CSCI 320 Practice Midterm II NAME: Solutions

Note that we use "TM" to mean "Turing Machine". The following languages are known to be **Undecidable**

$A_{TM} = \{< M, w > \mid M \text{ is a TM that accepts string } w\}$.
$Halt_{TM} = \{< M, w > \mid M \text{ is a TM that halts on string } w\}$.
$E_{TM} = \{< M > \mid M \text{ is a TM that accepts no strings}\}$.
$EQ_{TM} = \{< M_1, M_2 > \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$.

1. (10 marks) Show the the following language is **undecidable**:
   $OneAccepts_{TM} = \{< M_1, M_2, w > \mid M_1 \text{ and } M_2 \text{ are TMs exactly one of them accepts } w\}$. Do not use Rice's Theorem. For full marks, give a brief argument that shows your reduction works.

   Solution

   BWOC. Suppose $OneAccepts_{TM}$ is decided by a TM OA. Then the following TM A decides A_{TM}, where:

   A= On input $< M, w >$, where M is a TM and w a string:
   1. Construct a TM Rej as follows:
      Rej= "on any input, REJECT"
   2. Run OA on input $< M, Rej, w >$.
      if OA accepts, ACCEPT.
      if OA rejects, REJECT."

   If M does not accept w, then OA must reject $< M, Rej, w >$, and A returns the right anwer – it REJECTS $< M, w >$.
   If M accepts w, then OA accepts $< M, Rej, w >$, and A again returns the right answer – it ACCEPTS $< M, w >$.
   Therefore A decides A_{TM}. But A_{TM} is undecidable. That's a contradiction. Hence OA does not exist, and $OneAccepts_{TM}$ is undecidable.

2. (a) (4 marks) Prove that the class of Turing-decidable languages is closed under intersection. [Hint: "Let L_1 and L_2 be two Turing-decidable languages, decided by X_1 and X_2 respectively. Then we can construct a TM X that decides the language $L_1 \cap L_2$ as follows: ..."]

   Solution:

   Claim: The class of Turing-decidable languages is closed under intersection.
   Proof: Let L_1 and L_2 be two Turing-decidable languages, decided by X_1 and X_2 respectively. Then we can construct a TM X that decides the language $L_1 \cap L_2$ as follows:

   X= "On input $< w >$, where w a string:
   1. Run X_1 on $< w >$.
      if X_1 rejects, REJECT.
      if X_1 accepts, continue.
   2. Run X_2 on $< w >$.
      if X_2 rejects, REJECT.
      if X_2 accepts, ACCEPT."

(b) (6 marks) Prove that the class of Turing-recognizable languages is closed under union.

Solution: Proof: Let L_1 and L_2 be two Turing-recognizable languages, decided by R_1 and R_2 respectively. Then we can construct a TM R that decides the language $L_1 \cup L_2$ as follows:

X= "On input $< w >$, where w a string:
1. Run X_1 on $< w >$, and in parallel run R_2 on $< w >$.
   if either accepts, ACCEPT.
   if either rejects, continue with the other one.
   if both reject, REJECT."

Since any string in the union will be accepted eventually by either R_1 or R_2, it will give the correct answer if the answer is ACCEPT.

If w is not in the language, X will either REJECT or loop forever.

3. Are the following languages recognizable? Prove your answer.

(a) (10 marks) $\text{Reject}_{any} = \{< M > | M$ is a TM, and there is a string w that M rejects $\}$.

Solution

Reject_{TM} is recognized by a TM R, where:

R= On input $< M >$, where M is a TM:
1. Let S_i be the list of strings of length $\leq i$ over the alphabet, in shortlex order.
2. For $i = 1, 2, 3, ...$, run M on each string of S_i for i steps
   if R rejects any string, ACCEPT.
   Otherwise, $i = i + 1$ and go to 2."

If M does rejects some string w of rank i and the computation to reject takes j steps, then R will accept $< M >$ at or before the iteration $\max(i, j)$. If M does not reject any string, then R will never accept or reject, but then $< M >$ is not in the language, and a recognizer is allowed to loop forever on strings not in the language.

(b) (10 marks) OneAccepts= $\{< M_1, M_2, w > | M_1$ and M_2 are TMs, and exactly one of M_1 and M_2 accepts $w.\}$.

Solution

BWOC. Suppose $OneAccepts_{TM}$ is recognized by a TM OA_R. Then the following TM A decides A_{TM}, where:

A= On input $< M, w >$, where M is a TM and w a string:
1. Construct a TM Acc as follows:
   Acc = "on any input, ACCEPT"
2. Run OA on input $< M, Acc, w >$, and in parallel run M on $< w >$.
   if OA accepts, REJECT.
   if M accepts, ACCEPT."

If M does not accept w, then OA must accept $< M, Acc, w >$, and A returns the right anwer – it REJECTS $< M, w >$.

If M accepts w, then A again returns the right answer – it ACCEPTS $< M, w >$.