CSE105 (spring 2008): Homework 4

Instructor: Daniele Micciancio

Solutions

1 Turing Machines

The Turing machine for this problem is given in Figure 1. The main idea for this Turing machine is on input $w\#v$, to read v one character at a time and compare it to the beginning of w. To keep track of the characters we have already read, replace a 0 with x and a 1 with y[1].

If while reading v we encounter a $\sqcup$, then we are done. Otherwise, we read a 0 (state q_3) or a 1 (state q_4). In either case, scan left until we reach the first X, Y, or $\sqcup$ and move back to the right. If the character matches the one we read in v, then replace it with X or Y as appropriate, move right, and continue the loop.

If the character does not match, then scan to the right replacing x and y with 0 and 1, respectively. Now scan back left replacing X and Y with 0 and 1, respectively. Once at a $\sqcup$, move right, blank out that character, move right and begin the loop again.

2 Undecidability (updated)

To show that a problem is undecidable, we must first specify it formally as a language. To that end, let

$$USELESS_{TM} = \{\langle M\rangle : M \text{ is a Turing machine with a useless state}\}.$$

If we can give a reduction $A_{TM} \leq USELESS_{TM}$, then since A_{TM} is undecidable, $USELESS_{TM}$ is undecidable. Assume that U is a Turing machine that decides $USELESS_{TM}$ and construct a Turing machine H that decides A_{TM} as follow.

H = "On input $\langle M, w\rangle$,

1. If the input is not a valid representation of a Turing machine and a string, reject.
2. Construct TM M' = 'On input x, if $x \neq w$, then reject, otherwise run $M(w)$ and accept or reject as M does.'
3. Modify M' so that whenever it is about to transition to the reject state, it transitions to every other state except for the accept state first. This can be done by adding a new tape symbol γ, writing it to two consecutive spaces and then moving back and forth between them while switching states. Call the modified machine M''.
4. Run $U(\langle M''\rangle)$ and if it accepts, then reject, otherwise accept."

To see that the reduction is correct, consider a TM M and input w. Since any string other than w causes M'' to enter every state except for the accept state, M'' will have a useless state if and only if $w \notin L(M)$. [Note: This can be modified slightly to define a map reduction from A_{TM} to the complement of $USELESS_{TM}$, that is $A_{TM} \leq_m \overline{USELESS_{TM}}$.]

[1] In actuality, I used both X for 0s in w and x for 0s in v and similarly for Y and y for 1s. This was simply to avoid needing another state between states q_3 and q_5 and also between states q_4 and q_6.

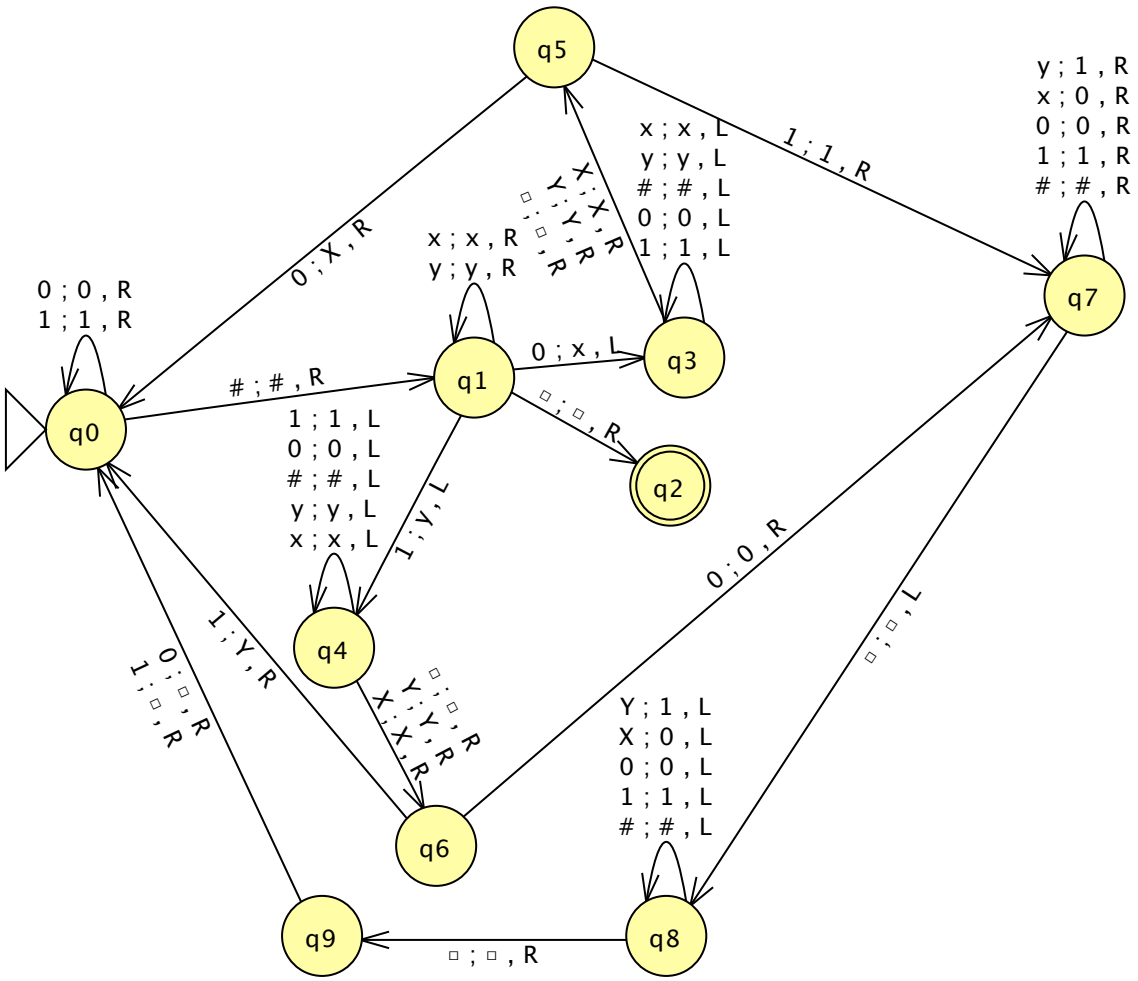

Figure 1: Turing machine for problem 1.

3 Map Reductions

If A and B are sets and $A \cup B \subseteq A \cap B$, then $A = B$. The converse is clearly true as well. Since Turing-recognizable languages (recursively enumerable languages) are closed under union—simulate both machines at once and accept if either does—and intersection—simulate one and then the other, accepting if both do—given two machines M_1 and M_2, we can produce M_1' such that $L(M_1') = L(M_1) \cup L(M_2)$ and M_2' such that $L(M_2') = L(M_1) \cap L(M_2)$. Therefore, $\langle M_1, M_2 \rangle \mapsto \langle M_1', M_2' \rangle$ is a computable function such that $L(M_1') \subseteq L(M_2')$ if and only if $L(M_1) = L(M_2)$.

The only other thing we have to be careful about is if $\langle M_1, M_2 \rangle$ is not a valid representation of a pair of Turing machines, we must output something not in $SUBSET_{TM}$. In particular $\langle M_1, M_2 \rangle \notin SUBSET_{TM}$, so outputting the input in this case works. Fully, specified, our function $F : \Sigma^* \to \Sigma^*$ is

$$F(\langle M_1, M_2 \rangle) = \begin{cases} \langle M_1, M_2 \rangle, & \text{if the representation is invalid,} \\ \langle M_1', M_2' \rangle, & \text{otherwise.} \end{cases}$$

Therefore, $EQ_{TM} \leq_m SUBSET_{TM}$.

4 Undecidability (Optional)

This problem is an easy consequence of a theorem we will not cover in this course called Rice's Theorem. It states that any nontrivial property of Turing machines is undecidable, where nontrivial means the property holds for at least one Turing machine and does not hold for at least one. In this case, the property is having a minimal number of states.