ECS120 Introduction to the Theory of Computation
Fall Quarter 2007

# Homework 3 Help
## Due Friday October 19, 2007

## Problem 3.1

Use the procedure described in Lemma 1.55 to convert the following regular expressions to nondeterministic finite automata.

**a**. $(0 \cup 1)^*000(0 \cup 1)^*$

**b**. $(((00)^*(11)) \cup 01)^*$

**c**. $\emptyset^*$

Follow the algorithms described in class or in the book to create the NFAs for the given regular expressions. It is more important to provide a **correct** NFA versus a minimal NFA, so just focus on following the algorithm correctly.

**Answers from JFLAP or without work will not be accepted!**

## Problem 3.2

Let $\Sigma = \{0, 1\}$ and let

$$D = \{w \mid w \text{ contains an equal number of occurrences of the substrings 01 and 10}\}$$

Thus $101 \in D$ because 101 contains a single 01 and a single 10, but $1010 \notin D$ because 1010 contains two 10s and one 01. Show that $D$ is a regular language.

Try listing out several possibilities, and then create a regular expression from there. You may want to *test* your regular expression in JFLAP to be sure. To show it is regular, well... that part should be easy if you do the first part!

## Problem 3.3

If $A$ and $B$ are regular languages, show that the following languages are also regular:

**a.** $A - B$

**b.** $A \oplus B = \{w \mid w \text{ is in exactly one of } A \text{ or } B\}$.

**c.** $Echo(A) = \{a_1 a_1 a_2 a_2 \dots a_n a_n \in \Sigma^* \mid a_1 a_2 \dots a_n \in A\}$

**d.** $NOEXTEND(A) = \{w \in A \mid w \text{ is not the proper prefix of any string in } A\}$.

There are two ways we discussed in class for showing that a language is regular.

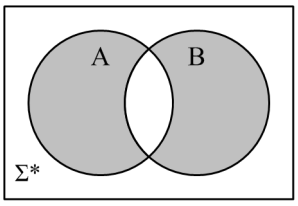
1. Show how to express the language using operations already proven to be closed under the regular languages.

   For example, consider the intersection example we covered in the last discussion section. We already know that union and complement are closed under the regular languages. Therefore, since $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$, intersection is also closed under the regular languages.

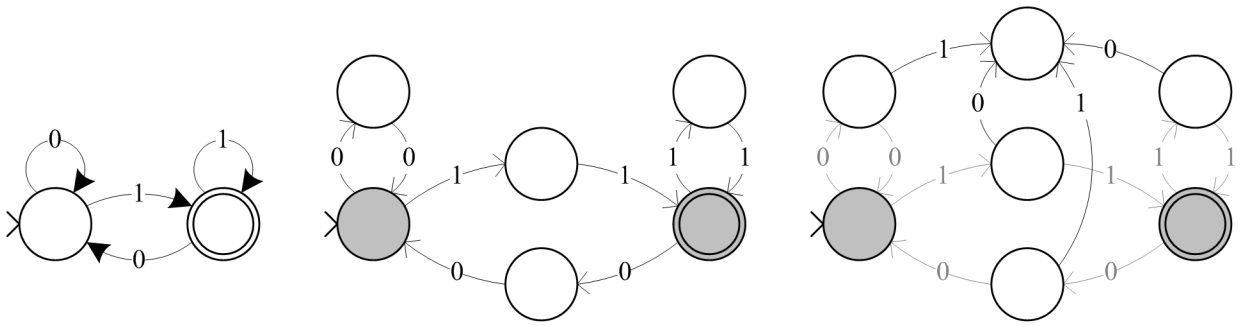
2. Show how to construct a new finite state machine to accept the new language.

   For example, this is how we proved that union, concatenation, and star are regular operations.

For part (b) notice that the Venn diagram for $A \oplus B$ looks like:




For part (c) try working out an example first:




Problem 1.40 (b) is given on page 89 of your book. It is the counterpart to the $NOPREFIX(A)$ example covered in class and in the book.

For example, let $L = \{ a, ab, abc \}$ where $\Sigma = \{ a, b, c \}$. Lets look at each string:

> Let $y = a$. This can be written as $y = \epsilon \circ a$ or as $y = a \circ \epsilon$. Therefore the prefix of $y$ can be $x = \epsilon$ or $x = a$. The proper prefix however is $x = \epsilon$ since if $x = a$ then $y = x$.
>
> Let $y = abc$. This can be written as $y = a \circ bc$, $y = ab \circ c$, $y = \epsilon \circ abc$ and $y = abc \circ \epsilon$. The prefixes of $y$ can be either $a$, $ab$, $abc$, or $\epsilon$. However the proper prefixes are $a$, $ab$, and $\epsilon$.

Let $L_1 = NOPREFIX(L)$, $L_2 = NOEXTEND(L)$, and $x$ be the proper prefix of $y$. We can summarize everything with the following table:

| $y$ | $x$ | $x \in A$ | $y \in L_1$ | $y \in L_2$ |
|---|---|---|---|---|
| $y = a$ | $\epsilon$ | $no$ | $yes$ | $no$ |
| $y = ab$ | $\epsilon, a$ | $yes$ | $no$ | $no$ |
| $y = abc$ | $\epsilon, a, ab$ | $yes$ | $no$ | $yes$ |

## Problem 3.4

Describe a decision procedure (i.e. algorithm) that, given a finite automaton $A$ and a string $u$, determines if $u$ is a prefix of some string $w \in L(A)$ (that is $w = uv$ for some string $v$).

Decision procedures are basically algorithms which are able to produce a yes or no answer with certainty. Several examples were given in lecture, including:

- Is $w$ in $L$?
- Is $L = \emptyset$?
- Is $L = \Sigma^*$?
- Is $L_1 \subseteq L_2$?
- Is $L_1 = L_2$?

We know that $L$ is regular, but we do not know how $L$ is expressed. It could be expressed as a regular expression, NFA, or DFA.