Homework 07

Aaron Wang

April 11 2025

1. **Bounds Checking**

   (a) [Problem 5.30 (US ed. 5.14)] Prove that it is undecidable whether a Turing machine M, on input w, ever attempts to move its head past the left end of the tape.

   Proof idea: Reduce the Universal decider to a decider for whether a TM M ever attempts to move its head past the left end of the tape on input w.

   Proof:

   $$\text{LEFT}_{TM} = \{\langle M, w\rangle | M \text{ attempts to moves head past the left on input } w\}$$

   Suppose LEFT_{TM} is decided by R. Construct the universal decider S.

   $S =$ "on input $\langle M, w\rangle$

   1. Construct $M' =$ "on input x
      a. Prepend x with a `#`.
      b. Run M on x such that whenever the head is on `#`, it move R.[1]
      c. If M accepts, move L to the `#` and then move L one more time and *accept*.
      d. otherwise, halt."
   2. Run R on $\langle M', w\rangle$.
   3. *accept* or *reject* according to R."

   Thus, as we have constructed the universal decider S, we have a contradiction $\lightning$. Consequently our original claim that LEFT_{TM} is decidable is false. Therefore, it is undecidable whether a Turing machine M, on input w, ever attempts to move its head past the left end of the tape.

   (b) Prove that it is decidable whether a Turing machine M, on input w, ever attempts to move its head past the right end of the input string w. Your answer should be a high-level description of a TM.

   Proof idea: Use the property that there is a finite amount of different configurations if the head never moves past the end of the input. With that principle, if we ever go that amount of steps without going to the right of the input, none of the subsequent loops will either.

   Proof:

   $$\text{RIGHT}_{TM} = \{\langle M, w\rangle | M \text{ moves head past the right end of the input } w\}$$

   Create a TM L that decides RIGHT_{TM}.

   $L =$ "on input $\langle M, w\rangle$

   i. Simulate M on input w for qng^n steps[2] such that $n = |w|$, $q = |Q|$ and $g = |\Gamma|$ or until it halts such that only fake blanks are written.[3]
   ii. If M at any point during this simulation reads $\sqcup$, *reject*.
   iii. Otherwise *accept*."

[1] With this modification, M never attempts to move L past the end of the tape.
[2] Lemma 5.8 from Sipser. Maximum size of a loop in which the head never goes past the right end of the input.
[3] In class we did an example like this. For all transitions that write a $\sqcup$ write a `X` instead and for all transitions that read a $\sqcup$, make an equivalent transition that reads the `X`

2. ***The Power of 10*** is a set of rules for writing mission-critical code developed at JPL.1 Let us call a Turing machine that complies with these rules *10-compliant*. All that you need to know about 10-compliance is:

   - A 10-compliant Turing machine always halts on every input
   - It is decidable whether a Turing machine is 10-compliant.

   In this problem, we'll show that any such set of rules will be incomplete in the sense that there is a language L_2 that is decidable, yet no TM that decides L_2 complies with the rules.

   Consider the language

   $$L_2 = \{\langle M \rangle | M \text{ is a 10-compliant TM that rejects } \langle M \rangle\}.$$

   (a) Prove that L_2 is decidable.

   Create a TM TM_{L2} that decides L_2.

   TM_{L2} = "on input $\langle M \rangle$

   1. Let D be the TM that decides whether a TM is 10-compliant
   2. Run D on $\langle M \rangle$.
   3. If D rejects, *reject*, otherwise continue
   4. Simulate M on $\langle M \rangle$
   5. If M accepts, *reject*; if M rejects, *accept*. (Never loops because 10-compliant)"

   (b) Prove that any TM that decides L_2 must not be 10-compliant.

   Run TM_{L2} on $\langle TM_{L2} \rangle$. Assume towards a contradiction that TM_{L2} is 10-compliant. Thus we will be at step 4 and simulate TM_{L2} on $\langle TM_{L2} \rangle$. This should either accept or reject.

   If TM_{L2} accepts $\langle TM_{L2} \rangle$, then we must reject $\lightning$.
   If TM_{L2} rejects $\langle TM_{L2} \rangle$, then we must accept $\lightning$.

   Since we have a contradiction, TM_{L2} must not be 10-compliant.

   (c) Where in your solution to (a) is the violation of 10-compliance?

   The violation of 10-compliance occurs because it does not halt. It has an infinite recursion as TM_{L2} on input $\langle TM_{L2} \rangle$ accepts or rejects according to the inverse of TM_{L2} on input $\langle TM_{L2} \rangle$ (itself), never really giving an answer.

3. **grep and sed**

(a) Prove that it is decidable whether two regular expressions are equivalent to each other. Hint: First prove that it is decidable whether a DFA recognizes the empty language. Then use closure properties.

Proof idea: Show that it is decidable whether a DFA recognizes the empty set. Use that TM (the TM created in step 2) to decide if the symmetric difference of the two regular expressions is empty to make a TM to decide whether two regular expressions are equivalent.

Proof:

$$EQ_{REX} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are regular expressions s.t. } L(A) = L(B)\}$$

$F =$ "On input $\langle A, B\rangle$, where A and B are regular expressions.

1. Let $L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$. Construct DFA C to recognize $L(C)$.
2. Create TM $T =$ "On input $\langle D\rangle$ where D is a DFA
   i. Mark the initial state of D.
   ii. Mark any state that has an incoming transition from a marked state.
   iii. Repeat step ii. until no new states are marked.
   iv. If no accept states are marked *accept*; otherwise *reject*."
3. Simulate T on input $\langle C\rangle$.
4. *accept* or *reject* according to T."

(b) Prove that it is undecidable whether two `sed` programs (as defined in CP3) are equivalent to each other. Hint: Use CP3 Part 3.

Proof idea: Reduce the universal decider to a decider for `sed` programs that reject all strings. Reduce this new decider to a decider for whether two `sed` programs are equivalent. By reduction, it is undecidable whether two `sed` programs are equivalent.

Proof:

$$E_{\texttt{sed}} = \{\langle P\rangle \mid P \text{ is a } \texttt{sed} \text{ program as defined in CP3 s.t. } L(P) = \emptyset\}$$

Towards a contradiction, assume that R_E decides $E_{\texttt{sed}}$. Construct S, the universal decider.

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

1. Construct $M' =$ "On input x
   i. if $x \neq w$, *reject*.
   ii. if $x = w$, run M on w and *accept* or *reject* according to M."
2. Run R_E on $\langle M'\rangle$.
3. If R_E accepts, *reject*; if R_E rejects, *accept*."

Thus, as we have constructed the universal decider S, we have a contradiction $\lightning$. Consequently our original claim that $E_{\texttt{sed}}$ is decidable is false.

Let us now look at $EQ_{\texttt{sed}}$ as defined below.

$$EQ_{\texttt{sed}} = \{\langle P_1, P_2\rangle \mid P_1 \text{ and } P_2 \text{ are } \texttt{sed} \text{ programs as defined in CP3 s.t. } L(P_1) = L(P_2)\}$$

Towards a contradiction that R_{EQ} decides $EQ_{\texttt{sed}}$. Construct R_E the decider for $E_{\texttt{sed}}$

$R_E =$ "On input $\langle P\rangle$ where P is a `sed` program.

i. Run R_{EQ} on $\langle P, P_0\rangle$ where P_0 is a `sed` program s.t. $L(P) = \emptyset$.
ii. *accept* or *reject* according to R_{EQ}."

Thus, as we have constructed the R_E, we have a contradiction $\lightning$. Consequently our original claim that $EQ_{\texttt{sed}}$ is decidable is false. Therefore it is undecidable whether two `sed` programs are equivalent to each other.