University of Virginia, Department of Computer Science

cs302 Theory of Computation Spring 2008

Problem Set 1

Due: Tuesday, 29 January (2:02pm)

This problem set covers material in Sipser Chapter 0 and Chapter 1 through Section 1.1 (page 47). Solve all ten problems. You may handwrite your answers so long as your writing is legible and easily interpreted. For full credit, answers must be concise, clear, and convincing, not just correct.

Collaboration Policy. For this assignment, we will follow the "Gilligan's Island" collaboration policy invented by Larry Ruzzo. This means:

- You may discuss and work on the problems with anyone you want.
- You may not bring any written records out of your group work session. After your group discussion, you should destroy all paper notes from the meeting.
- You should wait at least an hour after the group meeting before writing up your own solutions. Ideally, you should spend that hour doing something inane and mindless (hence the "Gilligan's Island" traditional name for this, although we recommend other options) to make sure you understand things well enough to derive them yourself, instead of just holding them in your short-term memory.
- You must clearly acknowledge others you worked with on the solutions you turn in.

The goal of this policy is to enable students to work with each other to explore and discuss solution ideas, and most importantly, to convince yourselves what the best solution is. But, to ensure that everyone understands everything in the assignments well enough to be able to derive it yourself. On the exams, you will be required to solve similar problems on your own, so the by following this policy on the problem sets you will be well-prepared to do well on the exams.

Resource Policy. Some of the problems on this assignment are well known problems for which answers are readily available on the web or from previous courses. Using found solutions violates the honor expectations for this assignment — it is detrimental to your learning the material and unfair to your fellow students. Solve the problems by thinking, not by google searching.

Problem 1: Sets and Languages. For each pair of sets, answer whether they are equal or unequal. Include a brief proof supporting your answer. Recall that for sets A and B to be equal, $A \subseteq B$ and $B \subseteq A$ must both be true.

$\mathcal{N}$ is the set of natural numbers. $\mathcal{Z}$ is the set of all integers.

a. (i) $\{i \mid i \in \mathcal{Z} \wedge i \notin \mathcal{N}\}$ (ii) $\{i \mid i = -k \text{ for some } k \in \mathcal{N}\}$

b. (i) A (ii) B where A and B are sets such that $A \cup B = A \cap B$.

c. (i) English (ii) Spanish

d. (i) the set of all even-length strings in $\{0, 1\}^*$
(ii) the set S defined recursively:

1. $\epsilon \in S$
2. If $w \in S$ then $w00$, $11w$, $0w1$, and $10w \in S$.

Problem 2: Functions. Describe 3 ways a Java method is different from a mathematical *function* (as defined in Section 0.2).

Problem 3: Graphs and Trees. These questions are intended to check your carefull understanding of the definitions in Section 0.2.

Consider the undirected graph $G = (V, E)$ where $V = \{1, 2, 3, 4\}$ and $E = V \times V$.

a. How long is the longest *simple path* in G?

b. How many subgraphs does G have?

c. How many of the subgraphs of G are *trees*?

Problem 4: Subsets. Prove that the set subset operator ($\subseteq$) is (a) reflexive and (b) transitive, but (c) not symmetric.

Problem 5: Powerset. Prove that the size of the powerset of any set S is $2^{|S|}$. (Hint: use induction on the size of the set.)

Problem 6: Pie Slicing. Consider the problem of slicing a circular pie into pieces using cuts that are straight lines all the way across the pie. What is the maximum number of pieces that can be cut using n cuts? Support your answer with a convincing argument.

Problem 7: Divisibility. Draw a DFA that recognizes the language of strings in $\{0, 1, 2\}^*$ that are not divisible by 2 when interpreted as base-3 numbers. Use as few states as possible.

Problem 8: Language Intersection. Prove that the class of regular languages is closed under intersection: $A \cap B = \{x | x \in A \wedge x \in B\}$.

Problem 9: Perfect Shuffle. (Problem 1.41 in Sipser.) For languages A and B, let the *perfect shuffle* of A and B be the language

$$\{w | w = a_1 b_1 \cdots a_k b_k, \text{ where } a_1 \cdots a_k \in A \text{ and } b_1 \cdots b_k \in B, \text{ each } a_i, b_i \in \Sigma\}$$

Show that the class of regular languages is closed under perfect shuffle.

Problem 10: Lexical Analysis. DFAs are often used in compilers to recognize lexical units in programs. Tools such as flex (*http://flex.sourceforge.net/*) generate code that performs lexical analysis (breaking a stream of text into a sequence of tokens) by translating regular expressions into finite state machines. Many programming languages also provide library functions that perform scanning such as Java's `java.util.Scanner` class and Python's `re` library. Later in the course we will explore regular expressions and how a program might construct a DFA from a regular expression. In this problem, you will manually construct DFAs that recognize a few Java programming language constructs. For each part, your answer should be a description of a DFA that exactly recognizes the described language. You can either draw the complete DFA, or explain clearly how you would construct it. For simplicity we assume an alphabet that is a subset of the actual Java alphabet:

$\Sigma = Letters \cup Digits \cup \{\text{Space}, \text{NewLine}, \texttt{/}, \texttt{*}\}$
$Letters = \{\texttt{a}, \texttt{b}, \texttt{c}, \ldots, \texttt{z}, \texttt{A}, \texttt{B}, \texttt{C}, \ldots, \texttt{Z}\}$
$Digits = \{\texttt{0}, \texttt{1}, \texttt{2}, \ldots, \texttt{9}\}$

a. Identifiers: a sequence of one or more $Letters$ and $Digits$, the first of which must be a letter.

b. Single-line comments: a `//`, followed by any number of $Letters$, $Digits$, and Space characters until a NewLine.

c. Traditional comments: `/*` *comment* `*/` where *comment* is any element of Σ^* that does not contain the substring `*/`. (Note that comments do not nest, so `/* abcd /* // defg */` is a single comment.)

d. Any string in Σ^* that does not contain a comment (of either type).