PDAs

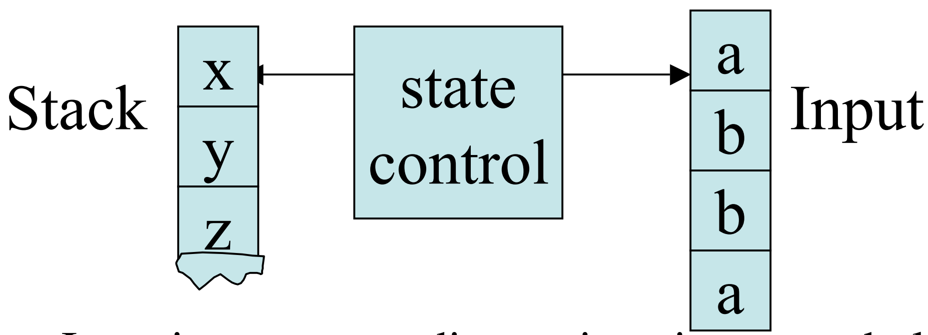
CS154
Chris Pollett
Mar 19, 2007.

Outline

- Pushdown Automata
- Equivalence

Pushdown Automata

- Our goal is a machine model corresponding CFG. This might help to develop parsers.
- To do this we will consider machines that have a stack:



- In a given state reading a given input symbol and a given stack symbol, the machine can switch states, advance to the next character of the input, pop the top symbol off the stack, or push a new symbol onto the stack.
- For instance, the language $\{0^n1^n \mid n>=0\}$ could be recognized by such a machine. When one reads an 0 push it onto the stack. When one starts reading 1's, if one ever sees another 0 reject, also start popping 0's off of the stack. If when one gets to the end of the string the stack is empty, then accept.

Formal Definition

- A **pushdown automaton** is a 7-tuple $M=(Q, \Sigma, \Gamma, \delta, q_0, z, F)$ where
  1. Q is the set of states
  2. Σ is the input alphabet
  3. Γ is the stack alphabet
  4. $\delta: Q \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \text{-->} 2^{(Q \times (\Gamma \cup \{\lambda\}))}$ is the transition function
  5. $q_0 \in Q$ is the start state, and
  6. $z \in \Gamma$ is the start of stack symbol
  7. $F \subseteq Q$ is the set of accept states.
- M ***accepts*** $w= w_1w_2\ldots w_n$ where each $w_i \in \Sigma \cup \{\lambda\}$ if there is a sequence of states $r_0, r_1, \ldots, r_m$ in Q and a sequence of strings $s_0, s_1, \ldots, s_m$ in Γ^* such that (1) $r_0= q_0$, $s_0= z$, (2) for $i =0,\ldots, m-1$, we have $(r_{i+1},b) \in \delta(r_i, w_{i+1}, a)$ where $s_i = at$ and $s_{i+1}= bt$ for some $a,b \in \Gamma \cup \{\lambda\}$ and $t \in \Gamma^*$, and (3) $r_m \in F$.

Remarks on the Definition

- Notice the machine is a generalization of an NFA not a DFA.
- One can show deterministic pushdown automata are a strictly weaker then nondeterministic pushdown automata.

Example

- We can define a machine to recognize $\{0^n 1^n \mid n>=0\}$ as $M=(Q, \Sigma, \Gamma, \delta, q_1, \$, F)$ where:

  $Q=\{q1, q2, q3, q4\}$

  $\Sigma=\{0,1\}$

  $\Gamma=\{0,\$\}$

  $F=\{q1, q4\}$

  and $\delta=\{(q1, \lambda, \lambda)\text{->}(q2, \$),$

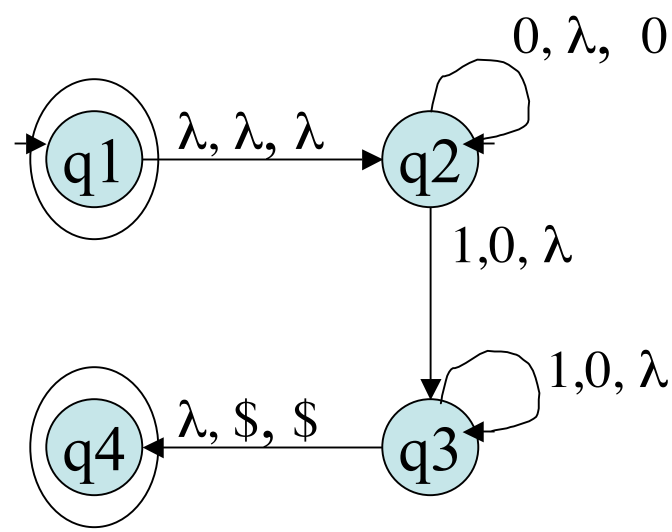
  $(q2, 0, \lambda)\text{-->}(q2, 0),$

  $(q2, 1, 0)\text{-->}(q3, \lambda),$

  $(q3, 1, 0)\text{-->}(q3, \lambda)$

  $(q3, \lambda,\$)\text{-->}(q4, \lambda)$

  $\}$



- Can then using the definition show this machine accepts 0011.

Equivalence

- We now works towards showing a language is context free if and only if some pushdown automata recognizes it.
- The proof split into two parts:
  - If a language is context-free then some pushdown automata recognizes it
  - If a pushdown automata recognizes some language then there is a context-free grammar that recognizes the same language.

CFL=> PDA recognizes

- Let A be a CFL. Let G be a CFG for this language, and let w be a string generated by G (and hence in A). We will have a machine with three main states $\{q_{start}, q_{loop}, q_{accept}\}$ together with some auxiliary states E.

1. We have transitions $(q_{start}, \lambda, \lambda) \dashrightarrow (q_{loop}, S)$ that push the start variable S of the CFG onto our machine's stack.
2. Then what we want to do is to simulate the steps to generate w on our PDAs stack.
   a) If A is a variable of the CFG on the top fo the stack, and we are in the state q_{loop} we nondeterministically choose a rule $A \rightarrow w_1 w_2 .. w_n$ and using a sequence of transitions $(q_{loop}, \lambda, A) \dashrightarrow (q_1, w_n)$, $(q_1, \lambda, \lambda) \dashrightarrow (q_2, w_{n-1})$ … $(q_n, \lambda, \lambda) \dashrightarrow (q_{loop}, w_1)$ We simulate this rule on the stack. Here q_i are some of the auxiliary states in E.
   b) To handle a terminal such as b on the top of the stack we have transitions $(q_{loop}, b, b) \dashrightarrow (q_{loop}, \lambda)$.
3. Finally, we have a transition $(q_{loop}, \lambda, \$) \dashrightarrow (q_{accept}, \$)$ where q_{accept} is our accept state.