University of Virginia - cs3102: Theory of Computation	Spring 2010
Pushdown Automata	**9 February**

A *pushdown automata* is a finite automaton with a stack. A stack can contain any number of elements, but only the top element may be accessed.

We represent a stack as a sequence of elements, $s_0 s_1 \ldots s_n$ where s_0 is the top of the stack. We use Γ (Gamma) to represent the stack alphabet. Γ is a finite set of symbols. So, a stack is an element of Γ^*.

A *deterministic pushdown automaton* (DPDA) is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q, Σ, q_0, and F are defined as they are for a deterministic finite automaton, Γ is a finite state (the stack alphabet), and transition function:

$$\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow (Q \times \Gamma_\epsilon) \cup \{\emptyset\}$$

Note: Sipser defines a *nondeterministic pushdown automaton* (Definition 2.1) and uses *pushdown automata* to mean "deterministic pushdown automata", but does not define a *deterministic pushdown automaton.*

Since the DPDA is *deterministic*, the δ function must not have only one possible choice at all steps. What rules ensure this?

We can use any symbols we want in the stack alphabet, Γ. As with state labels, in designing a DPDA, it is important to give symbols names that have meaning. Typically, we use $ as a special symbol, often meaning the bottom of the stack.

We use label arrows in a DPDA as $\Sigma, \Gamma_\epsilon \rightarrow \Gamma_\epsilon$. For $a \in \Sigma$, $h_t, h_p \in \Gamma$:

- $a, h_t \rightarrow h_p$ means if the current input is a and the top-of-stack is h_t, follow this transition and pop the h_t off the stack, and push the h_p.
- $a, \epsilon \rightarrow h_p$ means if the current input is a, follow this transition and push h_p on the stack. (It doesn't matter what is on top of the stack.)
- $a, h_t \rightarrow \epsilon$ means if the current input is a and the top-of-stack is h_t, follow this transition and pop the h_t off the stack.
- $a, \epsilon \rightarrow \epsilon$ means if the current input is a, follow this transition and don't modify the stack.

Prove that a DPDA is *more powerful* than a DFA.

Describe a DPDA that can recognize the language $\{w|w \text{ contains more } a\text{s than } b\text{s}\}$.

Model of Computation for Deterministic Pushdown Automata

To define the model of computation for a DPDA, we define the extended transition function, δ^*, similarly to how we did for DFAs, except we need to model the stack.

$\forall q \in Q, \forall a \in \Sigma, x \in \Sigma^*, \gamma \in \Gamma^*, h \in \Gamma$:

$$\delta^*(q, \epsilon, \gamma) = E(q, \gamma)$$
$$\delta(q, a, h_t) \to (q_t, h_p) \Rightarrow \delta^*(q, ax, h_t\gamma) = \delta^*(q_r, x, \gamma_r) \text{ where } (q_r, \gamma_r) = E(q_t, h_p\gamma)$$

$E : Q \times \Gamma^* \to Q \times \Gamma^*$ is the forced-follow ϵ-transitions function defined by:

$$\delta(q, \epsilon, \gamma) = \emptyset : E(q, \gamma) = (q, \gamma)$$
$$\delta(q, \epsilon, h_t\gamma) = (q_t, h_p\gamma) : E(q, h_t\gamma) = E(q_t, h_p\gamma)$$

Accepting State Model: A deterministic pushdown automata, $A = (Q, \Sigma, \Gamma, \delta, q_0, F)$ accepts a string $w \in \Sigma^*$ if and only if: $\delta * (q_0, w, \epsilon) \to (q_f, s) \wedge q_f \in F$.

Empty Stack Model: A deterministic pushdown automata, $A = (Q, \Sigma, \Gamma, \delta, q_0)$ (note there is no F now) accepts a string $w \in \Sigma^*$ if and only if: $\delta * (q_0, w, \epsilon) \to (q, s) \wedge s = \epsilon$.

Nondeterministic Pushdown Automaton

A *nondeterministic pushdown automaton* (this is what Sipser calls a *pushdown automaton*) is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where $Q, \Sigma, \Gamma, q_0, F$ are defined as they are for DPDA and the transition function is defined:

$$\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$$

Example. Define a NPDA that recognizes the language $\{ww^R|w \in \Sigma^*\}$.