Pushdown Automata: Examples

First off a few things looking ahead…

- Complete check-in survey (HW5)
- We are in **Week 9** this week
  - Today (03/07): examples of PDA
  - Wed 03/09: CFG to PDA Conversion
    - Possible hints on remaining difficult problems on HW5
  - Fri 03/11: conclude CFL topics
- Next week is **Spring Break**
- We will have Mid-term 2 in **Week 11** (3/23 or 3/25)
  - The exam will be only on CFL (Chapter 2 of Sipser)
  - No HW will be issued in Week 11
- Starting from **Week 12**
  - We will be dealing with **Computability Theory** (Turing Machines, Decidability, Reducibility)

Pushdown Automata (PDA)

- PDA generalize nondeterministic finite automata (NFA) but have an extra component called a stack

- The stack provides additional memory beyond that available in NFA

- The stack allows PDA to recognize some nonregular languages

- PDA are equivalent in power to context-free grammars
    - This gives us two options for proving that a language L is context free:
    1) Give a CFG generating L
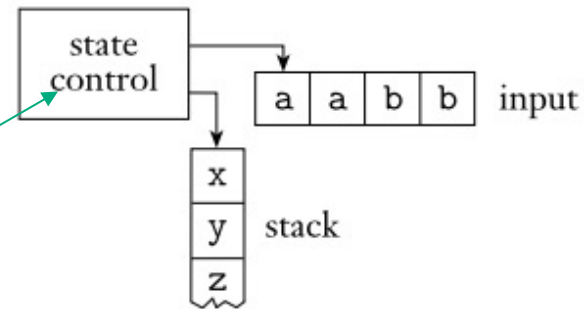   2) Give a PDA recognizing L

NFA vs PDA schematic

Regular languages
- NFA
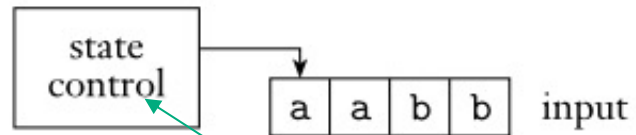
Context-free languages
- PDA

state control:
states and transition function

PDA

- A PDA can write symbols on the stack and read them back later
- Writing a symbol "pushes down" all the other symbols on the stack
- At any time, the symbol on the top of the stack can be read and removed.
  - The remaining symbols then move back up

- Writing a symbol on the stack is referred to as **pushing** the symbol, and removing a symbol is referred to as **popping** it.

- A stack is a "Last In First Out" (LIFO) storage device

Revisiting the language $B = \{0^n 1^n \mid n \geq 0\}$

- Recall that a finite automaton is unable to recognize B because it cannot store very large numbers (unbounded information) in its finite memory

- A PDA is able to recognize this language because it can use its stack to store the number of 0s it has seen

- Thus, the unlimited nature of a stack allows the PDA to store numbers of unbounded size

Revisiting the language $B = \{0^n1^n \mid n \geq 0\}$

Informal description of how PDA for the language B works

- 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**

- If

  - stack becomes empty while 1s remain OR

  - the 1s are finished while the stack still contains 0s OR

  - any 0s appear in the input following 1s

  then **reject** the input

Formal definition of PDA: the ingredients

- Similar to NFA, except for the stack

- The machine may use different alphabets for its input and stack, so we now have an input alphabet Σ and a stack alphabet Γ

- Recall that $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ and $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$
- Domain of the **transition function**: $Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon$
- Range of the **transition function**: $Q \times \Gamma_\varepsilon$
- **Nondeterminism** allowed: transition function δ returns a member of $P(Q \times \Gamma_\varepsilon)$

Formal definition of PDA: put together

DEFINITION 2.13

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. $\delta \colon Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function,
5. $q_0 \in Q$ is the start state, and
6. $F \subseteq Q$ is the set of accept states.

How a PDA computes

- A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ computes as follows.

- It accepts input w if w can be written as $w = w_1 w_2 ... w_m$, where each $w_i \in \Sigma_\varepsilon$ and sequence of states $r_0, r_1, \ldots r_m \in Q$ and strings $s_0, s_1, \ldots, s_m \in \Gamma^*$ exist that satisfy the following three conditions. The strings s_i represent the sequence of stack contents that M has on the *accepting branch* of the computation.

  1. $r_0 = q_0$ and $s_0 = \varepsilon$
  2. For $i = 0, \ldots, m\text{-}1$, 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 $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$
  3. $r_m \in F$

Example 1:
Design PDA for the language B = $\{0^n 1^n \mid n \geq 0\}$

A couple of notations before presenting the PDA
1) **Testing for empty stack**
   - The formal definition of a PDA contains no explicit mechanism to allow the PDA to test for an empty stack
   - The same effect can be attained by initially putting a special symbol (e.g., $) on the stack
2) **Short-hand notation**
   - for (state-input-stack) transitions we write $a, b \rightarrow c$ to signify that when the machine is reading an a from the input, it replaces the symbol b on the top of the stack with a c
   - any of a, b, and c may be ε
     - $a = \varepsilon$ means machine may make this transition without reading any symbol from the input
     - $b = \varepsilon$ means machine may make this transition without reading & popping any symbol from the stack
     - $c = \varepsilon$ means machine does not write any symbol on the stack when going along this transition
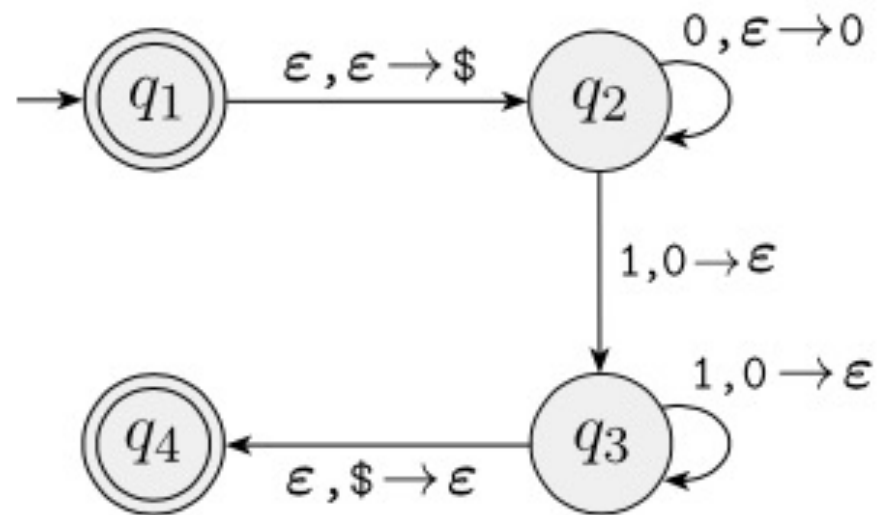
Example 1:
PDA for the language $B = \{0^n 1^n \mid n \geq 0\}$

How would you go about designing a PDA for B?

How many states would you need?

Example 1:
PDA for the language B = {$0^n 1^n \mid n \geq 0$}

$\rightarrow (q_1) \xrightarrow{\varepsilon, \varepsilon \rightarrow \$} (q_2)$ with self-loop $0, \varepsilon \rightarrow 0$

$q_2 \xrightarrow{1, 0 \rightarrow \varepsilon} q_3$ with self-loop $1, 0 \rightarrow \varepsilon$

$q_3 \xrightarrow{\varepsilon, \$ \rightarrow \varepsilon} q_4$

Example 2

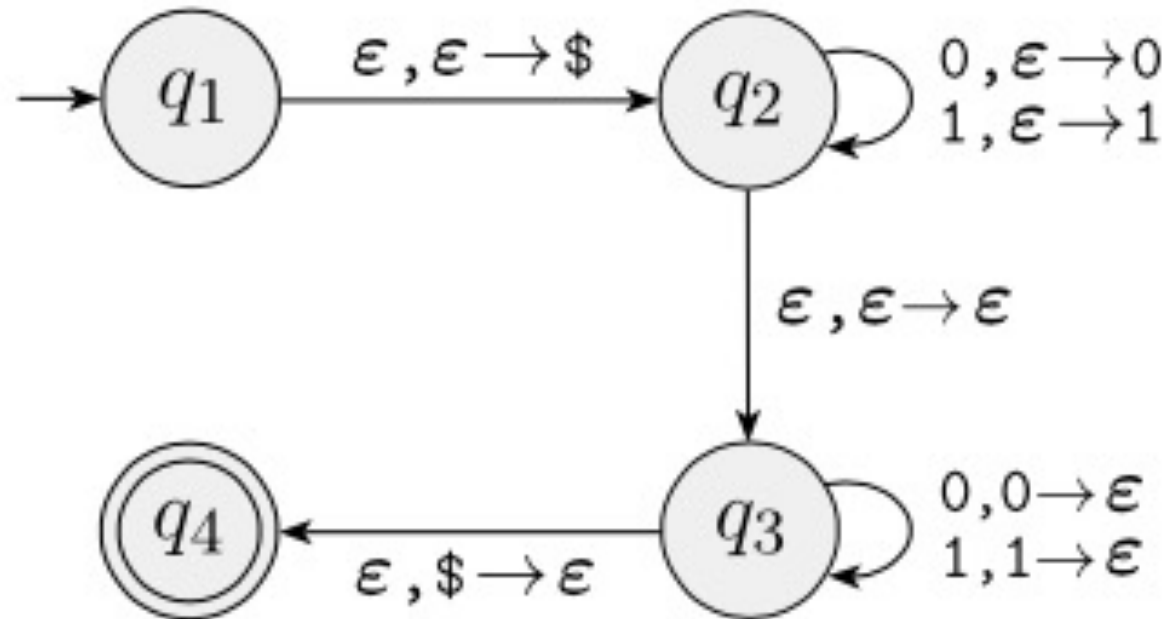Design a PDA that recognizes the language

$$\{ww^R \mid w \; \{0, 1\}^*\}$$

Recall that w^R means w written backwards

Note that this language is a palindrome

How would you go about designing the PDA?

Example 2 (palindrome)

Example 3

- Design a PDA that recognizes the language

$$D = \{a^i\, b^j\, c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } i = k\}$$

- How would you go about designing the PDA?

Example 3 (PDA for language D)