Lesson 7: Turing machines and the notion of algorithm

Theme: Turing machines and the notion of algorithms.

1 Multi-tape Turing machines

A multi-tape Turing machine is a Turing machine that has a few tapes. On each tape, the Turing machine has one head. Formally, it is defined as follows. Let $k \geq 1$. A $k$-tape Turing machine is $M = \langle \Sigma, \Gamma, Q, q_0, q_{acc}, q_{rej}, \delta \rangle$, where $\delta$ is a function

$$
\delta : (Q - \{q_{acc}, q_{rej}\}) \times \Gamma^k \to Q \times (\Gamma - \{\bot\})^k \times \{\text{Left, Right}\}^k
$$

As before, an element of $\delta$ is written in the form:

$$(q, a_1, \ldots, a_k) \rightarrow (p, b_1, \ldots, b_k, \alpha_1, \ldots, \alpha_k).$$

Intuitively, it means that if the TM is in state $q$, and on each $i = 1, \ldots, k$, the head on tape $i$ is reading $a_i$, then it enters state $p$, and for $i = 1, \ldots, k$, the head on tape $i$ writes the symbol $b_i$ and moves according to $\alpha_i$.

A configuration of $M$ is of the form $(q, u_1, \ldots, u_k)$, where $q \in Q$ and each $u_i$ is a string over $\Gamma \cup \{\bullet\}$ and the symbol $\bullet$ appears exactly once in each $u_i$. The symbol $\bullet$ is to denote the position of the head.

The input is always written in the first tape. All the other tapes are initially blank. Formally, the initial configuration on input $w$ is $(q_0, \bullet w, \bullet, \ldots, \bullet)$.

The notion of “one step computation” $C \vdash C'$ is defined similarly as in the standard Turing machine. Likewise, the conditions of acceptance and rejection are defined as when the Turing machines enter the accepting and rejecting states, respectively.

**Theorem 7.1** For $k$-tape TM $M$, there is a single tape TM $M'$ such that for every word $w$, the following holds.

- If $M$ accepts $w$, then $M'$ accepts $w$.
- If $M$ rejects $w$, then $M'$ rejects $w$.
- If $M$ does not halt on $w$, then $M'$ does not halt on $w$.

2 Some theorems on decidable and recognizable languages

**Theorem 7.2**

- If a language $L$ is decidable, so is its complement $\Sigma^* - L$.
- If both a language $L$ and its complement $\Sigma^* - L$ are recognizable, then $L$ is decidable.

**Theorem 7.3**

- Recognizable languages are closed under union, intersection, concatenation and Kleene star.
- Decidable languages are closed under union, intersection, complement, concatenation and Kleene star.
Appendix

A An informal definition of algorithm

We define an algorithm (informally) as consisting of one “main” Boolean function of the form:

```plaintext
Boolean main (String w)
{
    statement;
    :
    statement;
}
```

and some (finite number of) functions of the form:

```plaintext
⟨value-type⟩ function ⟨function-name⟩ (⟨variable-name⟩,...,⟨variable-name⟩)
{
    statement;
    :
    statement;
}
```

Statements in the algorithm are of the following form:

- ⟨variable-name⟩ := ⟨expression⟩;
- ⟨variable-name⟩ := ⟨function-name⟩(⟨variable-name⟩,...,⟨variable-name⟩);
- return ⟨variable-name⟩/⟨some-value⟩;
- if ⟨condition⟩
  {
    statement;
    :
    statement;
  } else
  {
    statement;
    :
    statement;
  }
- while ⟨condition⟩ do
  {
    statement;
    :
    statement;
  }
```
Note that we define our algorithm to mimic closely the C++ language. For simplification, we assume that \( \text{condition} \) in the \textbf{if} and \textbf{while} is always of the form: \( x = 1 \), where \( x \) is a variable name. Moreover, we also assume that the variables used in each function have different names.

Informally, one can argue that every computer program can be written as an algorithm defined above. Note that when we write an algorithm (or any computer program, in fact), it uses only a fixed number of variables (including variables for data structures such as linked list, arrays, etc). Multi-tape Turing machines and our definition of algorithms above are equivalent in the sense that a \( k \)-tape Turing machine can be viewed as an algorithm that uses \( k \) variables, and conversely, an algorithm that uses \( k \) variables can be viewed as a \( k \)-tape Turing machine.