Bram Schuijff
CSCI 387, §S01/02
Homework 4
March 6th, 2025 by 10:30am

1. No, NOHALT is undecidable. By Theorem 4.22 in the textbook, a language is decidable if and only if it is Turing recognizable and co-Turing recognizable. We know that HALT is undecidable but it is recognizable. To prove this, take $\langle M, w \rangle$ to be an encoding of a Turing machine M and a string w in the alphabet of M. We can construct a Turing machine U that recognizes HALT as follows:

   1. On input $\langle M, w \rangle$:
   2. Run M on w.
   3. If M eventually halts on w, accept.

   U recognizes HALT because all Turing machines that halt on given inputs will eventually halt and therefore U will accept. Thus, HALT is recognizable. NOHALT is clearly the complement of HALT because for any Turing machine and string $\langle M, w \rangle$, either M halts on w—either by accepting or rejecting—or it does not; it cannot somehow do both or neither. Thus, NOHALT is the complement of HALT. Assume for the sake of contradiction that NOHALT is decidable. Then, $HALT$ would be recognizable and co-recognizable—because decidable languages are recognizable—and would therefore be decidable. This creates a contradiction by Theorem 5.1 which says that HALT is undecidable. Therefore, we conclude that NOHALT is undecidable.

2. We know that the reverse of a regular language is also regular by closure. We also have a construction such that, when given a DFA M, we can construct another DFA M^R such that $L(M^R) = L(M)^R$ where $w^R \in L(M)^R$ if and only if $w \in L(M)$. This construction is rather simple: any DFA can be converted without loss of generality into a generalized nondeterministic finite automata as on page 70 in the textbook. This will have only one accept state. We then reverse all the arrows on the transition function, making the start state the single accept state and the old accept state the new start state. This produces a GNFA that accepts the reverse strings of our original DFA. We can then convert from a GNFA back into a DFA to produce M^R. By Theorem 4.5, EQ_{DFA} is decidable. Thus, we can produce a Turing machine R that decides L as follows:

   1. On input DFA M:
   2. Use our construction to produce M^R.
   3. Run EQ_{DFA} on $\langle M, M^R \rangle$.
   4. If EQ_{DFA} accepts, accept. Otherwise, reject.

   Because there exists a Turing machine that decides L, L is decidable by definition.

3. No, L is undecidable. Suppose for the sake of contradiction that L is decidable. Then, there exists a Turing machine R such that R decides L. We will show that this yields a Turing machine S that decides EQ_{TM}, a contradiction by Theorem 5.2. We do so as follows:

   1. For Turing machine S on input Turing machines $\langle A, B \rangle$:
   2. Construct a Turing machine C such that $L(C) = L(A) \cup L(B)^R$
   3. For Turing machine R on input Turing machine $\langle C \rangle$:

4. If R accepts $\langle C \rangle$, accept. Otherwise, reject.

To demonstrate that we can construct C, we know that reversing a string can be done by a Turing machine. Therefore, if we have such a reverser, we can make it so that C nondeterministically decides that a given input string w is in $L(B)^R$, reverse it and run B on w^R as a subroutine. If it accepts, C accepts; otherwise, C rejects or loops infinitely. Whether or not $L(C)$ is Turing-decidable is irrelevant as we are just running R on $\langle C \rangle$ which by assumption is decidable. Because S decides EQ_{TM}, we produce a contradiction and therefore declare that L is undecidable.

4. No, THREE is undecidable. Suppose for the sake of contradiction that THREE is decidable. Then, there exists a Turing machine R that decides THREE. We will show that this yields a Turing machine S that decides A_{TM} which is a contradiction. We do so as follows:

   1. For Turing machine S on inputs $\langle M, w \rangle$:
   2. Modify M so that M rejects all strings except for w. On w, M runs normally.
   3. If $w \in \{\epsilon, 0\}$, go to step 4. Otherwise, modify M to accept $\{\epsilon, 0\}$ and go to step 6.
   4. If $w = \epsilon$, go to step 5. Otherwise, modify M to accept $\{\epsilon, 1\}$ and go to step 6.
   5. Modify M to accept $\{0, 1\}$ and go to step 6.
   6. Run R on $\langle M \rangle$. If it accepts, accept. Otherwise, reject.

   This decides A_{TM} because if R accepts, we know that the language of the modified version of M contains the two strings it was specifically designed to accept and w. This allows us to know that the unmodified M accepts w because the modified M behaved normally when given w. Therefore, S decides A_{TM} but we have proven that A_{TM} is undecidable. Hence, we produce a contradiction and know that THREE is undecidable.

5. We can construct the language

$$L = \{\langle M, w \rangle \mid M \text{ a TM such that in the computation history of } M \text{ on } w \text{ there exists a step where } M \text{ performed more left moves than right moves}\}$$

   Because the head of a Turing starts at the leftmost cell on the tape, if at any point the net moves to the left is greater than 0, we have that M bumped into the left edge of the tape at least once. To demonstrate that L is undecidable, we suppose for the sake of contradiction that there exists a Turing machine R which decides L. We will demonstrate that this allows us to construct a Turing machine S that decides A_{TM}, a contradiction. First, I define the concept of a *left-aware Turing machine* $\hat{M}$ from a Turing machine M.

   $\hat{M}$ is the same as M in every way except for three key factors:

   - $\hat{M}$ has one additional symbol $\gamma \notin \Gamma$ in its tape alphabet.
   - On an input w, before performing any computation, $\hat{M}$ will first place the symbol γ at the leftmost space on the tape and move every symbol in w one space to the right. After that point, $\hat{M}$ scrolls left until it reads γ. It then moves one space right and runs M on w as normal. If M would ever bump into the left edge of the tape, $\hat{M}$ would read γ. When $\hat{M}$ reads γ, it immediately moves right. This places $\hat{M}$ back at the start of M's tape. In this way, $\hat{M}$ avoids ever bumping into its tape's left edge while running M on w.

- If M would accept w, $\hat{M}$ first scrolls left until it reads γ. It then moves one space further left—thereby bumping into its tape's left edge—and accepts. If M would reject, $\hat{M}$ rejects as normal.

Now, we can construct S:

1. For Turing machine S on inputs $\langle M, w \rangle$:
2. Construct a left-aware Turing machine $\hat{M}$ from M.
3. If R accepts $\langle \hat{M}, w \rangle$, accept. Otherwise, reject.