ECS120 Introduction to the Theory of Computation
Fall Quarter 2007

Discussion Notes
Wednesday, Due Wednesday November 14, 2007, 2007

Due to the upcoming holiday, this homework will be due on Wednesday, November 14th.

Problem 6.1

Find a decision procedure which determines if a given CFG (with alphabet $\{a, b\}$) accepts infinitely many strings which contain exactly 3 a's. You can assume that you have procedures that can convert PDAs into CFGs and CFGs into PDAs.

This is a similar to problem 1 on homework 5. After this discussion, you should know everything necessary for this decision procedure.

Problem 6.2

Design a context-sensitive grammar for the language $\{a^n b^n c^n \mid n \geq 0\}$ with 10 or fewer rules. What does your grammar imply about the relationship between CFLs and CSLs?

From Wikipedia:
A context-sensitive grammar (CSG) is a formal grammar in which the left-hand sides and right-hand sides of any production rules may be surrounded by a context of terminal and nonterminal symbols.

Example rules in a CSG may look like:

$$\begin{aligned} S &\rightarrow aAb \\ Ab &\rightarrow aaAbb \\ A &\rightarrow a \end{aligned}$$

And a derivation may look like:

$$S \Rightarrow aAb \Rightarrow aaaAbb \Rightarrow aaaabb$$

Hints: Be sure to look at the formal definition to make sure you provide valid rules. Do not copy CSGs from Wikipedia (or other sources). You need to consider $n \geq 0$, and must provide a grammar with less than 10 rules.

Problem 6.3

Show that a k-stack PDA (or k-PDA) for $k \geq 2$ is equivalent to a Turing Machine.

You still must be formal when showing this is true!

Prove both directions separately.

1. Given a k-PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$, can you build a Turing machine (or known TM equivalent) to simulate P?
2. Given a Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$, can you build a k-PDA (or just 2-PDA) to simulate M?

Problem 6.4

Specify fully a Turing Machine for the language $\{ ww^{\mathcal{R}} \mid w \in \{a, b\}^* \}$.

Draw it out!

Problem 6.5

A Stay-put Turing Machine is defined as a TM which after reading/writing to a tape cell can move the tape head either left, right, or leave it in the same cell. Show that a Stay-put TM is equivalent to a TM.

Prove both directions.

1. Can a Stay-put Turing machine simulate a regular Turing machine? (Formally show how.)
2. Can a regular Turing machine simulate a Stay-put Turing machine? (Formally show how.)

HINTS: Consider adding states to the finite control.