CSE596, Fall 2014 Problem Set 2 Due Fri. Sept. 19

Lectures and Reading. After coverage of the Myhill-Nerode Theorem on Monday continuing into Wednesday, next week will transit back to the text in Chapter 2. Hence read the rest of Chapter 2. In Section 2.5 you need not care about particular details—it is enough to skim between that and my "Universal RAM Simulator" handout which has different details to appreciate the ideas.

The text and I both have an ambivalent attitude to *details* of Turing machines, but with different talking points:

- The text says it is not important to give machine details even on homework problems, so long as you make it clear that the machine can carry out the stated procedure.
- I say it's important to know the machine details for some basic operations: copying a string, checking if substrings (on different tapes) are equal, shifting over a string to free up extra space (or mul/div by 2), doing numerical addition and subtraction (from which multiplication can be synthesized by a loop and shift operations), and comparing numbers. The work toward demonstrating the capability of a Turing machine to interpret random-access assembly language—my handout does this more engineering-graphically than the text does in section 2.5.
- The text and I have the same goal: once TMs can interpret assembly, the equivalence of TMs and high-level programming languages follows, and we can switch over to reasoning in pseudocode for the rest of the course. This combats the common misperception that concepts of *computable* and *decidable* and *computably enumerable* and the *Halting Problem* are "only about Turing machines."
- However, TM details will reappear near the end of the course when we encounter the fundamental problem of telling whehter there is a path from start to finish in a big graph or maze. Moreover, the rise in importance of *streaming* in cloud computing enhances the practical import of Turing's architecture, IMTO. ("In My Teacherly Opinion.")

For the second reason, this homework does ask for machine details in diagram form. However, this comes also with emphasis on writing *comments* that explain the function and *meaning* of each state. Note that the "Turing Kit" demo I gave has a commenting feature—if you find another Turing machine app on the Net that features comment boxes atatched to states and arcs, please tell me... Yes, you're required to "comment your code" even when it's Turing code you're writing. I will give full credit if the comments on states are correct even if the states and arcs themselves are absent—which finally harmonizes this assignment with what the text says in a footnote to its Homework 2.1.

Office Hours: My regular office hours are 1–2pm on Tuesdays and Thursdays, and 3:15–5pm on Mondays. The optional weekly *Review Session* will be Fridays 1–2pm beginning next week, Sept. 19. Mike Wehar's other office hour is 1–2pm on Mondays.

Assignments will move to Wednesday due dates beginning with Assignment 3. The **First Prelim Exam** will be on **Friday, Oct. 10**, in class period.

Assignment 2 — due Friday, Sept. 19

(A) As remarked in class, the one disharmony between the definitions of Turing machines and finite automata is that:

- A Turing machine halts when it reads a character c in Γ that it can't process, and it accepts if its current state q is accepting even if it hasn't read all of its input.
- In the same situation, where a finite automaton reads a character c in Σ that it can't process, the computation is considered *not* to accept even if $q \in F$.

Sketch how to harmonize them by imposing the following changes which can apply to *any* TM or FA:

- For the FA, add a new state q_d and add an arc on c from q to q_d. Then make q_d loop back to itself, becoming a "dead state."
- For the TM, similarly connect an arc on c from q to a new state q_d, but don't make it dead. Make it loop to itself and move the head right on all characters other than the blank, and finally make it go on the blank to a single accepting state re-christened q_{accept}.

For good measure, in TM cases where q was not a/the accepting state, add a separate state $q_{d'}$ with a similar loop that goes to q_{reject} on the blank instead. Satisfy yourself also that this reconciles the text's definition with the picture in the "Turing Kit" where any TM state can be double-circled to make it accepting. (3 pts. checkoff credit)

(B) Is it possible to give every DFA M a meaningful dead state? Here "meaningful" means that there is at least one input that leads the new DFA M' into that state, while the overall language is unchanged: $L(M') = L(M)$. If you say "yes," justify with a sketch like in (A); if you say "no," give an example of a regular *language* L for which no DFA accepting L can be given such a dead state. (3 pts.)

(1) Design a deterministic one-tape Turing machine that on any input $x \in \{0,1\}^+$ changes the tape to xx. That is, given any non-empty binary string, it makes a copy of the string right afterward. As a design hint, you are free to add "alias characters" a, b to Γ, using them to keep track of how much of x has been copied. Just be sure to include a final "cleanup loop" that aliases a back to 0 and b back to 1 before halting.

Then do the same task, but with a *two*-tape Turing machine, this time leaving xx on the initially-empty second tape. Besides being faster, is the new machine easier to code?

(Per remarks above, a well-commented design is more important than machine details, but the answer must not be so "high-level" that it negates the efficiency and clarity difference of having the second tape. 18 + 12 = 30 pts.)

(2) Calculate a regular expression over $\Sigma = \{a, b\}$ for the language of strings that are *not* accepted by the following NFA: $Q = \{s, q, f\}$, $F = \{f\}$, and

$$\delta = \{(s, a, q), (s, b, f), (q, b, s), (q, a, f), (f, a, s), (f, \lambda, q)\}.$$

(Note that if the last instruction were on b not λ it would be a DFA.) You must use a strategy based on theorems in lectures and posted notes, not just inspection (that is, "hacking"). (24 pts., for 60 total including the checkoff credit)