York University
CSE 2001 Fall 2017 – Assignment 4 of 4
Instructor: Jeff Edmonds

1. Let $P = \{\langle \text{“}M\text{”}, I\rangle \mid M \text{ is a TM that has a state that it never enters on input } I\ \}$.

   (a) Suppose I prove $A \leq B$.
   By Dec, I mean Computable/Decidable.
   By Rec, I mean Recognizable but not Co-Recognizable. By Co-Rec, I mean Co-Recognizable but not Recognizable.
   By Neither, I mean neither Recognizable nor Co-Recognizable.
   Circle **ALL** that are possible.

• If A is decidable then B is:	Ans: All	Dec	Rec	Co-Rec	Neither
• If A is not co-recognizable then B is:	Ans: Rec,None	Dec	Rec	Co-Rec	Neither
• If B is recognizable then A is:	Ans: Dec,Rec	Dec	Rec	Co-Rec	Neither
• If B is not decidable then A is:	Ans: All	Dec	Rec	Co-Rec	Neither
• If B is Rec and Co-Rec then B is:	Ans: Dec	Dec	Rec	Co-Rec	Neither

   (b) Is the problem P *recognizable/acceptable*? Either prove it is or argue that it is not.
   Is the problem P *co-recognizable/acceptable*? Either prove it is or argue that it is not.
   (7 sentences.)

   - Answer: It is co-recognizable but not recognizable. The problem is very similar to $\neg Halting$. You could run M on I. If it is *no* instance then it eventually enters each state at least once. The first time this happens our algorithm stops and says *no*. On the other hand, if it is a *yes* instance there is a state that is never entered. We could run M for a long time, but we would never know that we can stop say yes.

   (c) Either prove $Halting \leq_{compute} P$ or argue that it is impossible.
   Hint: In one case, try having a new character c_{loop}
   and the transition function rules $\delta(q_i, c_{loop}) = \langle q_{i+1}, c_{loop}, stay\rangle$.
   Hint: In other case, give the chain of consequences that would follow leading to a contradiction.
   (We only want *No-No, Yes-Yes* reductions.)

   - Answer: Reducing $Halting \leq_{compute} P$ is impossible.
   The short answer worth 50% is that P is like $\neg Halting$ and we know $\neg Halting \leq_{compute} Halting$ using *No-No, Yes-Yes* reductions is impossible.
   The full mark consequences are as follows.
   Such a reduction would prove that if P has an algorithm then so does $Halting$. But P does have a co-recognizing algorithm. This would mean that $Halting$ would also have a co-recognizing algorithm. But because $Halting$ has an recognizing algorithm, it would follow that $Halting$ is decidable. But we know that this is not true.

   (d) Again either prove $\neg Halting \leq_{compute} P$ or argue impossible.

   - Answer: Suppose I have an oracle that decides P.
   Here is an algorithm that will decide the $\neg Halting$ Problem.
   Given an input $\langle \text{“}M\text{”}, I\rangle$,
   I construct another TM M'.
   On input I', M' first loops through each of its states entering each except for its halting state (same as Ms halting state). This is done by writing a new character c_{loop} on the tape. The transition function would then state that as long as the head is on this character, don't do what current state wants you to do, but simply cycle to the next state. Namely $\delta(q_i, c_{loop}) = \langle q_{i+1}, c_{loop}, stay\rangle$ and $\delta(q_{last}, c_{loop}) = \langle q_{\langle start\ M\rangle}, bank, stay\rangle$.

Next M' runs like M on I'.
I give $\langle$"M'", $I\rangle$ to my oracle and if the oracle says "yes", then I say "yes" and if it says "no", then I say "no".
I prove that my algorithm works as follows.
Suppose M halts on I.
Then we know M' enters each of its states on I, because at the beginning it enters each except the halting state and when it runs like M it enters its halting state.
Then the oracle says "no".
Then I say "no".
Hence I gave the correct answer.
Suppose M runs forever on I.
Then M' also runs forever and hence never enters its halting state.
Then the oracle says "yes".
Then I say "yes".
Again I gave the correct answer.
This proves that if P is decidable, then the Halting Problem is decidable.
However, the Halting Problem is not decidable and hence P is not decidable.

(e) State Rice's Theorem. Can you directly use it to prove P or $\neg P$ is undecidable?

- Answer: Rice's theorem says that if $P = \{$"M"$|\ L(M) has some property\}$, and both P and $\neg P$ are non-empty, then P is undecideable. This is not helpful here because the definition of P does not speak at all of $L(M)$.

(f) A Yes/No computational problem P (language) can be viewed as the set of *yes* instances. Define what it means for P to be enumerable. Compare and contrast this concept with the "list" definition of P being countable?
Is the problem P *countable*? Is it *enumerable*? Give a one sentence argue.
Is the problem $\neg P$ *countable*? Is it *enumerable*?

- Answer: Both being enumerable and being countable require forming a (likely) infinite list of all the elements in P. The difference is that to be countable an all knowing Goddess could form the list and to be enumerable a lowly TM must. In particular, P is enumerable if there is a TM that prints *yes* instances so that each *yes* instance is eventually printed. (Note that if there are an infinite number of such instances then at no point in time are all printed.) We proved that a problem P is enumerable iff it is recognizable. We stated that $\neg P$ but not P is recognizable and hence $\neg P$ but not P is enumerable. Both are countable. Each TM has a finite description and hence any subset of them is countable, meaning the all knowing Goddess could list them.