

## Problem Set 1

---

Handout written by David Underhill.

Due: Wed 09-July-2008

### Instructions:

- **Show and explain your work.** Do this whenever possible. It allows us to better characterize your understanding of the material - and better grade your hard work.
- **DFA transitions.** You must include a transition out of every state for every symbol in the alphabet for a DFA.
- **Submission.** You may submit your work electronically via the same submission script you used for the first project. *We prefer electronic submission* (scans ok), but will accept hard copies during office hours or via the SCPD courier.

### Problem 1 The Loneliest Regular Expression

Construct a regular expression over the alphabet  $\{c, s\}$  such that no two adjacent letters are identical. Make your regular expression as compact as you can.

### Problem 2 Nothing Ever Changes

- Take your regular expression from Problem 1 and construct a  $\lambda$ -NDFFA using the rule from Thompson's construction. Show your work step-by-step and indicate which rule you apply at each step in order to get to next one.
- Now that you have an  $\lambda$ -NDFFA, convert it to an equivalent DFA. You should first remove any  $\lambda$ -cycles (if any), then remove any  $\lambda$ -transitions, and finally remove any remaining non-determinism. Show each of these stages (e.g. no  $\lambda$ -cycles, no  $\lambda$ -transitions, and the final product).
- Minimize the number of states in your DFA (or prove that it already has the minimal number of states). Show your work.

**Problem 3 Your Instructor Has Lots of Problems**

- a. Show that the following grammar is ambiguous, or prove that it is not ambiguous.

**Non-Terminals:**  $\{S, E, T, I, Op\}$

**Terminals:**  $\{=, +, d, g, o\}$

**Productions:**

$S \rightarrow E$

$E \rightarrow E Op E$

$E \rightarrow I$

$Op \rightarrow = \mid +$

$I \rightarrow T \mid IT$

$T \rightarrow d \mid g \mid o$

- b. Explain why this grammar is not an LL(1) grammar. Rewrite the grammar so that it could be parsed using an LL(1) parser.
- c. Generate a top-down predictive parser's parse table for the LL(1) grammar you just created.
- d. Trace through a parse for the following input with your predictive parser's table:  
g=o+dog

**Problem 4 When You Get to the Bottom of the Hole, Stop Digging**

- a. Have you visited the newsgroup?
- b. Do you have any remaining technical issues which are preventing you from being able to access the newsgroup?
- c. Do you have any general feedback regarding the course format or assignments thus far?