MA513: Formal Languages and Automata Theory
Topic: Pushdown Automata (PDA) Continued
Lecture Number 25   Date: October 11, 2011

1 Equivalence of PDA's and CFG's

The goal is to prove that the following three classes of the languages are all the same class.

1. The context-free languages (The language defined by CFG's).
2. The languages that are accepted by empty stack by some PDA.
3. The languages that are accepted by final state by some PDA.

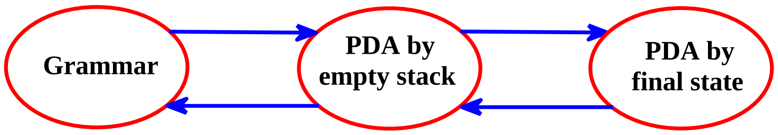

Figure 1: Organization of constructions showing equivalence of three ways of defining the CFL's

We have already shown that (2) and (3) are the same. Now, we prove that (1) and (2) are same.

Recall the following theorem from the chapter *context-free grammar*.

Theorem: Let $G = (V, T, R, S)$ be a CFG, and suppose there is a parse tree with root labeled by variable **A** and with yield $w (\in T^*)$. Then there is a leftmost derivation $A \overset{*}{\Rightarrow}_{lm} w$ in grammar G.

1.1 From Grammar to Pushdown Automata

Given a CFG G, we construct a PDA that simulates the leftmost derivations of G. Any left-sentential form that is not a terminal string can be written as $xA\alpha$, where A is the leftmost variable, x is whatever terminals appear to its left, and α is the string of terminals and variables that appear to the right of A. We call $A\alpha$ the tail of this left-sentential form. If a left-sentential form consists of terminals only, then its tail is ϵ.

The idea behind the construction of a PDA from a grammar is to have the PDA simulate the sequence of left-sentential forms that the grammar uses to generate a given terminal string **w**. The tail of each sentential form $xA\alpha$ appears on the stack, with A at the top. At that time, **x** will be **represented** by our having consumed **x** from the input, leaving whatever of **w** follows its prefix **x**. That is, if $w = xy$, then **y** will remain on the input.

Suppose the PDA is in an ID $(q, y, A\alpha)$, representing left-sentential form $xA\alpha$. It guesses the production to use to expand **A**, say $A \rightarrow \beta$. The move of the PDA is to replace **A** on the top of the stack by β, entering ID $(q, y, \beta\alpha)$. Note that there is only one state, **q**, for this PDA.

Now, $(q, y, \beta\alpha)$ may not be a representation of the next left-sentential form, because β may have a prefix of terminals. In fact, β may have no variables at all, and α may have a prefix of terminals. Whatever terminals appear at the beginning of $\beta\alpha$ need to be removed, to expose the next variable at the top of the stack. These terminals are compared against the next input symbols, to make sure our guesses at the leftmost derivation of input string w are correct; if not, this branch of the PDA dies.

If we succeed in this way to guess a leftmost derivation of w, then we shall eventually reach the left-sentential form w. At that point, all the symbols on the stack have either been expanded (if they are variables) or matched against the input (if they are terminals). The stack is empty, and we accept by empty stack.

The above informal construction can be maid precise as follows. Let $G = (V, T, R, S)$ be a CFG. Construct the PDA P that accepts $L(G)$ by empty stack as follows:

$$P = (\{q\}, T, V \cup T, \delta, q, S)$$

where transition function δ is defined by:

1. For each variable A, $\delta(q, \epsilon, A) = \{(q, \beta) | A \rightarrow \beta \text{ is a production of } P\}$.
2. For each terminal **a**, $\delta(q, a, a) = \{(q, \epsilon)\}$.

Example: Consider the grammar $G = (V, T, R, S)$ with $V = \{S\}$, $T = \{a, b, c\}$, and $R = \{S \rightarrow aSa, S \rightarrow bSb, S \rightarrow c\}$, which generates the language $\{wcw^R | w \in \{a, b\}^*\}$. The corresponding pushdown automaton (acceptance by empty stack) is $P = (\{q\}, T, V \cup T, \delta, q, S)$, where the transition function δ is given by:

a) $\delta(q, \epsilon, S) = \{(q, aSa), (q, bSb), (q, c)\}$

b) $\delta(q, a, a) = (q, \epsilon), \delta(q, b, b) = (q, \epsilon), \delta(q, c, c) = (q, \epsilon)$

Theorem: If PDA P is constructed from CFG G by the construction above, then $N(P) = L(G)$.
See Hopcroft, Motwani and Ullman book for proof of this theorem.

1.2 From PDA's to Grammar

The construction of an equivalent grammar uses variables each of which represent an **event** consisting of:

1. The net popping of some symbol X from the stack.
2. A change in state from some p at the beginning to q when X has finally been replaced by ϵ on the stack.

• If $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0)$ is a PDA, then there is a context-free grammar $G = (V, \Sigma, R, S)$ such that $L(G) = N(P)$, where the set of variables V consists of :

1. The special symbol S, which is the start symbol of G and
2. All symbols of the form $[pXy]$, where $p, q \in Q$ and $x \in \Gamma$.

The rules R of G are as follows:

a) For all states p, G has the rules $S \to [q_0 z_0 p]$ (since $(q_0, w, z_0) \overset{*}{\vdash} (p, \epsilon, \epsilon)$).

b) Let $\delta(q, a, X)$ contains the pair $(r, Y_1Y_2 \ldots Y_k)$, where

   1. **a** is either a symbol in Σ or $a = \epsilon$.
   2. k be any number, including 0, in which case the pair is (r, ϵ).

   Then for all lists of states $r_1, r_2, \ldots, r_k$, G has the rules
   $[qXr_k] \to a[rY_1r_1][r_1Y_2r_2] \ldots [r_{k-1}Y_kr_k]$
   This rules says that one way to pop X and go from state q to state r_k is to read **a** (which may be ϵ), then use some input to pop Y_1 off the stack which going from state r to state r_1, then read some more input that pops Y_2 off the stack and goes from state r_1 to state r_2, and so on.

Example: Consider the PDA $P_N = (\{q\}, \{0, 1\}, \{Z, A, B\}, \delta_N, q, Z)$ in Figure 2. The corresponding context-free grammar $G = (V, \{0, 1\}, R, S)$ is given by:

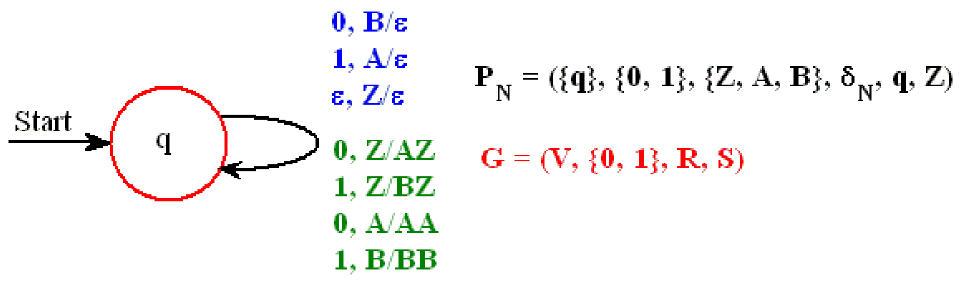

Figure 2: Example of PDA

$V = \{S, [qZq], [qAq], [qBq]\}$.

$R =$

1. $S \rightarrow [qZq]$
2. $[qZq] \rightarrow 0[[qAq][qZq]$ (since $\delta_N(q, 0, Z)$ contains (q, AZ))
3. $[qZq] \rightarrow 1[[qBq][qZq]$ (since $\delta_N(q, 1, Z)$ contains (q, BZ))
4. $[qAq] \rightarrow 0[[qAq][qAq]$ (since $\delta_N(q, 0, A)$ contains (q, AA))
5. $[qBq] \rightarrow 1[[qBq][qBq]$ (since $\delta_N(q, 1, B)$ contains (q, BB))
6. $[qAq] \rightarrow 1$ (since $\delta_N(q, 1, A)$ contains (q, ϵ))
7. $[qBq] \rightarrow 0$ (since $\delta_N(q, 0, B)$ contains (q, ϵ))
8. $[qZq] \rightarrow \epsilon$ (since $\delta_N(q, \epsilon, Z)$ contains (q, ϵ))