Theory of Computation HW6 Deciders and Recognizers Ryzeson Maravich

1

M_{bad} = "On input $\langle p \rangle$, a polynomial over variables $x_1, \ldots, x_k$:

1. Try all possible settings of $x_1, \ldots, x_k$ to integer values.
2. Evaluate p on all of these settings.
3. If any of these strings evaluates to 0, *accept*; otherwise $reject$."

Claim: M_{bad} is not a description of a legitimate Turing Machine.
Proof: The description does not provide an upper bound for integer values the Turing Machine must check, so it is possible that it will never stop checking values. Therefore there will be no point where M_{bad} rejects.

2

A Turing Machine with left reset is similar to an ordinary Turing machine, but the transition function has the form $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{R, RESET\}$.
Claim: Turing machines with left reset recognize the class of Turing-recognizable languages.
Proof: Let us define the ordinary Turing machine as M, and the Turing machine with left reset at M'. To show Turing machines with left reset recognize the class of Turing-recognizable languages, we will imitate M' using M. If M makes a right transition, M' will make a right transition. If M makes a left transition, do the following on M':

1. Mark current symbol and left reset.
2. If the current symbol is marked, stop. Otherwise mark it and transition right. If this symbol is marked, go to step 3, otherwise go to step 4.
3. Unmark the symbol and left reset. Move right until you reach the marked symbol. Stop.
4. Left reset. Move right until you reach a marked symbol. Unmark it and move right. Mark this symbol and move right. If this is marked, go to step 3, otherwise repeat step 4.

Note that after this is completed, the read/write head will be pointing to a marked symbol. Treat this as an unmarked symbol in M' for the following transitions.

3

Claim: The collection of decidable languages is closed under the operation of intersection.
Proof: For any two decidable languages L_1 and L_2, let M_1 and M_2 be the Turing machines that decide them. Construct a Turing Machine $M^{'}$ that decides the intersection of L_1 and L_2.

M' ="On input w:

1. Run M_1 on w. If it accepts, continue to step 2. Otherwise $reject$.
2. Run M_2 on w. If it accepts, *accept*. Otherwise $reject$."

$M^{'}$ accepts w only if M_1 and M_2 both accept it. If either reject, $M^{'}$ rejects.

4

Claim: The collection of Turing-recognizable languages is closed under the operation of concatenation.
Proof: For any two Turing-recognizable languages L_1 and L_2, let M_1 and M_2 be the Turing machines that recognize them. Construct a non-deterministic Turing Machine $M^{'}$ that recognizes the concatenation of L_1 and L_2.

$M' =$"On input w:

1. Divide w into all possible two part splittings w_1 and w_2. Non-deterministically check each possible splitting using the following steps.
2. Run M_1 on w_1. If it accepts, continue to step 3; otherwise $reject$.
3. Run M_2 on w_2. If it accepts, *accept*; otherwise $reject$."

If there is a spitting of w that is accepted, then M' will find it (using a "breadth-first search"). If all splittings fail, then M' will determine this the same way. If all loops run forever, this is also allowed for Turing-recognizable languages.

5

Here is a high level description of a Turing machine M that decides $L = \{\langle G, a\rangle | G$ is a graph that contains a cycle from node a back to itself$\}$.

$M =$ "On input $\langle G\rangle$, the encoding of a graph G and node a:

1. Select a and mark it.
2. Repeat the following step until no new nodes are marked.
3. For each node in G, mark it (differently than in step 1) if it attached by an edge to a node that is already marked.
4. If a is marked (by step 3) *accept*; otherwise $reject$."

I affirm that I have upheld the highest principles of honesty and integrity in my academic work and have not witnessed a violation of the honor code.