Chapter 4 – Turing Machines

Goal: To study computability of functions and languages.

Mechanism: Turing Machines

Definition: A Deterministic Turing Machine (DTM)

\[ M = (S, \sum, \delta, s, H) \]
where
- \( S \) is a finite set of states.
- \( \sum \) is a tape alphabet which includes D, U in addition to symbols used to describe the problem
- \( s \) is the start state
- \( H \subseteq S \) is the set of halting states.
- \( \delta : (S - H) \times \sum \rightarrow S \times (\sum \cup \{-, -, \}) \)
(When halting state is reached, you’re done!)
(Reading a symbol or writing a symbol)

Configuration:

A configuration is used to describe the tape content, the head location and the state of a particular moment. Hence, sometimes the configuration is also called Instantaneous description (ID). This means that we can use the following notation to capture the configuration:
(q, uav) - uav is the tape content
- a is the head location
- q is the state

We’ll specify the initial configuration as follows:
Initial:

Next Configuration:
**Case 1**

Suppose $\delta(p, a) = (q, b)$, then the next configuration is $(q, u \, b \, v)$, written as $(p, u \, a \, v) \vdash (q, u \, b \, v)$

**Case 2**

Suppose $c$ (not writing, only moving head)
Then the next configuration is $(q, u' c \, a \, v)$, where $u = u' c$, $c$ is a tape symbol.

Suppose $\delta(p, a) = (q, \rightarrow)$, then the next configuration is $(q, u \, a \, c \, v')$,
Where $cv' = \delta$

A computation path is a sequence of configurations
$\sigma_1, \sigma_2, \ldots, \sigma_k$ such that $\sigma_1 \vdash \sigma_2 \vdash \sigma_3 \vdash \ldots \vdash \sigma_{k-1} \vdash \sigma_k$
We often write it as $\sigma_1 \vdash^* \sigma_k$ (similar to debugging)

Let $x$ be a string over $\sum_0 = \sum - \{ \!, \, U \}$

If $(s, \triangleright \, a \, x') \vdash^* (h, \triangleright \, u \, c \, v)$ for some $u, c, v$
where $x = ax'$, $a$ is a symbol. Then we can say that $x$ is accepted by $M$, and $ucv$ is the output of the machine $M$ on input $x$. 
Let \( L(M) = \{ x \mid (s, \triangleright a x') \vdash^* (h u \triangleright v), x = ax' \} \)
We say that \( L(M) \) is the language accepted by \( M \). Let \( f \) be a function and \( \text{dom}(f) \subseteq \sum^* \).
If \( M \) is a DTM such that on any input \( x \subseteq \text{dom}(f) \), there is \( u, c, v \) such that \( (s, \triangleright a x') \vdash^* (h, \triangleright u c v) \) and \( f(x) = ucv \), then we say that \( M \) completes \( f \) (or \( f \) is computable).

**Example 4.1.3 (Text - Page 129)**

Construct a turing machine (TM) \( M \) that computes function \( f(w) = w \# w \), \( w \in \sum_0^* \# \notin \sum \)

![Diagram](image1)

Use a state to memorize the current symbol.

Technique: Sweeping

(\textit{Go back and forth until we find what we want to read})

\( \Sigma_0 = \sum_0 U \sum_{o_2} \)

\( \Sigma = \sum_0 U \{ \triangleright, W \} \) for tape symbols.

**Stage 1:**

Assume the leftmost symbol of \( w \) is \( a \). Replace \( 'a' \) with \( 'A \notin \Sigma' \).

Then go to a new state \( q_a \). Keep moving to the right until a blank is read. When a blank is read, replace it with \# . Move to the right. Replace that blank with \( 'a' \).

**Stage 2:**

Change state to \( q_< \). Keep moving to the left while the symbol on tape is not \( A \) or \( B \). (it stops when \( A \) or \( B \) is read).

Change state to \( q_> \). Move to the right. Assume the tape symbol is \( b \) or \( a \).
Change state to $q'_b$. Then replace $b$ with $B$, and keep moving to the right until $U$ is read.
Replace that blank with $b$.
Go back to stage 2.
Repeat until no more original symbols on the left side of $#$ are on the tape.

**Stage 3:**

One sweep to the left and replace every $B$ with $b$, $A$ with $a$. When $\triangleright$ is read, halt!