Homework 6 Solutions

COMS W3261, Summer B 2021

This homework is due **Monday, 8/9/2021, at 11:59PM EST**. Submit to GradeScope (course code: X3JEX4).

Grading policy reminder: LaTeX is preferred, but neatly typed or handwritten solutions are acceptable. Your TAs may dock points for indecipherable writing. Proofs should be complete; that is, include enough information that a reader can clearly tell that the argument is rigorous.

The tool http://madebyevan.com/fsm/ may be useful for drawing finite state machines.

If a question is ambiguous, please state your assumptions. This way, we can give you credit for correct work. (Even better, post on Ed so that we can resolve the ambiguity.)

1 Problem 1 (12 points)

1. (4 points). Consider a variant Turing Machine with a transition function

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{3L, 2R\},$$

where the instructions $3L$ and $2R$ move the tape head three spaces to the left (halting if we reach the leftmost tape square) and two spaces to the right, respectively. Show that this variant TM is equivalent to an ordinary TM by explaining how to simulate it with an ordinary TM and vice versa. (For this question, an implementation-level description of head movement is sufficient: you do not need to formally specify the simulating TM.)

To simulate the variant TM with an ordinary TM, we simply replace each '3L' transition with a sequence of three 'L' transitions that begins by writing the appropriate tape symbol and ends by transitioning to the correct new state. We do a similar operation to replace each '2R' transition.

To simulate an ordinary TM with the variant TM, we replace each 'L' transition with a sequence containing a '3L' transition followed by a '2R' transition, writing the appropriate tape symbol before moving and transitioning to the correct new state afterwards. To replace the 'R' transition, we use the sequence '2R', '2R', '3L'.

2. (8 points). Consider a variant Turing Machine that reads and writes on an infinite 2-dimensional tape. Each tape square is indexed by a pair of non-negative integers (i.e., we can think of the tape as beginning at an origin square (0,0) and extending into the first quadrant of the coordinate grid.) The TM begins execution with the tape head at $(0, 0)$ and the generic input string $w = w_0 w_1 w_2 \dots w_n$ written on the first row of the tape (i.e., squares $(0, 0), (1, 0), \dots, (n, 0)$). This variant TM has the transition function

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, U, D\},$$

where the symbols U and D move the tape head up and down, respectively (and the tape head cannot go below 0 in either coordinate).

Show that this variant TM is equivalent to an ordinary TM by explaining how to simulate it with an ordinary TM and vice versa. (For this question, a description of memory management is sufficient: you do not need to formally specify the simulating TM or describe the precise details of head movement.)

Simulating an ordinary TM with the variant TM is trivial: we just duplicate the transition function and ignore tape squares with y-coordinate greater than 0.

Simulating the variant TM with an ordinary TM requires mapping the 2-dimensional tape onto a 1-dimensional tape. One way to do this is to consider the ordering of 2-dimensional coordinates (x, y) created by (1) sorting by the sum $x + y$ and (2) sorting coordinates with the same sum $x + y$ by x-value. This yields the ordering

$$(0,0), (0,1), (1,0), (0,2), (1,1), (2,0), (0,3), (1,2), (2,1), (3,0)...$$

and ensures that every 2-dimensional coordinate pair has some finite index in the sequence. We can then simulate the variant TM with an ordinary TM, using our mapping to simulate the two-dimensional tape.

2 Problem 2 (8 points)

1. (3 points). Prove that the language

$$ALL_{DFA} := \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \Sigma^*\}$$

is decidable by giving a high-level description of a Turing Machine that decides ALL_{DFA}.

Consider the Turing Machine M defined as follows.

M_1 = "On input $\langle A \rangle$, where A is a DFA: add the start state of A to a set of reachable states. Repeatedly add the set of all states to which we can transition from our set of reachable states to the set of reachable states until our set stops growing. At this point, accept if all reachable states are accept states and reject otherwise."

M_1 decides ALL_{DFA} because a DFA rejects a string if and only if that string terminates in a reject state.

2. (5 points). Prove that the language

$$Y_1 := \{\langle A \rangle \mid A \text{ is a DFA and } L(A) \text{ contains all strings over } \{0,1\} \text{ with at least one 1.}\}$$

is decidable. (Hint: you can refer to the fact that ALL_{DFA} is decidable.)

The language of all strings over $\{0,1\}$ with no 1's is 0^*, which is regular. Let D be a DFA that recognizes 0^*. Given an input DFA A, we observe that $\langle A \rangle \in Y_1$ if and only if

$$L(A) \cup 0^* = L(A) \cup L(D) = \Sigma^*.$$

Thus the following TM recognizes Y_1.

M_2 = "On input $\langle A \rangle$, where A is a DFA: Construct a DFA D_2 that recognizes $L(A) \cup L(D)$ using our construction for the closure of the regular languages under union. Then simulate a TM M_1 that recognizes ALL_{DFA} on $\langle D_2 \rangle$ and accept if and only if M_1 accepts."

3 Problem 3 (5 points)

1. (5 points.) Show that the language

$$ALL_{TM} := \{\langle M \rangle \mid M \text{ is a Turing Machine and } L(M) = \Sigma^*\}$$

is undecidable by reducing A_{TM} to ALL_{TM}. (That is, show that if you could decide this problem, you could decide A_{TM}. Conclude that because A_{TM} is undecidable, ALL_{TM} must be undecidable.)

Suppose for contradiction that ALL_{TM} is decidable, in which case there exists some Turing Machine T that decides ALL_{TM}. Given T, we can define a Turing Machine H that decides A_{TM} as follows.

H = "On input $\langle M_1, w \rangle$, where M_1 is a Turing Machine and w is an input string: Define a Turing Machine M_2 that accepts on every input string $x \neq w$. On input w, M_2 simulates M_1 on w and accepts if M_1 accepts. Thus M_2 accepts Σ^* if and only if M_1 accepts w. H then simulates the decider T on M_2 and accepts or rejects if T accepts or rejects."

H decides A_{TM} because H simulates the decider T, which is guaranteed to terminate and accept or reject if and only if M_1 accepts w. Note that H never actually runs M_1 on w (a computation that is not guaranteed to terminate.)

4 Problem 4 (Extra credit, 3 points)

1. (3 points). Which two previous homework problems did you find most challenging? Download the solutions from the course website, look them over, and bring any questions to TA/office hours. Good luck on the final!

   Any coherent answer.