Lesson 10: Universal Turing machine and Halting problem

Theme: Universal Turing machine and Halting problem

The string representation of a Turing machine. Recall that a Turing machine is defined as a system \( M = \langle \Sigma, \Gamma, Q, q_0, q_{\text{acc}}, q_{\text{rej}}, \delta \rangle \), where we can assume that \( \Sigma = \{0, 1\} \) and \( \Gamma = \{\langle, 0, 1, \sqcup\} \). Without loss of generality, we can also assume that \( Q = \{0, 1, \ldots, n\} \) for some positive integer \( n \) with 0 being the initial state.

We note the following.

- Each state \( i \in Q \) is written as a string in its binary form.
- Each transition \((i, a) \rightarrow (j, b, \alpha) \in \delta\) can be written as string over the symbols \( 0, 1, (, ) , \langle, \sqcup , L, R, S \), where the symbol \( \sqcup \) represents \( \sqcup \), and \( L, R, S \) represent Left, Right, Stay, respectively.

So, the whole system \( M = \langle \Sigma, \Gamma, Q, q_0, q_{\text{acc}}, q_{\text{rej}}, \delta \rangle \) can be written as a string:

\[
[\Sigma] \# [\Gamma] \# [Q] \# [0] \# [q_{\text{acc}}] \# [q_{\text{rej}}] \# [\delta]
\]

where \([\cdot]\) denotes the string representing the component \( \cdot \) and \( \# \) the symbol separating two consecutive components.

This shows that every Turing machine (whose tape alphabet is \( \Gamma = \{\langle, 0, 1, \sqcup\}\}) can be described as a string over a fixed set of the symbols, i.e., \( 0, 1, (, ) , \langle, \sqcup , L, R, S, \# \). All these symbols can be further encoded into strings over 0 and 1 to obtain a binary string, which we denote by \( [M] \). That is, \( [M] \) is the binary string representing the Turing machine \( M \). Sometimes, we will also say \( [M] \) is the string description of \( M \), or the description of \( M \), for short.

Universal Turing machine (UTM). A universal Turing machine (UTM) is a Turing machine \( U \) that gets as input a description of a Turing machine \( [M] \) and a word \( w \). On such input, it simulates \( M \) on \( w \). (Some textbooks use the phrase “it runs \( M \) on \( w \)” for “it simulates \( M \) on \( w \).”)

Halting problem. We define the following languages:

\[
\text{HALT} := \{ [M] \$w \mid M \text{ accepts } w \text{ where } w \in \{0, 1\}^* \}.
\]

\[
\text{HALT}_0 := \{ [M] \mid M \text{ accepts } [M] \}.
\]

\[
\text{HALT}'_0 := \{ [M] \mid M \text{ does not accepts } [M] \}.
\]

Theorem 10.1 \( \text{HALT}'_0 \) is undecidable.

Corollary 10.2 \( \text{HALT}_0 \) and \( \text{HALT} \) are undecidable.

Proposition 10.3 The language \( \text{HALT}_0 \) and \( \text{HALT} \) are recognizable (recursively enumerable).

Recall that if both \( L \) and its complement \( \overline{L} = \Sigma^* - L \) are recognizable, then both are decidable. Then, the following corollary follows immediately from above.

Corollary 10.4 The language \( \overline{\text{HALT}} \) is not recognizable (recursively enumerable).

\*Obviously, since we consider only Turing machines with \( \Sigma = \{0, 1\} \) and \( \Gamma = \{\langle, 0, 1, \sqcup\} \), it is not necessary to include them in \( [M] \). But for the sake of consistency in our notation, we simply include them.