Theory of Computation
Lecture 6: Pushdown Automata

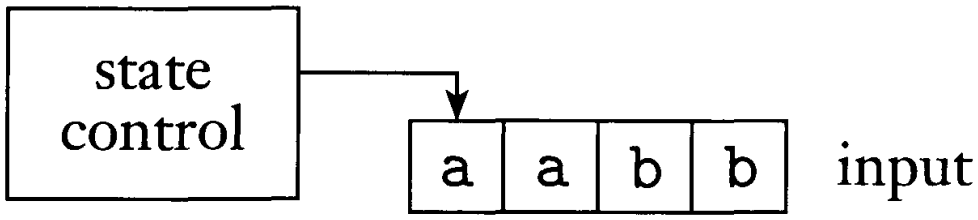
Dr. Nahla Belal

Pushdown Automata

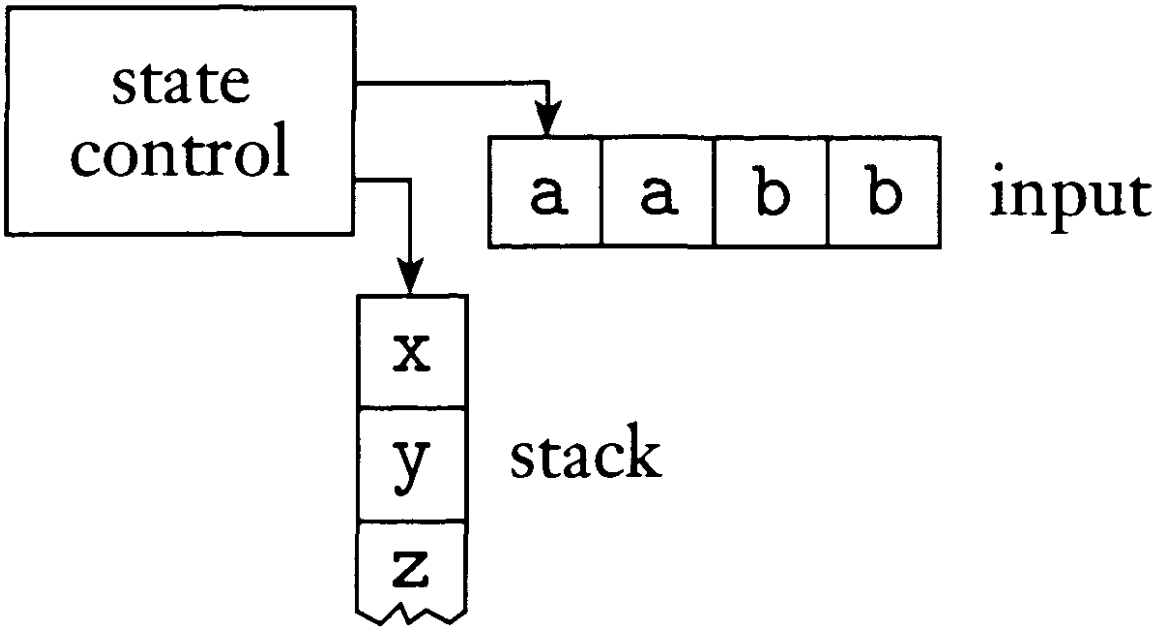
- Similar to NFAs but with a stack for extra memory.
- The stack allows the recognition of some non-regular languages.
- Equivalent in power to context-free grammars.
- Two options to prove that a language in context free, give a grammar that generates it or a pushdown automata that recognizes it.

Schematic of a Finite Automaton

The following figure is a schematic representation of a finite automaton. The control represents the states and transition function, the tape contains the input string, and the arrow represents the input head, pointing at the next input symbol to be read.



Schematic of a Pushdown Automaton (PDA)



Formal Definition

A ***pushdown automaton*** is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where Q, Σ, Γ, and F are all finite sets, and

1. Q is the set of states,
2. Σ is the input alphabet,
3. Γ is the stack alphabet,
4. $\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function,
5. $q_0 \in Q$ is the start state, and
6. $F \subseteq Q$ is the set of accept states.

Example 1: $\{0^n 1^n | n \geq 0\}$

- Finite automata unable to recognize this language.
- Stack in PDA allows for unbounded memory.

Read symbols from the input. As each 0 is read, push it onto the stack. As soon as 1s are seen, pop a 0 off the stack for each 1 read. If reading the input is finished exactly when the stack becomes empty of 0s, accept the input. If the stack becomes empty while 1s remain or if the 1s are finished while the stack still contains 0s or if any 0s appear in the input following 1s, reject the input.

Example 1 (cont.)

$Q = \{q_1, q_2, q_3, q_4\}$,

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

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

$F = \{q_1, q_4\}$, and

δ is given by the following table, wherein blank entries signify $\emptyset$.

Input:	0			1			ε		
Stack:	0	\$	ε	0	\$	ε	0	\$	ε
q_1									$\{(q_2, \$)\}$
q_2			$\{(q_2, 0)\}$	$\{(q_3, \varepsilon)\}$					
q_3				$\{(q_3, \varepsilon)\}$				$\{(q_4, \varepsilon)\}$	
q_4									

State Diagram for a PDA

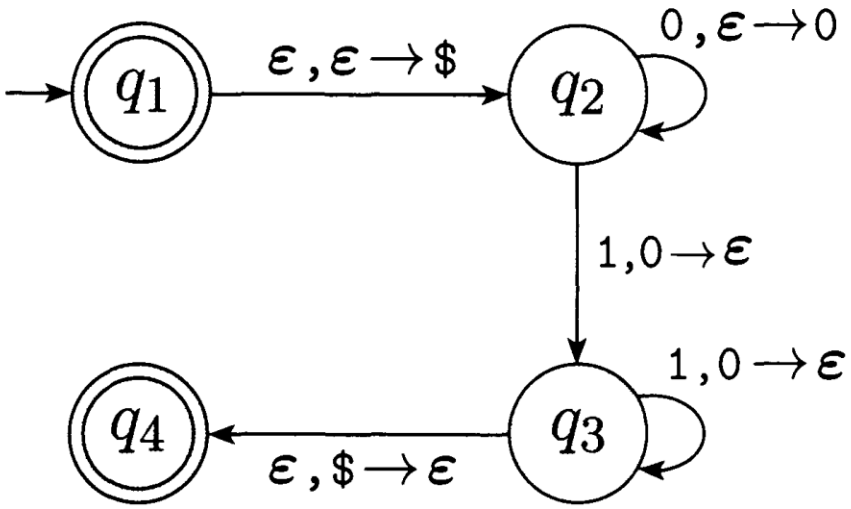


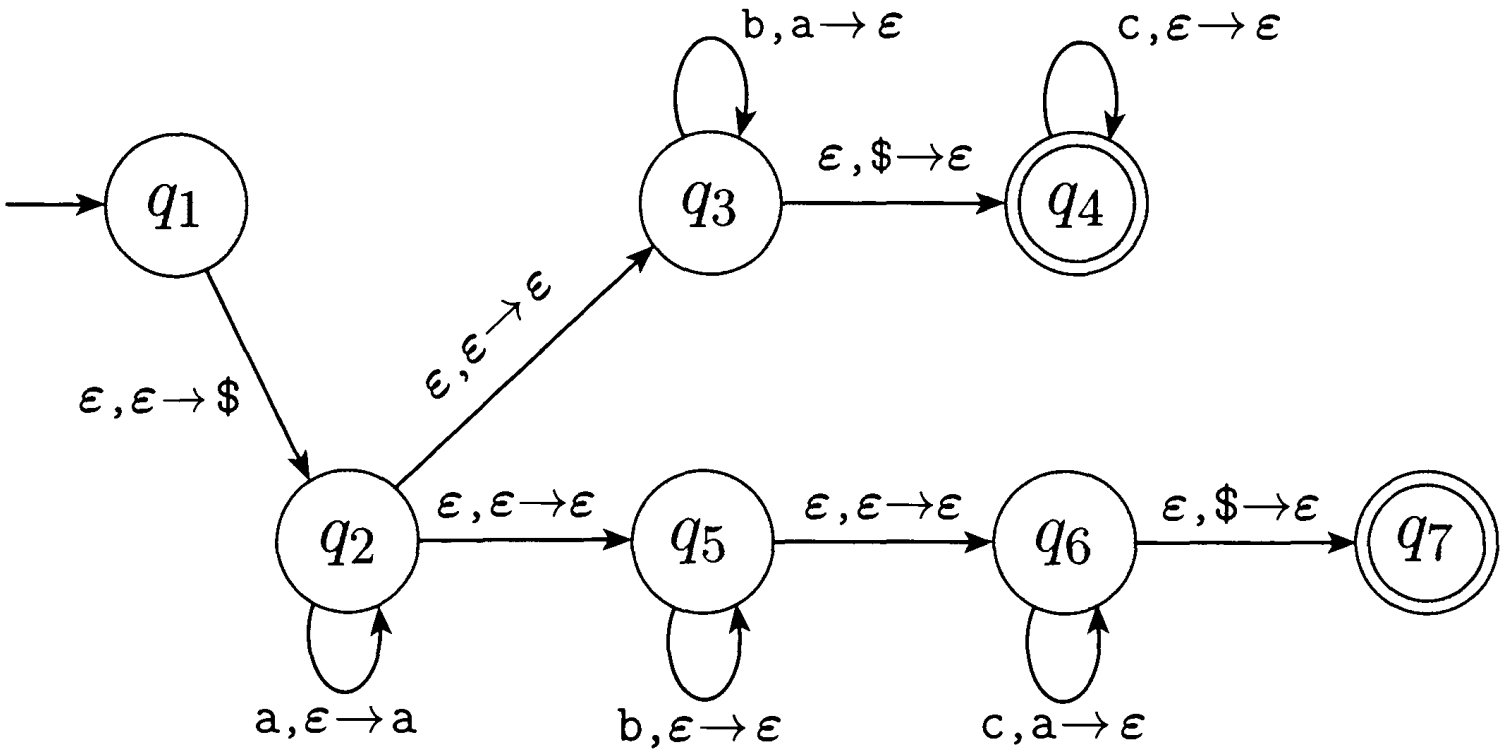
FIGURE 2.15
State diagram for the PDA M_1 that recognizes $\{0^n 1^n \mid n \geq 0\}$

Example 2: $\{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$.

Informally the PDA for this language works by first reading and pushing the a's. When the a's are done the machine has all of them on the stack so that it can match them with either the b's or the c's. This maneuver is a bit tricky because the machine doesn't know in advance whether to match the a's with the b's or the c's. Nondeterminism comes in handy here.

Using its nondeterminism, the PDA can guess whether to match the a's with the b's or with the c's, as shown in the following figure. Think of the machine as having two branches of its nondeterminism, one for each possible guess. If either of them match, that branch accepts and the entire machine accepts. In fact we could show, though we do not do so, that nondeterminism is *essential* for recognizing this language with a PDA.

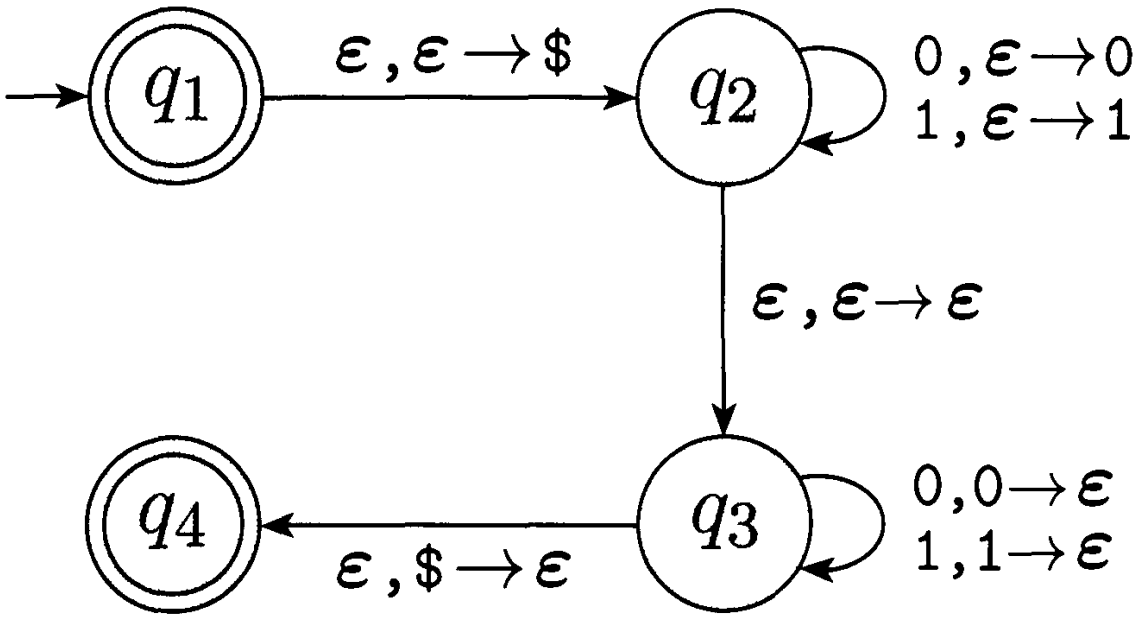
Example 2: State Diagram



Example 3: $\{ww^{\mathcal{R}} | w \in \{0,1\}^*\}$

Begin by pushing the symbols that are read onto the stack. At each point nondeterministically guess that the middle of the string has been reached and then change into popping off the stack for each symbol read, checking to see that they are the same. If they were always the same symbol and the stack empties at the same time as the input is finished, accept; otherwise reject.

Example 3: State Diagram



Reading Materials

- Sipser: Chapter 2, Section 2.2