MA513: Formal Languages and Automata Theory
Topic: Pushdown Automata (PDA) Continued
Lecture Number 23   Date: September 30, 2011

1 The Languages of a PDA

We have assumed that a PDA accepts its input by consuming it and entering an accepting state. We call this approach **acceptance by final state**. We may also define for any PDA the language **accepted by empty stack**, that is, the set of strings that cause the PDA to empty it stack, starting from the initial ID.

These two methods are equivalent, in the sense that a language L has a PDA that accepts it by final state **if and only if** L has a PDA that accepts it by empty stack. However for a given PDA P, the languages that P accepts by final state and by empty stack are usually different. We will show conversion of a PDA accepting L by final state into another PDA that accepts L by empty stack, and vice-versa.

1.1 Acceptance by Final State

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. Then $L(P)$, the language accepted by P *by final state*, is

$$L(P) = \{w | (q_0, w, Z_0) \overset{*}{\vdash} (q, \epsilon, \alpha)\}$$

for some state $q \in F$ and any stack string α. That is, starting in the initial ID with w waiting on the input, P consumes w from the input and enters an accepting state. The content of the stack at that time is irrelevant.

1.2 Acceptance by Empty Stack

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. Then $N(P)$, the language accepted by P *by empty stack*, is

$$N(P) = \{w | (q_0, w, Z_0) \overset{*}{\vdash} (q, \epsilon, \epsilon)\}$$

for any state q. That is, $N(P)$ is the set of inputs w that P can consume and at the same time empty its stack. (The N in $N(P)$ stands for **null stack**, a synonym for **empty stack**.)

2 From Empty Stack to Final State

Objective of this section is show the conversion from a PDA P_n that accepts a language L by *empty stack* to a PDA P_f that accepts L by *final state*.

Theorem: If $L = N(P_n)$ for some PDA $P_n = (Q_n, \Sigma, \Gamma_n, \delta_n, q_0, Z_0, F_n)$, then there is a PDA $P_f = (Q_f, \Sigma, \Gamma_f, \delta_f, p_0, X_0, F_f)$ such that $L = L(P_f)$.

Proof: The idea behind the proof is in Figure 1. We use a new symbol X_0, which must not be a symbol of Γ_n; X_0 is both the start symbol of P_f and a marker on the bottom of the stack that lets us know when P_n has reached an *empty stack*. That is, if P_f sees X_0 on top of the stack, then it knows that P_n would empty its stack on the same input.

We also need a new start state, p_0, whose sole function is to push Z_0, the start state of P_n, onto the top of the stack and enter state q_0, the start state of P_n. Then, P_f simulates P_n, until the stack of P_n is empty, which P_f detects because it sees X_0 on the top of the stack. Finally, we need another new state, p_f, which is the accepting state of P_f; this PDA transfers to state p_f whenever it discover that P_n would have emptied its stack.

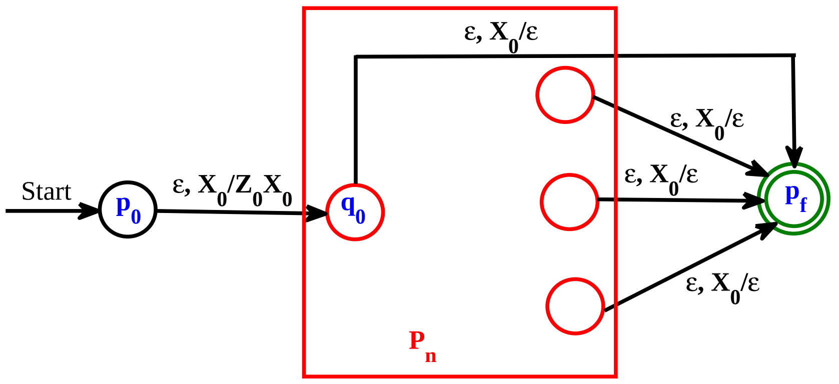

Figure 1: P_f simulates P_n and accepts if P_n empties its stack

The specification of P_f is as follows:

$$Q_f = Q_n \cup \{p_0, p_f\}.$$

$$\Gamma_f = \Gamma_n \cup \{X_0\}.$$

$$F_f = \{p_f\}.$$

δ_f is defined by

1. $\delta_f(p_0, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$. In its start state, P_f makes a spontaneous transition to the start state of P_n, pushing its start symbol Z_0 onto the stack.
2. For all state $q \in Q_n$, inputs $a \in \Sigma_n$ or $a = \epsilon$, and stack symbol $Y \in \Gamma_n$, $\delta_f(q, a, Y)$ contains all the pairs in $\delta_n(q, a, Y)$.
3. In addition to rule (2), $\delta_f(q, \epsilon, X_0)$ contains (p_f, ϵ) for every state $q \in Q_n$.

We must show that w is in $L(P_f)$ **if and only if** w is in $N(P_n)$.

(If) We are given that $(q_0, w, Z_0) \overset{*}{\vdash}_{P_n} (q, \epsilon, \epsilon)$ for some state q. Insert X_0 at the bottom of the stack and conclude $(q_0, w, Z_0X_0) \overset{*}{\vdash}_{P_n} (q, \epsilon, X_0)$. Since by rule (2) above, P_f has all the moves of P_n, we may also conclude that $(q_0, w, Z_0X_0) \overset{*}{\vdash}_{P_f}$ (q, ϵ, X_0). If we put this sequence of moves with the initial and final moves from rules (1) and (3) above, we get:

$$(p_0, w, X_0) \vdash_{p_f} (q_0, w, Z_0X_0) \overset{*}{\vdash}_{P_f} (q, \epsilon, X_0) \vdash_{p_f} (p_f, \epsilon, \epsilon) \ldots \text{(A)}$$

Thus, P_f accepts w by final state.

Only-if The converse requires only that we observe the additional transitions of rules (1) and (3) gives us very limited ways to accept w by final state. We must use rule (3) at the last step, and we can only use that rule if the stack of P_f contains only X_0. No X_0's ever appear on the stack except at the bottommost position. Further, rule (1) is only used at the first step, and it must be used at the first step.

Thus, any computation of P_f that accept w must look like sequence (A). Moreover, the middle of the computation - all but the first and last steps - must also be a computation of P_n with X_0 below the stack. The reason is that, except for the first and last steps, P_f cannot use any transition that is not also a transition of P_n, and X_0 cannot be exposed or the computation would end at the next step. We conclude that $(q_0, w, Z_0) \overset{*}{\vdash}_{P_n} (q, \epsilon, \epsilon)$. That is $w \in N(P_n)$.