

CSE105 (Winter 2014): Homework 6

Instructor: Daniele Micciancio

Due on Wednesday, March 5, 2014.

Solutions to this homework assignment should consist of a writeup with your answers to all problems, and a jflap file `HW6.jff` containing your Turing machine for problem 1. For consistency, even if submitting a single file, still zip it, and submit the resulting zipped file using `turnin`.

Problem 1

Let **BIGGER** be the set of all strings over the alphabet $\{0, 1, >\}$ of the form $x > y$, where x and y are binary numbers such that x is bigger than y . For example, $100 > 0011$ and $11 > 00$ are in **BIGGER**, but $1010 > 1111$ and $11 > 11$ are not. Give a deterministic Turing machine that decides the language **BIGGER**. Design your TM using jflap, and submit your jflap file together with a brief English description of your solution. Make sure you run and test your TM before submitting it. (In your solution, you may use any extended TM feature supported by jflap, as long as your TM is **deterministic**.)

Problem 2

The textbook defines Turing machines as finite state automata with a read/write tape that extends indefinitely to the right. If the Turing machine attempts to move beyond the left hand boundary of the tape, the tape head simply stays in its current position. JFLAP allows a seemingly more powerful type of Turing machine, whose tape is unbounded both to the left and to the right, and if the machine attempts to move beyond the left margin, a new blank cell is created to allow moving left. In this problem you are asked to give a formal definition of this extended Turing machine, including

1. the definition of a machine as a tuple of sets and functions
2. an appropriate set of configurations
3. an appropriate transition relation between configurations
4. a subset of halting configurations
5. an output function mapping each halting configuration to either 0 (for reject) or 1 (for accept).

You can use the notes on the class webpage on DFA as a guide on what your solution should look like. Study carefully how all of the above was defined for DFA, and adapt it to the TM defined in this problem.

Optional: look again at the haskell programs (`DFA.hs`) implementing DFA, configuration, etc., and provide a similar haskell implementation for the Turing machines in this problem, including a type definition, the definition of configurations, a function mapping each configuration to the next, etc. This part is optional, and will not receive a formal grade. But if you do it, you can include it in your submission as file `HW6p2.hs`.

Problem 3

Prove that the model of computation defined in problem 2 is not more powerful than the Turing machine as defined in the textbook, giving a method to transform any extended Turing machine (with doubly infinite

tape) into an equivalent regular Turing machine. Describe your transformation in detail, either by giving a formal description of the resulting Turing machine, or by providing an accurate low level description of all the steps it performs.

Optional: again, if you feel like it, you can provide a haskell implementation of regular TM (as defined in the textbook), and then also give a haskell program implementing your transformation from one type of TM to the other. If you do it, include your program in your solutions as `HW6p3.hs`.