CSE 105: Automata and Computability Theory **Fall 2015**

Homework #5

Due: Saturday, November 21th, 2015, 11:59 PM

Problem 1: Modeling Computation

A Circular-DFA (CDFA) is defined similarly to a DFA, but with the following changes:

- When the DFA tries to move past the right end of the input, it goes back to the beginning of the input. (In other words, the input string keeps repeating indefinitely.)
- Similarly to a TM, a DFA terminates by entering one of two special states, q_a (accept) and q_r (reject). When the CDFA enters q_a or q_r, the computation terminates immediately.
- The empty string is always rejected.

Formally, a CDFA is a 6-tuple $(Q, \Sigma, \delta, q_s, q_a, q_r)$ where

- Q is a finite set of states
- Σ is a finite input alphabet
- $q_s, q_a, q_r \in Q$ are the start, accept and reject states. (You may assume these states are always distinct.)
- $\delta\colon Q \times \Sigma \to Q$ is the transition function.

This defines the syntax of a CDFA. Your task is to formally specify the behavior/computations of the CDFA by defining a transition system, similarly to what we have done in class for DFAs, FSTs, and TMs. (See Lecture Notes 2, 9, 10 and 11 on the course webpage as a reference on how to define configuration transition systems for DFAs and FSTs.)

Submit a pdf file `HW51.pdf` containing a brief mathematical description of the transition system $(C_M, I_M, R_M, H_M, O_M)$, and a haskell implementaton based on the starter file `HW51.hs`. (See Lecture Notes 11 for additional information and guidance on the haskell implementation.)

Problem 2: Equivalence between models

The problem section of chapter 1 in the textbook[1] informally defines a simple FST model $T = (Q, \Sigma, \Gamma, \delta, s)$ where $\delta\colon Q \times \Sigma \to Q \times \Gamma$. We will call them SFST for "simple" FST, or "Sipser FST". For a formal "executable" definition, see the starter file `SFST.hs`. SFSTs differ from the FSTs defined in the notes in two respects:

[1] This is in problems 1.24-27 of the second and third editions. You can read and solve these problems for extra practice, but it is not needed for the solution of this homework.

- The output of an SFST is specificed "on the edges" by the transision function δ, rather than a separate output function $\gamma\colon Q \to \Gamma^*$. (This corresponds to Mealy machines in JFLAP.)
- At each step, an SFST outputs a single symbol from Γ, while FSTs can output longer strings $\gamma(q) \in \Gamma^*$.

In this problem you are asked to prove that FSTs, as defined in class (see automata library file `FST.hs`, and Lecture Notes 9,10,11) are at least as powerful as SFSTs, by showing that any SFST can be transformed into an equivalent FST that computes the same function. Submit a formal description of your transformation starting from file `HW52.hs`, together with a brief English explanation as `HW52.pdf`.

It should be the case that for any SFST t and input string w,
`evalFST (convertFST t) w == evalSFST t w`.

Problem 3: Turing Machines

Give a Turing machine for the language $L = \{a^n b^n a^n \mid n \geq 0\}$.

Submit a pdf file `HW53.pdf` with an informal description of your machine, and a jflap file `HW53.jff` with your Turing machine implementation. You can make use of any extension/feature provided by JFLAP when implementing your Turing machine, but you are required to test your implementation in JFLAP on a few input strings to make sure your TM diagram works as intended. (We are too close to the end of the quarter to accept regrade request based on JFLAP "misinterpreting" your drawings.)