Pushdown Automata - Introduction

Lecture 15
Section 2.2

Robb T. Koether
Hampden-Sydney College

Wed, Oct 1, 2014
Outline

1. Assignment
2. Machines for CFGs
3. Acceptance Mode
4. Equivalence of Acceptance Modes
5. Pushdown Automata
6. An Example
1. Assignment
2. Machines for CFGs
3. Acceptance Mode
4. Equivalence of Acceptance Modes
5. Pushdown Automata
6. An Example
Assignment

Chapter 2: Exercises 5, 7, 10.
Given a regular language, we can describe it by using a regular grammar or a machine (a DFA).

A context-free language can be described by using a context-free grammar.

Can it also be described by a machine?
Machines for CFLs

- If a machine is to process the string $a^n b^n$, then it must be able to “remember” the number of $a$’s.
- We will use a stack to do this.
- As each $a$ is read, push it onto the stack.
- When $b$ is read, pop an $a$.
- When we are finished reading the string, the stack should be empty.
Each transition will include three parts.
- The symbol read.
- The symbol popped.
- The symbol pushed.

Any of these could be $\varepsilon$.

In other words, PDAs are inherently nondeterministic.
Machine for \( \{a^n b^n \mid n \geq 0\} \)

Example (Machine for \( \{a^n b^n \mid n \geq 0\} \))

First attempt
This machine has one shortcoming.

There are several ways that we could define acceptance of a string.

They all involve the final “state” of the machine.

But the “state” of the machine includes the state that it is in together with the contents of the stack.
Outline

1. Assignment
2. Machines for CFGs
3. Acceptance Mode
4. Equivalence of Acceptance Modes
5. Pushdown Automata
6. An Example
Acceptance of Strings

**Definition (Acceptance by final state)**

A machine accepts by final state if the input is accepted only if the last state is a final, or accepting, state.

**Definition (Acceptance by empty stack)**

A machine accepts by empty stack if the input is accepted only if the stack is empty once the last symbol is read.

**Definition (Acceptance by final state and empty stack)**

A machine accepts by final state and empty stack if the input is accepted only if the last state is a final state and the stack is empty once the last symbol is read.
Acceptance of Strings

- We will use acceptance by final state.
- Thus, our machine actually accepts

\[ \{a^n b^m \mid n \geq m \geq 0 \}. \]

- What if we had accepted by empty stack?
- What if we had accepted by final state and empty stack?
Testing for an Empty Stack

Definition

A **bottom marker** is a unique symbol in the stack alphabet that is placed at the bottom of the stack.

- In this example, we need to check that the stack is empty after reading the last symbol.
- To do this, we will use a bottom marker on the stack.
- Let $ be the bottom marker.
Testing for an Empty Stack

- At the beginning, push $ onto the stack.
- At the end, pop $ off the stack.
- That guarantees that the stack is empty.
Machine for \( \{a^n b^n \mid n \geq 0\} \)

Example (Machine for \( \{a^n b^n \mid n \geq 0\} \))

\[
\begin{align*}
\epsilon, \epsilon & \rightarrow \epsilon \\
a, \epsilon & \rightarrow a \\
b, a & \rightarrow \epsilon \\
\epsilon & \rightarrow \epsilon \\
\epsilon & \rightarrow \epsilon \\
\epsilon, \$ & \rightarrow \epsilon
\end{align*}
\]
Outline

1. Assignment
2. Machines for CFGs
3. Acceptance Mode
4. Equivalence of Acceptance Modes
5. Pushdown Automata
6. An Example
Theorem (Equivalence of acceptance modes)

The following modes of acceptance are all equivalent.

- Acceptance by final state.
- Acceptance by empty stack.
- Acceptance by both final state and empty stack.

That is, if a machine uses any one of these acceptance modes, then it can be modified into an equivalent machine that uses either one of the other acceptance modes.
Acceptance Modes

By empty stack ⇒ By final state.

Accept by empty stack
Acceptance Modes

By empty stack ⇒ By final state.

Accept by final state
Acceptance Modes

By final state $\Rightarrow$ By final state and empty stack.

Accept by final state
Acceptance Modes

By final state ⇒ By final state and empty stack.

Accept by final state and empty stack
Acceptance Modes

By final state and empty stack \(\Rightarrow\) By empty stack.

Accept by final state and empty stack
Acceptance Modes

By final state and empty stack $\Rightarrow$ By empty stack.

Accept by empty stack
A **pushdown automaton**, abbreviated PDA, is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where

- $Q$ is a finite set of states.
- $\Sigma$ is a finite input alphabet.
- $\Gamma$ is a finite stack alphabet.
- $\delta : Q \times \Sigma \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma)$ is the transition function.
- $q_0 \in Q$ is the start state.
- $F \subseteq Q$ is the set of accept states.
Example

Example (PDA for \( \{a^n b^n \mid n \geq 0\} \))

- The PDA that accepts \( \{a^n b^n \mid n \geq 0\} \) is

  - \( Q = \{q_0, q_1, q_2, q_3\} \)
  - \( \Sigma = \{a, b\} \)
  - \( \Gamma = \{a, \$\} \)
  - \( F = \{q_3\} \)
Example (PDA for \( \{ a^n b^n \mid n \geq 0 \} \))

and \( \delta \) is given by

- \( \delta(q_0, \varepsilon, \varepsilon) = \{ (q_1, \$) \} \)
- \( \delta(q_1, a, \varepsilon) = \{ (q_1, a) \} \)
- \( \delta(q_1, \varepsilon, \varepsilon) = \{ (q_2, \varepsilon) \} \)
- \( \delta(q_2, b, a) = \{ (q_2, \varepsilon) \} \)
- \( \delta(q_2, \varepsilon, \$) = \{ (q_3, \varepsilon) \} \)
Outline

1. Assignment
2. Machines for CFGs
3. Acceptance Mode
4. Equivalence of Acceptance Modes
5. Pushdown Automata
6. An Example
Example (Pushdown automaton)

- Design a PDA that accepts the language

\[ \{ w \mid w \text{ contains an equal number of a’s and b’s} \}. \]
Example (Pushdown automaton)

- The strategy will be to keep the excess symbols, either $a$’s or $b$’s, on the stack.
- One state will represent an excess of $a$’s.
- Another state will represent an excess of $b$’s.
- We can tell when the excess switches from one symbol to the other because at that point the stack will be empty.
- In fact, when the stack is empty, we may return to the start state.
Example (Pushdown automaton)

- $a, \varepsilon \rightarrow a$
- $b, a \rightarrow \varepsilon$
- $a, \varepsilon \rightarrow \$
- $b, \varepsilon \rightarrow \$
- $b, \$, $\rightarrow \varepsilon$
- $a, \$, $\rightarrow \varepsilon$
- $a, b \rightarrow \varepsilon$
- $b, \varepsilon \rightarrow b$

States:
- $a > b$
- $a < b$
Example (Pushdown automaton)

Note that this solution is inspired by the grammar

\[ S \rightarrow SS \mid aSb \mid bSa \mid \varepsilon \]