Cpt 3/17

Mon Feb 24

Pushdown Automata
Example 3

Design a PDA that recognizes the language

\[ L = \left\{ a^i b^j c^k \right\} \quad i, j, k \geq 0 \text{ and } \ 
i = j \text{ or } i = k \]

Informally, the PDA for \( L \) comes by first reading and pushing the a's.

When we are done with a's, the machine has all of them on stack. It can match up with either b's or c's.
- The machine does not know in advance whether to match the a's with the b's or with the c's.

- Solution: Non-determinism
\[ L_3 = \{ a^i b^j c^k \mid i, j, k \geq 0 \} \]
\[ i = j \quad \text{or} \quad i = k \]
Theorem

A language is context free if and only if some pushdown automaton recognizes it.

We will look at the proof idea for this theorem.

We first break it into its if- part and only if part.
Lemma

If a language is Context free, then some pushdown automaton recognizes it.

Proof Idea

Let $A$ be a CFL.

By definition, $A$ has a CFG $G$ that generates it.

We want to convert $G$ into an equivalent PDA, which we call $P$. 
P will accept its input w by determining whether there is a derivation for w.

Each step of the derivation yields an intermediate string of variables and terminals.

We design P to determine whether some series of substitutions using the rules of G can lead from start variable to w.
- One of the difficulties in testing whether there is a derivation for co is figuring out which rule to use.

- Solution: The PDA's non-determinism.

At each step of the derivation, one of the rules for a particular variable in detected non-deterministically and used to substitute for that variable.
- The PDA began by peeling the stack one card at a time.

- It goes through a series of intermediate states, making one substitution after another.

- Eventually, it may arrive at a stack that contains only terminal symbols, meaning it has used the grammar to derive a string.

- Then it accepts if the string is identical to the string it has received as input.
Implementing this strategy on PDA requires one additional idea.

We need to see how the PDA stores the intermediate strings as it goes from one to another.

Simply using the stack for storing intermediate variables won't work because the stack is LIFO (we have access only to the top of the stack).
One solution is to keep only part of the intermediate string in the stack - the symbols starting with the first variable in the intermediate stack.

Any terminal symbols appearing before the first variable are matched immediately with the symbols in the input string.

```
Control

011001

01 A1 A0

intermediate string 01A1A0
```
The following is an informal description of P

1. Place the marker symbol $\delta$ and the start variable on the stack.

2. Repeat the following forever:

(a) If the top of the stack is a variable symbol $A$, non-deterministically select one of the rules for $A$ and substitute $A$ by the string on the RHS of the rule.
b) If the top of the stack is terminal symbol a, read the next symbol from the input and compare with a. If they match, repeat.

If they don't, reject or return to the branch of non-determinism.

c) If the top of the stack is $\$, insert an accept state. Doing so allows the input of it being accepted.