Sigma, S, P)$ a CFG without ε–productions. Removing in P all unit–productions and adding new productions obtained from the unit set of the latter, a CFG $\hat{G}$ is created such that $\hat{G}$ does not have any ε–productions nor unit–productions and $L(\hat{G}) = L(G) \setminus \{\varepsilon\}$.

Modifying Unait–Productions

For each $B \in \mathrm{Unit}(A), B \neq A$, such that $B \to w$, for some $x \in \Sigma^*$ (non–unit–production of G), we are adding in $\hat{G}$ the production $A \to x$.

Removing Unit–Productions, II

Example

Let the CFG G, which includes the productions:

$$S \to A \mid Aa, A \to B, B \to C \mid b, C \to D \mid ab, D \to b.$$

Clearly,

$$\mathrm{Unit}(S) = \{S, A, B, C, D\}.$$

The productions of the corresponding unit–productions–free CFG $\hat{G}$ are converted in the table:

Productions of G	Productions of $\hat{G}$
$S \to Aa$	$S \to Aa$
$B \to b$	$B \to b$
$C \to ab$	$C \to ab$
$D \to b$	$D \to b$
$\mathrm{Unit}(S) = \{S, A, B, C, D\}$	$S \to b \mid ab$
$\mathrm{Unit}(A) = \{A, B, C, D\}$	$A \to b \mid ab$
$\mathrm{Unit}(B) = \{B, C, D\}$	$B \to b \mid ab$
$\mathrm{Unit}(C) = \{C, D\}$	$C \to b$

Chomsky Normal Form, I

Definition: Chomsky Normal Form

A CFG $G = (V, \Sigma, S, P)$ is in **Chomsky normal form** if all productions are of the form

$$A \to BC,$$

or

$$A \to a,$$

where $A, B, C \in V$ and $a \in \Sigma$. (Notice that $a \neq \varepsilon$.)

Theorem

Let $G = (V, \Sigma, S, P)$ a ε–productions–free CFG. Modifying G's productions, an equivalent CFG $\hat{G}$ in Chomsky normal form can be created.

Chomsky Normal Form Conversion

- G–productions of the form $A \to BC$ or $A \to a$ are preserved in $\hat{G}$.
- G–productions of the form $A \to B_1 B_2 \cdots B_m$, for $m \geq 2, B_1, \ldots, B_m \in V \cup \Sigma$, preserve the (old) variables A, B_1, B_{m-1}, B_m and create the (new) $\hat{G}$–variables $D_1, \ldots, D_{m-1}$ into the (new) $\hat{G}$–productions $A \to B_1 D_1, D_1 \to B_2 D_2, \ldots,$ $D_{m-3} \to B_{m-2} D_{m-2}, D_{m-2} \to B_{m-1} B_m$.

Chomsky Normal Form, II

Example

Let the CFG G, which includes the productions:

$$S \to bA \mid aB, A \to bAA \mid aS \mid a, B \to aBB \mid bS \mid b.$$

The productions of the corresponding Chomsky normal form CFG $\hat{G}$ are converted in the table:

Productions of G	Productions of $\hat{G}$
$S \to bA \mid aB$	$S \to C_b A \mid C_a B$ $C_a \to a, C_b \to b$
$A \to bAA$	$A \to C_b D_1$ $D_1 \to AA$
$B \to aBB$	$B \to C_a D_2$ $D_2 \to BB$
$D \to b$	$D \to b$
$A \to aS$	$A \to C_a S$
$B \to bS$	$B \to C_b S$
$A \to a$	$A \to a$
$B \to b$	$B \to b$

Pushdown Automata from Context–Free Grammars, I

Theorem

For every CFG G, there is a PDA M such that $L(M) = L(G)$.

Construction of a PDA from a CFG

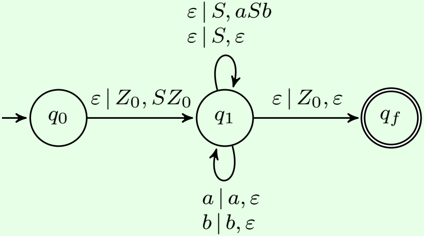
Let $L = L(G)$, where $G = (V, \Sigma, S, P)$ is a CFG. We are constructing a PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, F, \delta)$, which accepts language L, as follows:

$$
\begin{aligned}
Q &= \{q_0, q_1, q_f\}, \text{ three (arbitrary) states,} \\
q_0 &= \text{the start state,} \\
F &= \{q_f\}, \\
\Gamma &= V \cup \Sigma \cup \{Z_0\}, \\
Z_0 &= \text{the start stack symbol,} \\
\delta(q_0, \varepsilon, Z_0) &= (q_1, SZ_0), \\
\delta(q_1, \varepsilon, A) &\ni (q_1, w), \text{ whenever } A \to w \text{ is a } G\text{–production,} \\
\delta(q_1, a, a) &\ni (q_1, \varepsilon), \text{ whenever } a \in \Sigma, \\
\delta(q_1, \varepsilon, Z_0) &= (q_f, \varepsilon).
\end{aligned}
$$

Pushdown Automata from Context–Free Grammars, II

Example

The PDA corresponding to the CFG $G = (\{S\}, \{a, b\}, S, \{S \to aSb \mid \varepsilon\})$, i.e., generating the language $L = \{a^i b^i \mid i \geq 0 \text{ integer}\}$, with three states $\{q_0, q_1, q_f\}$, input alphabet $\{a, b\}$, and stack alphabet $\{S, a, b, Z_0\}$:



Context–Free Grammars from Pushdown Automata, I

Theorem

For every PDA M, there is a CFG G such that $L(G) = L(M)$.

Construction of a CFG from a PDA

Let the PDA $M = (Q, \Sigma, \Gamma, q_0, Z_0, F, \delta)$ such that, for any $p, q \in Q, \sigma \in \Sigma$, $\gamma, \delta \in \Gamma$,

- $F = \{q_f\}$ (and M terminates only with empty stack),
- every transition is of one the two forms

  $$\delta(q, \sigma, \gamma) \ni (p, \varepsilon) \text{ or } \delta(p, \sigma, \gamma) \ni (p, \delta\gamma),$$

  which means that every transition either decreases or increases the stack content by a single symbol.

We are constructing a CFG $G = (V, \Sigma, S, P)$, which generates the language $L(G)$, as follows:

$$\begin{aligned} V &= \{[q\gamma p] \,|\, p, q \in Q, \gamma \in \Gamma\}, \\ S &= [q_0 Z_0 q_f], \\ P &= \{[q\gamma p] \to \sigma, \text{ whenever } \delta(q, \sigma, \gamma) \ni (p, \varepsilon)\}. \end{aligned}$$

Context–Free Grammars from Pushdown Automata, II

Example

Let the PDA $M = (\{q_0, q_1, q_f\}, \{a, b\}, \{X, Z_0\}, q_0, Z_0, \{q_f\}, \delta)$ possess the following transitions:

$$\delta(q_0, a, Z_0) = (q_0, XZ_0),$$
$$\delta(q_0, a, X) = (q_0, XX),$$
$$\delta(q_0, b, X) = (q_1, \varepsilon),$$
$$\delta(q_1, b, X) = (q_1, \varepsilon),$$
$$\delta(q_1, \varepsilon, X) = (q_1, \varepsilon),$$
$$\delta(q_1, \varepsilon, Z_0) = (q_f, \varepsilon).$$

We are constructing the corresponding CFG $G = (V, \Sigma, S, P)$ with the following variables:

$$\begin{aligned} V = \{&[q_0Xq_0], [q_0Xq_1], [q_0Xq_f], [q_0Z_0q_0], [q_0Z_0q_1], [q_0Z_0q_f] \equiv S, \\ &[q_1Xq_0], [q_1Xq_1], [q_1Xq_f], [q_1Z_0q_0], [q_1Z_0q_1], [q_1Z_0q_f], \\ &[q_fXq_0], [q_fXq_1], [q_fXq_f], [q_fZ_0q_0], [q_fZ_0q_1], [q_fZ_0q_f]. \end{aligned}$$

Example (cont.)

Moreover, the productions of G are created as follows:

Transversal Transitions of M	Productions of G
$\delta(q_0, b, X) = (q_1, \varepsilon)$	$[q_0 X q_1] \to b$
$\delta(q_1, b, X) = (q_1, \varepsilon)$	$[q_1 X q_1] \to b$
$\delta(q_1, \varepsilon, X) = (q_1, \varepsilon)$	$[q_1 X q_1] \to \varepsilon$
$\delta(q_1, \varepsilon, Z_0) = (q_f, \varepsilon)$	$[q_1 Z_0 q_f] \to \varepsilon$

Productions of G created by transition $\delta(q_0, a, Z_0) = (q_0, XZ_0)$
$[q_0 Z_0 q_0] \to a[q_0 X q_0][q_0 Z_0 q_0] \mid a[q_0 X q_1][q_1 Z_0 q_0] \mid a[q_0 X q_f][q_f Z_0 q_0]$
$[q_0 Z_0 q_1] \to a[q_0 X q_0][q_0 Z_0 q_1] \mid a[q_0 X q_1][q_1 Z_0 q_1] \mid a[q_0 X q_f][q_f Z_0 q_1]$
$[q_0 Z_0 q_f] \to a[q_0 X q_0][q_0 Z_0 q_f] \mid a[q_0 X q_1][q_1 Z_0 q_f] \mid a[q_0 X q_f][q_f Z_0 q_f]$

Productions of G created by transition $\delta(q_0, a, X) = (q_0, XX)$
$[q_0 Z_0 q_0] \to a[q_0 X q_0][q_0 X q_0] \mid a[q_0 X q_1][q_1 X q_0] \mid a[q_0 X q_f][q_f X q_0]$
$[q_0 Z_0 q_1] \to a[q_0 X q_0][q_0 X q_1] \mid a[q_0 X q_1][q_1 X q_1] \mid a[q_0 X q_f][q_f X q_1]$
$[q_0 Z_0 q_f] \to a[q_0 X q_0][q_0 X q_f] \mid a[q_0 X q_1][q_1 X q_f] \mid a[q_0 X q_f][q_f X q_f]$

Notice that the only variables which are used in the previous productions are:

$$V = \{[q_0 Z_0 q_f] \equiv S, [q_0 X q_0], [q_0 X q_1], [q_0 X q_f], [q_0 Z_0 q_0], [q_0 Z_0 q_1], [q_1 Z_0 q_f], [q_1 X q_1]\},$$

while all the other variables may be dropped from the corresponding productions.

The Pumping Lemma for Context–Free Languages

Theorem: The Pumping Lemma for Context–Free Languages

If L is a context–free language (**CFL**) over alphabet Σ, then there is a positive integer n so that, for every $x \in L$ with $|x| \geq n$, x can be written as $x = uvwxy$, for some strings $u, v, w, x, y \in \Sigma^*$ satisfying:

- $|vwx| \leq n$,
- $|vx| \geq 1$, i.e., $v \neq \varepsilon$ or $x \neq \varepsilon$,
- for every integer $m \geq 0, uv^m wx^m y \in L$.

Corollary

Let L be a CFL and n the positive integer of the Pumping Lemma. Then:

- $L \neq \varnothing$ if and only if there exists a $w \in L$ with $|w| < n$, and
- L is infinite if and only if there exists a $z \in L$ such that $n \leq |z| < 2n$.

Theorem: Ogden's Lemma

If L is a context–free language (**CFL**) over alphabet Σ, then there is a positive integer n so that, for every $x \in L$ with $|x| \geq n$, if we mark at least n symbols of x (i.e., if we choose n or more "distinguished" positions in the string x), x can be written as $x = uvwxy$, for some strings $u, v, w, x, y \in \Sigma^*$ satisfying:

- the string vwx contains at most n marked symbols,
- the string vx contains at least one marked symbol, and
- for every integer $m \geq 0, uv^m wx^m y \in L$.

Closure Properties of Context–Free Languages

Theorem 1

The class of CFLs is closed under union, concatenation, and Kline star.

Theorem 2

The class of CFLs is not closed under intersection and complementation.

Theorem 3

The intersection of a CFL with a regular language is a CFL.