"CS 374" Fall 2015 — Homework 1

Due Tuesday, September 8, 2015 at 10am

••• Some important course policies •••

- **You may work in groups of up to three people.** However, each problem should be submitted by exactly one person, and the beginning of the homework should clearly state the names and NetIDs of each person contributing.
- **You may use any source at your disposal**—paper, electronic, or human—but you ***must*** cite ***every*** source that you use. See the academic integrity policies on the course web site for more details. For all future homeworks, groups of up to three students will be allowed to submit joint solutions.
- **Submit your pdf solutions in Moodle.** See instructions on the course website and submit a separate pdf for each problem. Ideally, your solutions should be typeset in LaTex. If you hand write your homework make sure that the pdf scan is easy to read. Illegible scans will receive no points.
- **Avoid the Three Deadly Sins!** There are a few dangerous writing (and thinking) habits that will trigger an automatic zero on any homework or exam problem. Yes, we are completely serious.
  - Give complete solutions, not just examples.
  - Declare all your variables.
  - Never use weak induction.
- Unlike previous editions of this and other theory courses we are not using the "I don't know" policy.

See the course web site for more information.

If you have any questions about these policies,
please don't hesitate to ask in class, in office hours, or on Piazza.

---

1. For each of the following languages over alphabet $\{0,1\}$, give a regular expression a brief explanation for why your expression captures the language.

   (a) $L = \{w \mid$ every maximal substring of 0s is of length divisible by 3. ($u \in \{0\}^*$ is a maximal substring of 0s of w if u occurs in w such that there is no 0 immediately to the left or right of u.)

   (b) $L = \{w \mid w$ has at most one substring of the form 00 and at most one substring of the form 11$\}$.

   (c) $L = \{w \mid w$ has equal number of occurrences of the substrings 01 and 10 $\}$.

2. We say that a regular expression is *flat* if it has no nested parentheses and no Kleene star. Let R denote the set of all strings over the alphabet $\{0,1,[,],p\}$ which are valid flat regular expressions over the alphabet $\{0,1\}$, where p denotes $+$, and [and] denote parentheses.

   Thus for example, $[01][0p1p0][00p1]$ is in R, but $p0$, $0pp1$, $[0p1$ or $[0[0p1]1]$ are not.

   Give a regular expression for R. For the sake of readability, you should define intermediate regular expressions (e.g., let $\alpha = (0+1)$) and define new regular expressions in terms of those (e.g., let $\beta = \alpha\alpha^*$). You should also briefly explain your regular expression.

3. Let $\text{pos} : \Sigma^* \times \mathbb{Z}^+ \to \Sigma \cup \{\varepsilon\}$ be such that $\text{pos}(w,i)$ is the i^{th} symbol in the string w (or ε if $i > |w|$). For two strings $u, v \in \Sigma^*$ such that $|u| = |v|$, define $u \boxplus v$ to be the unique string w such that $|w| = |u| + |v|$ and

$$\text{pos}(w,i) = \begin{cases} \text{pos}(u,(i+1)/2) & \text{if } i \text{ odd} \\ \text{pos}(v,i/2) & \text{if } i \text{ even} \end{cases}$$

   For example if $u = abc$ and $v = def$ where Σ is the English alphabet then $u \boxplus v = adbecf$. Let $L_1 \boxplus L_2 = \{u \boxplus v \mid u \in L_1, v \in L_2, |u| = |v|\}$.

   Suppose L_1 and L_2 are regular languages over Σ accepted by DFAs $M_1 = (\Sigma, Q_1, \delta_1, s_1, F_1)$, and $M_2 = (\Sigma, Q_2, \delta_2, s_2, F_2)$, respectively. Describe a DFA $M = (\Sigma, Q, \delta, s, F)$ in terms of M_1 and M_2 that accepts $L_1 \boxplus L_2$. Specify formally (using correct mathematical notation) the components Q, δ, s, and F for M in terms of the components of M_1 and M_2.

   Justify the correctness of your construction by stating a claim about $\delta^*(s,w)$ (the state M would reach on input w). Be sure to consider $|w|$ being odd as well as even. You do not need to prove the claim itself but justify why $L(M) = L_1 \boxplus L_2$ assuming the claim and your definition of F.

Not Homework Problems

The following are *not* being assigned, but are interesting problems you may wish to think about.

1. *Interesting; not to hand in.* A regular expression is said to be *top-plus* if it is of the form $(\alpha_1 + \alpha_2 + \cdots + \alpha_n)$ for some $n \geq 1$, where none of the regular expressions α_i contains an ocurrence of "+". Show by induction (on what?) that every regular language is represented by a top-plus regular expression.

   Hints:

   - You may assume that for all regular expressions r and s, $(r^*s^*)^* = (r+s)^*$
   - You may assume a distributive law of concatenation over union, so that $r(s+t) = rs + rt$ for regular expressions r, s, and t. You may use it more generally, for example, $(a+b+c)(d+e) = ad + ae + bd + be + cd + ce$ for regular expressions a, b, c, d, and e.

2. *Interesting; not to hand in.* A *linear set* of natural numbers is a set of the form $\{a + bi \mid i \in \mathbb{N}\}$. A set is *semi-linear* if it is the union of a finite number of linear sets. For all unary languages $L \subseteq 0^*$, prove that L is regular if and only if $\{i \mid 0^i \in L\}$ is semi-linear.

3. *Fun; not to hand in.* Design a collection of deterministic finite automata to solve the *firing squad problem*: Imagine a row of n DFAs. The leftmost DFA is the "general", and all of the other DFAs are "soldiers". Each DFA is adjacent to exactly two others, one on its right, and one on its left, except for the general and the rightmost soldier, who are adjacent to only one other DFA. Initially, all DFAs start in their initial state. At some arbitrary point in time, the general is placed in a special state called "fire when ready", and at some time later, all soldier DFAs must switch into a "fire" state simultaneously, and this must be the first time that any of them is in the "fire" state.

   The next state of a DFA may depend only on its own current state, and the current state(s) of its neighbors. The DFAs must all have the same program (that is, they are all identical DFAs), except perhaps for the general and the rightmost soldier. The number of states allowed SHOULD NOT DEPEND ON n, the number of DFAs in the line. More specifically, your design should consist of the description of a general DFA, a soldier DFA, and a rightmost soldier DFA, such that regardless of the number of soldiers we line up between the general and the rightmost soldier, the squad will still function properly. Note that this specifically disallows solutions where each soldier in the line figures out how far he is from the general, or the middle, or the right end, by "counting" the delay of some message passed back and forth.

   Here is an example of a design which fails miserably: If either of your neighbors are in "fire when ready" state, and your current state is not "fire when ready", then switch to "fire when ready" state. If your current state is "fire when ready", then switch to "fire" state.

   In the above example, if at time 0 the general is in state "fire when ready", then the i-th soldier (not counting the general) enters the "fire" state at time $i + 2$. Of course, the problem is to have them all enter the "fire" state for the first time simultaneously, so this is not a solution.

   For even more fun program your solution to create a graphic display of instantaneous "snapshots" showing how the squad performs.

4. *Very challenging; not to hand in.* In this problem we imagine little DFA robots running around a 2-dimensional grid world. We'll leave the formalization of such a DFA to you, but will give the basic idea: The world is made up of a finite 2-dimensional array of cells. A robot occupies one of the cells at any time. The cells are not labeled or numbered. Some of the cells are land cells, and some are water cells. The robot can not go in the water. The robot can "see" any of the eight cells

surrounding the one it is on, as well as the one it is on. It can see which of the adjacent cells are water cells, and if any of the nine cells within its sight contains anything else (such as another robot, or a pebble).

Based on the local information available form the nine cells, and the robot's current state, there is a unique next state that it switches to, as well as a next move, such as *go right, go left, go up, go down, pick-up-pebble, put-down-pebble, stay-here*.

Before describing the actual problem, we'll describe and solve a toy problem. We design a robot which, when placed anywhere on a square *island* (an arbitrary sized square grid of land cells, surrounded entirely by water cells), can find a morsel of food if there is one on the island. Note that the design is not allowed to depend on the size of the island. The idea is simple: The robot immediately goes down until the square below it is a water cell (the ocean). It then moves left until it reaches the southwest corner of the island. It then begins to exhaustively sweep the island by going up until it hits water, moving right one cell, moving down until it hits water, moving right one cell, etc. Note that a solution in which the robot tries to spiral from the outside to the inside will fail for islands larger than the number of states of the robot.

Now the real problem: (It's very hard). Design a robot which, with the aid of four pebbles, can exhaustively search (it needn't stop when it's done) an arbitrary (connected) island with arbitrary lakes. Thus the island can be any shape (not necessarily convex), as long as you can get from any point to any other, can be arbitrarily large (though finite), and can have arbitrarily shaped lakes in the middle which cannot be crossed by the robot. Initially the robot is placed somewhere on the island with four identical pebbles.

You might try approaching this problem in pieces: Assume there are no lakes, but the island is arbitrarily shaped. Assume you have a much larger collection of pebbles which are distinguishable, etc. Break the action of the robot up into subroutines.