York University
CSE 2001 – Unit 5.2 Reductions for Undecidability
Instructor: Jeff Edmonds

Don't cheat by looking at these answers prematurely.

1. Reductions: Let $P_{TM\ Eq} = \{\langle M, M'\rangle \mid \forall I,\ M(I) = M'(I)\}$. The "Reductions For Uncomputability" slides prove that $Halting \leq P_{TM\ Eq}$. Review this proof.

   (a) Later in the same slides stated but did not prove that $\neg Halting \leq P_{TM\ Eq}$. Do this (yes to yes, no to no) reduction. What does this say about the language $P_{TM\ Eq}$ with respect to it being undecidable, unrecognizable, and/or co-recognizable.
   Hint: We still say that $M(I) = M'(I) = \infty$ even if neither machines halt on I.

   - Answer: Suppose I have an oracle that decides $P_{TM\ Eq}$.
     Here is an algorithm that will decide the negation of Halting Problem.
     Given an input $\langle \text{“}M\text{”}, I\rangle$,
     I construct two more TMs M_I and M'.
     The first TM has M and I hard wired into it.
     M_I on input I' does the following:
       It ignores input I'.
       M_I simulates M on I and halts and accept if M halts.
     The second constructed TM M':
       M' runs forever no matter what the input I' is.
     I give $\langle \text{“}M_I\text{”}, \text{“}M'\text{”}\rangle$ 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 M_I halts and accepts every input I'.
     However M' runs forever on every input.
     Hence it is not the case that $\forall I,\ M(I) = M'(I)$.
     Hence, the oracle says "no".
     Then I say "no".
     Hence I gave the correct answer to the negation of the Halting Problem.
     Now suppose M runs forever on I.
     Then M_I runs forever on every input just like M' does.
     Hence it is the case that $\forall I,\ M(I) = M'(I)$.
     Remember the hint.
     Hence, the oracle says "Yes".
     Then I say "Yes".
     Again I gave the correct answer to the negation of the ¬Halting Problem.
     This proves that if $P_{TM\ Eq}$ is decidable, then the ¬Halting Problem is decidable.
     However, the ¬Halting Problem is not decidable and hence $P_{TM\ Eq}$ is not decidable.
     More over it proves that if $P_{TM\ Eq}$ is recognizable, then the ¬Halting Problem is recognizable.
     However, the ¬Halting Problem is not recognizable or else because it is co-recognizable, it would be decidable. Hence $P_{TM\ Eq}$ is not recognizable either.
     This reduction does not prove that $P_{TM\ Eq}$ is not co-recognizable.
     This is proved using the reduction $Halting \leq P_{TM\ Eq}$.

   (b) Remember we like Karp reductions that take Yes instances to Yes instances and No to no. Prove $Halting \leq \neg P_{TM\ Eq}$.
   Hint: Or give the changes to a previous proof.

   - Answer: The proof is identical to the proof of $\neg Halting \leq P_{TM\ Eq}$ except for both the oracle and our algorithm the Yes and No answers are switched.

2. Let $P_{reverse} = \{$"M" $\mid M$ is a TM that accepts I^R (reverse) whenever it accepts I $\}$, i.e. computational problem P says "yes" on inputs "M" in this set and says "no" on inputs not in this set.

   (a) Use a reduction to prove that this is undecidable.

   - Answer: We prove $Halting \leq P_{reverse}$ as follows.
     Suppose I have an oracle that decides $P_{reverse}$.
     Here is an algorithm that will decide the Halting Problem.
     Given an input $\langle$"M"$, I\rangle$,
     I construct another TM M_I.
     This TM has M and I hard wired into it.
     M_I on input I' does the following.
       If $I' =$ "10", then M_I halts and accepts.
       else M_I simulates M on I and halts and accept if M halts.
     I give "M_I" 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 M_I halts and accepts every input I'.
     Then M_I accepts I^R (reverse) whenever it accepts I.
     Then the oracle says "yes".
     Then I say "yes".
     Hence I gave the correct answer.
     Suppose M runs forever on I.
     Then M_I halts and accepts "10" but runs forever on "01".
     Then the oracle says "no".
     Then I say "no".
     Again I gave the correct answer.
     This proves that if $P_{reverse}$ is decidable, then the Halting Problem is decidable.
     However, the Halting Problem is not decidable, hence $P_{reverse}$ is not decidable either.

   (b) Can you directly use Rice's Theorem to prove this?

   - Answer: Yes, Rice's Theorem does apply.
     For every pair of TMs, if $L(M_1) = L(M_2)$ then $P(M_1) = P(M_2)$.
     There are TM for which $P(M)$ is yes and there are TM for which $P(M)$ is no.

   (c) Don't do an additional reduction for this question, but given the reductions we have seen, do you think $P_{reverse}$ is unrecognizable and/or co-recognizable.

   - Answer: The above reduction $Halting \leq P_{reverse}$ proves that if $P_{reverse}$ is co-recognizable, then the Halting Problem is co-recognizable.
     However, the Halting Problem is not co-recognizable or else because it is recognizable, it would be decidable. Hence $P_{reverse}$ is not recognizable either.
     The language $P_{reverse}$ is not co-recognizable for two reasons. First if we get $M'(I)$ to do what $M(I)$ does and get $M'(I^R)$ to reject, then we are asking if $M(I)$ rejects. This is co-recognizable. The second reason is that $P_{reverse}$ is not co-recognizable is that it asks what M does on every input. We gave a hard reduction to ¬Halting for this.

3. Let $P = \{\langle$"M"$, I\rangle \mid$ TM M on input I never tries to move its head left when it is already on the left hand most cell of the tape. $\}$.

   (a) Use a reduction to prove that this is undecidable.
   OOPS: The problem P is really equivalent to the negation of the halting problem. The halting problem asks for a "yes" if a given event happens while P asks for a "no". This means that P is in $co-recognizable$. It is hence impossible to prove $Halting \leq_{poly} P$. But it is possible to flip the "yes" and the "no" and prove $\neg Halting \leq_{poly} P$. This is sufficient to prove P is undecidable.

- Answer: Suppose I have an oracle that decides P.
  Here is an algorithm that will decide the Halting Problem.
  Given an input $\langle$"M"$, I\rangle$,
  I construct another TM M_I.
  This TM has M and I hard wired into it.
  M_I ignores input I'.
  It simulates M on I except if M tries to move its head left when it is already on the left hand most cell of the tape, then M_I does not do this.
  If M halts on I, then M_I moves it head to the left most cell of the tape and then tries to move the head left.
  I give "M_I" 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 the given input $\langle$"M"$, I\rangle$ is a "yes" instance.
  Then M runs forever on I.
  Then M_I never tries to move its head left when it is already on the left hand most cell of the tape.
  Then the oracle says "yes".
  Then I say "yes".
  Hence I gave the correct answer.
  Suppose the given input $\langle$"M"$, I\rangle$ is a "no" instance.
  Then M halts on I.
  Then M_I tries to move its head left when it is already on the left hand most cell of the tape.
  Then the oracle says "no".
  Then I say "no".
  Again I gave the correct answer.
  This proves that if P is decidable, then the $\neg Halting$ Problem is decidable.
  However, the $\neg Halting$ Problem is not decidable and hence P is not decidable. More over this proves that if P is recognizable, then the $\neg Halting$ Problem is recognizable.
  However, the $\neg Halting$ Problem is not recognizable or else because it is co-recognizable, it would be decidable.
  Hence P is not recognizable either.

(b) Can you directly use Rice's Theorem to prove this?

- Answer: No, Rice's Theorem does not apply.
  There can be two TM which accept the same language, i.e. $L(M_1) = L(M_2)$, however one tries to move its head too far left and one does not, i.e. $P(M_1) \neq P(M_2)$.

4. Let $P = \{\langle$"M"$, I\rangle \mid$ TM M on input I never tries to move its head left. $\}$. (Assume that TMs never leave their head stationary.)

(a) Which other model of computation does a TM that never moves its head left remind you of?

- Answer: A TM that never moves its head left is like a Deterministic Finite State Automata (DFA) in that it has a finite number of states and it reads its input once left to right.

(b) After the TM has moved its right past the input and moved right for a while on the blank tape what must eventually happen to the state that it is in (i.e. whats written on its black board)?

- Answer: After reading the input, the TM keeps coming to bank cells.
  Hence, it never gains any new information.
  Hence, the next state only depends only on its previous state.
  Because there is a finite number of states, the TM must eventually either halt or cycle, coming back to the same state a second time.
  If it ever cycles, then it will cycle forever.

(c) Give an algorithm that decides the problem.

- Answer: Simulate M on I.
  If it moves its head left, then halt and announce this.
  If it halts or cycles without moving left then halt and announce this.