COMS 3003A
HW 6

Dmitry Shkatov

Due 12 April, 2024

No new readings. This HW relies on understanding of Turing machines that you should have acquired from previous lectures and readings. Go back if you find these question hard—use the study break time for this.

We will now start describing TMs at a higher level, not wishing to be bogged down into the details of how to implement the machines we describe—this is what we expect you to do for this HW. Make sure, however, that, if pressed, you can translate your descriptions into detailed descriptions of TMs.

Mastering questions in this HW is essential to getting decent grades on subsequent tests and on the final exam.

1. **Let M be a Turing machine.**

   (a) **Design a Turing machine M' that accepts exactly those string that M rejects and rejects exactly those strings that M accepts.**

   Suppose that

   $$M \;=\; (Q, \Delta, \Gamma, q_0, q_{accept}, q_{reject}, \delta).$$

   Define

   $$M' \;=\; (Q \cup \{q'_{accept}, q'_{reject}, \}, \Delta, \Gamma, q_0, q'_{accept}, q'_{reject}, \delta')$$

   where δ' is δ plus the following instructions, for every $s \in \Gamma$:

   $$\begin{array}{rcl} q_{accept}, s & \to & q'_{reject}, s, R; \\ q_{reject}, s & \to & q'_{accept}, s, R. \end{array}$$

   It should be clear that M' works as intended.

   (b) **Under which conditions is M' going to be a decider?**

   M' is a decider if, and only if, M is a decider.

2. **Let Σ be an alphabet and let $L \subseteq \Sigma^*$. The *compliment* of L is the language**

   $$\bar{L} \;:=\; \Sigma^* \setminus L = \{w \in \Sigma^* : w \notin L\}.$$

(a) **Prove that, if L is decidable, then $\bar{L}$ is decidable.**

Suppose that L is decided by a Turing machine M. Construct a Turing machine M' as in Question 1. Then, M' decides $\bar{L}$; hence, $\bar{L}$ is decidable.

(b) **What can you say about $\bar{L}$ if L is undecidable?**

In that case, $\bar{L}$ is undecidable. Indeed, if it were decidable, then there would exist a Turing machine M that would decide it. But then the Turing machine M', defined as in Question 1, would decide L. This is, however, impossible since L is, by assumption, undecidable.

3. **Let Σ be an alphabet and let $L_1, L_2 \subseteq \Sigma^*$. Prove that, if L_1 and L_2 are decidable, then so are $L_1 \cap L_2$ and $L_1 \cup L_2$.**

We only consider $L_1 \cap L_2$ (the solution for $L_1 \cup L_2$ is similar). Suppose that L_1 and L_2 are decidable. Then, there exist Turing machines M_1 and M_2 deciding, respectively, L_1 and L_2. Let M be a Turing machine that works as follows: on input w, M simulates M_1 on w and then simulates M_2 on w; if both $M_1(w) = 1$ and $M_2(w) = 1$, then M accepts; if $M_1(w) = 0$ or $M_2(w) = 0$, then M rejects. Then, M decides $L_1 \cap L_2$:

- If $M(w) = 1$, then $M_1(w) = 1$ and $M_2(w) = 1$, which means that $w \in L_1$ and $w \in L_2$, i.e., $w \in L_1 \cap L_2$.
- If $M(w) = 0$, then $M_1(w) = 0$ or $M_2(w) = 0$, which means that $w \notin L_1$ or $w \notin L_2$, i.e., $w \notin L_1 \cap L_2$.

4. **We want to give as input to a Turing machine with the binary input alphabet a binary encoding $\langle M \rangle$ of a Turing machine M and a binary word w. We, therefore, want to represent both as a single string $\langle M, w \rangle$ so that the machine that is given input $\langle M, w \rangle$ knows where $\langle M \rangle$ ends and w begins. How could we do this?**

There are multiple ways of doing this. Here is one: prefix the input with the number of 0s that equals the length of $\langle M \rangle$ and then with a single 1 (after that, the encoding of M starts, immediately followed by w).

5. **Assume that M_A is a Turing machine with the input alphabet $\{0, 1\}$ that solves the following decision problem: given a binary encoding $\langle M \rangle$ of a Turing machine M and a binary word w, it decides if M accepts w, i.e.,**

$$M_A(\langle M \rangle, w) \quad = \quad \begin{cases} 1 & \textbf{if } M(w) = 1\textbf{;} \\ 0 & \textbf{otherwise (what does this mean?).} \end{cases}$$

Using M_A as a helper function, design a Turing machine M_{SA} that solves the following problem: given a binary encoding $\langle M \rangle$ of a Turing machine M, decide if M accepts its own encoding $\langle M \rangle$.

The machine M_{SA} works as follows on input $\langle M \rangle$:

(1) M_{SA} calls M_A with arguments $\langle M \rangle$ and $\langle M \rangle$.

(2) If $M_A(\langle M \rangle, \langle M \rangle) = 1$, then M_{SA} accepts; if $M_A(\langle M \rangle, \langle M \rangle) = 0$, then M_{SA} rejects.

It is not hard to see that M_{SA} works as required:

- If $M_{SA}(\langle M\rangle) = 1$, then $M_A(\langle M\rangle, \langle M\rangle) = 1$, which means, by specification of M_A, that $M(\langle M\rangle) = 1$.
- If, on the other hand, $M_{SA}(\langle M\rangle) = 0$, then $M_A(\langle M\rangle, \langle M\rangle) = 0$, which means, by specification of M_A, that $M(\langle M\rangle) = 0$.

6. **Assume that M_ϵ is a Turing machine with the input alphabet $\{0,1\}$ that solves the following decision problem: given a binary encoding $\langle M\rangle$ of a Turing machine M, it decides if M halts on the empty string ϵ, i.e.,**

$$M_\epsilon(\langle M\rangle) \;=\; \begin{cases} 1 & M(\epsilon) \neq \infty; \\ 0 & \textbf{otherwise.} \end{cases}$$

Using M_ϵ as a helper function, design a Turing machine M_H that solves the following problem: given a binary encoding $\langle M\rangle$ of a Turing machine M and a word w in its input alphabet, decide if M halts on w.

The machine M_H works as follows on input $\langle M, w\rangle$:

(1) M_H writes down (the program of) a Turing machine $M_{M,w}$ that accepts as input (an encoding of) a Turing machine and a binary string and works as follows on inputs $\langle M\rangle$ and x:
   (a) $M_{M,w}$ simulates M on xw.
   (b) If $M(xw) = 1$, then $M_{M,w}$ accepts; if $M(xw) = 0$, then $M_{M,w}$ rejects.

(2) M_H calls M_ϵ with the argument $\langle M_{M,w}\rangle$.

(3) If $M_\epsilon(\langle M_{M,w}\rangle) = 1$, then M_H accepts; if $M_\epsilon(\langle M_{M,w}\rangle) = 0$, then M_H rejects.

It is not hard to see that M_H works as required:

- If $M_H(\langle M\rangle, w) = 1$, then $M_\epsilon(\langle M_{M,w}\rangle) = 1$, which means, by specification of M_ϵ, that $M_{M,w}(\epsilon) \neq \infty$; looking at the definition of $M_{M,w}$, we see that this happens only if $M(w) \neq \infty$ (notice that $M_{M,w}$ would go into an infinite loop only if M went into an infinite loop on input $\epsilon w (= w)$).
- If $M_H(\langle M\rangle, w) = 0$, then $M_\epsilon(\langle M_{M,w}\rangle) = 0$, which means, by specification of M_ϵ, that $M_{M,w}(\epsilon) = \infty$; looking at the definition of $M_{M,w}$, we see that this happens only if $M(\epsilon w) = M(w) = \infty$.

7. **Assume that M_H is a Turing machine with the input alphabet $\{0,1\}$ that solves the following decision problem: given a binary encoding $\langle M\rangle$ of a Turing machine M and a word w in its input alphabet (which we may assume to be binary), it decides if M halts on w, i.e.,**

$$M_H(\langle M, w\rangle) \;=\; \begin{cases} 1 & \textbf{if } M(w) \neq \infty; \\ 0 & \textbf{otherwise.} \end{cases}$$

Using M_H as a helper function, design a Turing machine M_A that solves the following problem: given a binary encoding $\langle M\rangle$ of a Turing machine M and a word w in its input alphabet, decide if M accepts w.

The machine M_A works as follows, given an input $\langle M, w\rangle$:

(1) M_A writes down (the program of) a Turing machine M_M that accepts as input a binary string and works as follows on input x:
   (a) M_M simulates M on x.
   (b) If $M(x) = 1$, then M_M accepts; if $M(x) = 0$, then M_M goes into an infinite loop.
(2) M_A calls M_H with arguments $\langle M_M \rangle$ and w.
(3) If $M_H(\langle M_M, w \rangle) = 1$, then M_A accepts; $M_H(\langle M_M, w \rangle) = 0$, then M_A rejects.

It is not hard to see that M_A works as required:

- If $M_A(\langle M \rangle, w) = 1$, then $M_H(\langle M_M \rangle, w) = 1$, which means, by specification of M_H, that $M_M(w) \neq \infty$; looking at the definition of M_M, we see that this happens only if $M(w) = 1$.
- If $M_A(\langle M \rangle, w) = 0$, then $M_H(\langle M_M \rangle, w) = 0$, which means, by specification of M_H, that $M_M(w) = \infty$; looking at the definition of M_M, we see that this can happen for two reasons: either $M(w) = \infty$ or $M(w) = 0$; in either case, $M(w) \neq 1$.

8. **Assume that $M_\sqcup$ is a Turing machine with the input alphabet $\{0, 1\}$ that solves the following decision problem: given a binary encoding $\langle M \rangle$ of a Turing machine M, it decides if M writes a blank during the course of at least one of its computations, i.e.,**

$$M_\sqcup(\langle M \rangle) = \begin{cases} 1 & \textbf{if } M \textbf{ writes } \sqcup \textbf{ during at least one of its computations;} \\ 0 & \textbf{otherwise.} \end{cases}$$

Using $M_\sqcup$ as a helper function, design a Turing machine M_A that solves the following problem: given a binary encoding $\langle M \rangle$ of a Turing machine M and a word w in its input alphabet, decide if M accepts w.

The machine M_A works as follows, given an input $\langle M, w \rangle$:

(1) M_A writes down (the program of) a Turing machine $M_{M,w}$ that accepts no input and works as follows:
   (a) $M_{M,w}$ simulates M on w in such a way that it never writes a $\sqcup$ (notice that M's blank is encoded as a string of non-blanks from M_M's alphabet; also notice that M_M can always avoid writing a $\sqcup$: if it needs to delete the contents of a cell, it can simply write a special tape symbol, say #, in that cell);
   (b) If $M(w) = 1$, then $M_{M,w}$ writes a $\sqcup$ and accepts; if $M(w) = 0$, then $M_{M,w}$ immediately rejects.
(2) M_A calls $M_\sqcup$ with the argument $\langle M_{M,w} \rangle$.
(3) If $M_\sqcup(\langle M_{M,w} \rangle) = 1$, then M_A accepts; $M_\sqcup(\langle M_{M,w} \rangle) = 0$, then M_A rejects.

It is not hard to see that M_A works as required:

- If $M_A(\langle M \rangle, w) = 1$, then $M_\sqcup(\langle M_{M,w} \rangle) = 1$, which means, by specification of $M_\sqcup$, that $M_{M,w}$ writes a blank (since $M_{M,w}$ accepts no inputs, it always performs the same computation when it runs). This only happens if $M(w) = 1$.

- If $M_A(\langle M\rangle, w) = 0$, then $M_\sqcup(\langle M_{M,w}\rangle) = 0$, which means, by specification of $M_\sqcup$, that $M_{M,w}$ never writes a blank. This happen if $M(w) = \infty$ or if $M(w) = 0$, i.e., if $M(w) \neq 1$.

9. **Assume that M_2 is a Turing machine with the input alphabet $\{0, 1\}$ that solves the following decision problem: given a binary encoding $\langle M\rangle$ of a Turing machine M, it decides if M ever is in the same state in two consecutive configurations, i.e.,**

$$M_\sqcup(\langle M\rangle) \;=\; \begin{cases} 1 & M \textbf{ ever is in the same state in two successive configurations;} \\ 0 & \textbf{otherwise.} \end{cases}$$

Using M_2 as a helper function, design a Turing machine M_H that solves the following problem: given a binary encoding $\langle M\rangle$ of a Turing machine M and a word w in its input alphabet, decide if M halts on w.

The machine M_H works as follows, given an input $\langle M, w\rangle$:

(1) M_H writes down (the program of) a Turing machine $M_{M,w}$ that accepts no input and works as follows:
   (a) $M_{M,w}$ simulates M on w in such a way that it never stays in the same state in two successive configurations (notice that this can always be achieved: $M_{M,w}$ can always go to a dummy state and then on to the next state while keeping its head in the same cell just for the sake of fulfilling this requirement);
   (b) Upon the completion of the simulation (if it completes), $M_{M,w}$ moves write and writes $\sqcup$ while remaining in the same state; then, it accepts.

(2) M_H calls M_2 with the argument $\langle M_{M,w}\rangle$.

(3) If $M_2(\langle M_{M,w}\rangle) = 1$, then M_H accepts; $M_2(\langle M_{M,w}\rangle) = 0$, then M_H rejects.

It is not hard to see that M_H works as required:

- If $M_H(\langle M\rangle, w) = 1$, then $M_2(\langle M_{M,w}\rangle) = 1$, which means, by specification of M_2, that $M_{M,w}$ stays in the same state in two successive configurations. This can only happen if $M(w) \neq \infty$.
- If $M_H(\langle M\rangle, w) = 0$, then $M_2(\langle M_{M,w}\rangle) = 0$, which means, by specification of M_2, that $M_{M,w}$ never stays in the same state in two successive configurations. This can only happen if $M(w) = \infty$.