CSE 105—Theory of Computability

Lecture 8—April 27, 2006
Pushdown Automata

Instructor: Neil Rhodes
What is a pushdown automata?

A Finite-state automata augmented with a stack

Stack:
- Holds stack symbols (stack alphabet distinct from input alphabet)
- Can pop symbol from the stack
  - popping empty stack causes this computation to not accept
  - Can only retrieve topmost symbol
- Can push a stack symbol
  - Always goes on top
- No way to explicitly test whether stack is empty
  - But we’ve got a trick to be able to tell!

State diagram
- labels become: a, b→c
  - means
    - reading a from input
    - and top of stack is b
    - pop b
    - push c
Example

Language = \{0^n1^n | n \geq 0\}
Example

Language = \{w \in \{0, 1\}^* | w \text{ has equal numbers of 0's and 1's}\}
Example

Language = \{w#w^R| w \in \{0, 1\}^*\}
Example

Language = \{ww^R \mid w \in \{0, 1\}^*\}
Example

Language = w not a palindrome (over \{0, 1\})
Example

Language: \( L\{xy \mid x \neq y, x, y \in \{0, 1\}* \} \)
CFG can be converted to NPDA

Easiest to use Chomsky Normal Form
- Book uses any grammar without conversion; idea is the same

Let’s look at leftmost derivation with CNF
- Example grammar
  - S→TT | RT
  - T→0 | TT | 1
  - R→0 | RR

- Example string
  - 01101
CFG can be converted to NPDA

Without Chomsky Normal Form

- Can have arbitrary mix of terminals and non-terminals in sentential form
- Store everything except leading terminals on stack
- Match input symbols to stack terminal symbols
- Example:
  - S→0T1 ∣ 1
  - T→T0 ∣ ε
- Example string:
  - 0001
CFG equivalent to PDA

Still need to show can convert PDA to CFG
  - Believe me:) (In book, if you desire)