Pushdown Automata - Introduction
Lecture 17
Section 2.2

Robb T. Koether

Hampden-Sydney College

Fri, Oct 3, 2008
Outline

1. Collected Problems
2. Homework Review
3. Machines for CFGs
4. Acceptance Mode
5. Equivalence of Acceptance Modes
6. Pushdown Automata
7. Assignment
Collected Problems

Due Mon, Oct 6, 2008

- Page 83, Exercises 18bfi, 29b.
- Page 83, Problem 48.
- Page 128, Exercises 4e, 16.
Exercise 2.6, page 129.

Give a context-free grammar generating the set of strings over the alphabet \{a, b\} with more a’s than b’s.
Given such a string, one may single out certain $a$’s that separate substrings with an equal number of $a$’s and $b$’s.

For example, the string $aaabaabbbbaaabaa$ could be viewed as $a|a|ab|a|abbb|a|ba|a$ (one possibility).

The “balanced” substrings are generated by the grammar

$$X \rightarrow XX \mid aXb \mid bXa \mid \varepsilon$$
Solution

Thus, the original string may be broken up into a substring of the form $X$ followed by a sequence of one or more substrings of the form $aX$.

This gives us the grammar

$$S \rightarrow XT$$
$$T \rightarrow aXT \mid aX$$
$$X \rightarrow XX \mid aXb \mid bXa \mid \varepsilon$$
Grammars vs. Machines

- 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?
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 $a$. 
Machines for CFLs

- Each transition will include three parts.
  - The symbol read.
  - The symbol popped.
  - The symbol pushed.
- Any of these could be $\epsilon$. 
Machine for $\{a^n b^n \mid n \geq 0\}$

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

- First attempt:

```
a, \epsilon \rightarrow a
b, a \rightarrow \epsilon
\epsilon, \epsilon \rightarrow \epsilon
```
Machine for $\{a^n b^n \mid n \geq 0\}$

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

- 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.
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

- 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.

**Definition**

A bottom marker is a unique symbol in the stack alphabet that is placed at the bottom of 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 $\{a^n b^n \mid n \geq 0\}$)

\[
\begin{align*}
&\text{a, } \varepsilon \rightarrow \text{a} \\
&\text{b, } \text{a} \rightarrow \varepsilon \\
&\varepsilon, \varepsilon \rightarrow \$ \\
&\varepsilon, \varepsilon \rightarrow \varepsilon \\
&\varepsilon, \$ \rightarrow \varepsilon
\end{align*}
\]
Acceptance Modes

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 $\Rightarrow$ 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

- $\varepsilon, \varepsilon \rightarrow \varepsilon$
- $\varepsilon, a \rightarrow \varepsilon$
Acceptance Modes

By final state and empty stack ⇒ 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

$\varepsilon, \varepsilon \rightarrow \$ \Rightarrow \varepsilon, \$ \rightarrow \varepsilon$
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**.
- $\delta : Q \times \Sigma^\epsilon \times \Gamma^\epsilon \rightarrow \mathcal{P}(Q \times \Sigma^\epsilon)$ is the **transition function**.
- $q_0 \in Q$ is the **start state**.
- $F \subseteq Q$ is the set of **accept states**.
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, \epsilon, \epsilon) = \{(q_1, $)\}$
- $\delta(q_1, a, \epsilon) = \{(q_1, a)\}$
- $\delta(q_1, \epsilon, \epsilon) = \{(q_2, \epsilon)\}$
- $\delta(q_2, b, a) = \{(q_2, \epsilon)\}$
- $\delta(q_2, \epsilon, $) = \{(q_3, \epsilon)\}$
Assignment

Homework

- Read Section 2.2, pages 109 - 112.
- Exercises 5, 7, 10, page 129.