U Vic CSC320 W2020 Homework 4 Solutions

1. Define a reduction f as follows: on input $\langle M, w\rangle$, f outputs $\langle M_1\rangle$ where M_1 just simulates M on w, regardless of what is on its input tape, but modifies the simulation slightly so that it always prints a \$ before entering an accepting state. (7) This is the only time a \$ is printed (in particular, if \$ is already used by M, a new symbol should be introduced which can be used in its place.) (2)

   Claim: f is a computable function since it is adding just a few states which describe a TM which erases its tape and writes w on its tape, then a transition to M then nearly copying M, but adding some additional transitions from each accept state to write \$. (4)

   f is a reduction from A_{TM} to L: If $\langle M, w\rangle \in A_{TM}$ then M_1 writes a \$ when started with any input, including a blank tape, and $\langle M_1\rangle \in L$. If $\langle M, w\rangle \notin A_{TM}$ then $\langle M_1\rangle$ never writes a \$. Hence it's not in L. (7)

2. Given a TM M as input, the NDTM just guesses a string w and runs M on w. It accepts M if M accepts w. (10) Clearly, if $L(M) \neq \emptyset$, there is some $w \in L(M)$, so there will be a guess for which M is accepted. (5) On the other hand, if $L(M) = \emptyset$, there is no guess that will work. (5)

3. Let f take as input $\langle M\rangle$ and constructs $\langle M_1, M_2\rangle$ where M_1 which accepts every string and $\langle M_2\rangle = \langle M\rangle$. (8) It is easy to see that f is computable, since it just uses a constant TM M_1 and copies M to get M_2. (2) If $\langle M\rangle \in E_{TM}$ then $L(M) = \emptyset$ and $L(M_1) \cap L(M_2) = L(M_2) = \emptyset$. Therefore $\langle M_1, M_2\rangle \in L'$.(5) If If $\langle M\rangle \notin E_{TM}$ then $L(M) \neq \emptyset$ and $L(M_1) \cap L(M_2) = L(M_2) \neq \emptyset$ so $\langle M_1, M_2\rangle \notin L'$. (5)

4. Define a reduction f, which on input $\langle M, w\rangle$ outputs $\langle M_1\rangle$ which does the following: On input x, if $x = 10$, then accept. Otherwise, accept x iff M accepts w. (10) It is easy to see that f is computable – the code for checking whether $x = 10$ is fixed, and the rest is just a simulation M. (2) Note that if M accepts w then $L(M_1) = \Sigma^*$ (reversible) and if M does not accept w, then $L(M_1) = \{10\}$ (not reversible.) (8)

5. Given $\langle M, w\rangle$, f creates $\langle M_1, 1\rangle$ where M_1 checks if its input contains a 1. If so, it goes into a loop. Else if its input has no 1's, M_1 simulates M on w. (8) f is computable since it's easy to write a program that check the number of 1's in a string and go into a loop, and easy to simulate M on w given $\langle M, w\rangle$. (2) Proof that f is a reduction: if M halts on w then $L(M_1)$ halts exactly on strings which contain no 1's, and so $\langle M_1, 1\rangle \in L$. (5) If M doesn't halt on w, M_1 doesn't halt on any strings, so $\langle M_1\rangle \notin L$. (5)