Chapter 3, Part 1

Computability Theory
Turing Machines

A Turing machine is a step-wise computing device consisting of:

1. an infinitely long tape divided into tape squares (or tape cells),
2. a head for scanning the tape, and
3. a finite control that maintains the current state.
A **Turing machine** is a step-wise computing device consisting of:

1. an infinitely long **tape** divided into **tape squares** (or **tape cells**),
2. a **head** for scanning the tape, and
3. a **finite control** that maintains the current state.

![Diagram of a Turing machine](image-url)
Computation by a Turing Machine

In each computational step, a Turing machine:

• reads the symbol written at the current head location, and then,

• depending on the symbol and the current state
  • determines its next state,

• write a symbol at the current head location, and

• moves the head to either the next or the prior tape square.
Turing Machines (cont’d)

A Turing machine is a 7-tuple \((Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})\), where \(Q, \Sigma, \Gamma\) are finite sets,

1. \(Q\) is a set of states,
2. \(\Sigma\) is the input alphabet,
3. \(\Gamma\) is the tape alphabet such that \(\Sigma \subset \Gamma\). There is a special symbol \(\sqcup\) in \(\Gamma - \Sigma\).
4. \(\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}\) is the transition function,
5. \(q_0 \in Q\) is called the initial state,
6. \(q_{accept} \in Q\) is called the accept state, and
7. \(q_{reject} \in Q\) is called the reject state.
Initial Condition of a Turing Machine

At the beginning

1. the tape contains its input in the leftmost squares,
2. the rest of the tape is filled with $\square$,
3. the head is at the leftmost cell, and
4. the state is $q_0$. 

![Diagram of a Turing machine with an initial state and input on the tape.](image-url)
Termination of a Turing Machine

The Turing machine halts when it enters either $q_{\text{accept}}$ or $q_{\text{reject}}$. We say that the machine accepts when the former occurs and rejects when the latter occurs.
Computation of a Turing Machine in One Step (Again)

In each computational step, a Turing machine:

- reads the symbol written at the current head location, and then,
- depending on the symbol and the current state
  - determines its next state,
  - write a symbol at the current head location, and
  - moves the head to either the next or the prior tape square.
Computation of a Turing Machine in One Step (Again)

In each computational step, a Turing machine:

• reads the symbol written at the current head location, and then,

• depending on the symbol and the current state
  • determines its next state,
  • write a symbol at the current head location, and
  • moves the head to either the next or the prior tape square.

However, if the instruction is to move the head to the left and the current head position is the leftmost square, then the head stays at the leftmost square.
Example

A string of symbols, as shown below, could represent a sequence of events or tokens in a computational system. Each symbol might correspond to a different state or action. For example, if the symbols represent different instructions in a program or different states in a machine, then the sequence could be interpreted as a history or trace of the system's behavior.

In the example, the string starts with `a`, followed by `b`, then another `a`, followed by `b`, and finally `c`. This sequence could represent a series of transitions or interactions. The string ends with a delimiter, indicating the end of the sequence or a change in context.

The diagram shows two possible states or configurations of the system. The first state, `q1`, is reached after the sequence `a b a b c`. The second state, `q2`, is reached after the sequence `a b b b c`. These states could represent different phases or stages in the system's operation.
The Languages of Turing Machines

A Turing machine $M$ **accepts** (rejects) a string $x$ if it enters the accepting (rejecting) state on input $x$.

A Turing machine $M$ **recognizes** a language $L$, if for every input $x$, $M$ on $x$ accepts if $x \in L$ and does not accept if $x \notin L$.

A language is **Turing-recognizable** (or **recursively enumerable**) if there is a Turing machine that recognizes it.
The Languages of Turing Machines (cont’d)

A Turing machine $M$ decides a language $L$, if $M$ recognizes $L$ and halts on all inputs.

In other words, a Turing machine $M$ decides a language $L$ if, for all $x$, $M$ on $x$ accepts if $x \in L$ and rejects if $x \notin L$.

A language is Turing-decidable (or simply decidable) if there is a Turing machine that decides it.
Example 1: \( L = \{ w \# w \mid w \in \{0, 1\}^* \} \)

\( L \) is not context free.

To decide whether an input \( z \in \{0, 1, \#\}^* \) is in \( L \) our Turing machine \( M \) operates as follows:

- \( M \) attempts to match letter by letter the word to the left of \( \# \) (call the word \( u \)) and the word to the right of \( \# \) (call the word \( v \)).
- When a letter in \( u \) matches a letter in \( v \) at the corresponding position, both letters are erased using a special letter \( x \).
- When all the letters of \( u \) and \( v \) have been matched, \( M \) accepts.
Components of $M$

$M$ in itself has the following components:

- an automaton for $x^* \# x^*$;
- an automaton for $x^*0\{0, 1\}^* \# x^*0\{0, 1\}^* \#$;
- an automaton for $x^*1\{0, 1\}^* \# x^*1\{0, 1\}^* \#$;

Intuitively, $M$ runs these automata concurrently.

The last two automata replace the letter immediately after each run of $x$ with an $x$ if the two letters match.
The tape contents are always of the form $x^*\{0, 1\}^* \# x^*\{0, 1\}^*$, where the number of $x$’s on the left is equal to the number of $x$’s on the right.
Action in the Very First Step

- If the symbol being scanned is \( \square \), then immediately reject \( z \) because \( z = \varepsilon \).

- Otherwise, enter the loop.
Main Loop

- **(Case 1)** If the letter at the current head position is a $\#$, then:
  - scan to the right until a letter other than $x$ is found, and then
  - accept if that non-$x$ is $\square$; reject otherwise.
Method for Matching

• (Case 2) If the letter at the current head location is a 0 or a 1, then:
  • memorize the letter, say $c$, and write an $x$,
  • scan to the right until a $\#$ is found,
  • scan to the right until a non-$x$ is found, and then
  • if that non-$x$ is not a $c$, then reject,
  • replace the non-$x$ with an $x$,
  • scan to the left until a $\#$ is found,
  • scan to the left until an $x$ is found,
  • move to the right thereby locating the head to the very first non-$x$. 
Example $\{w\#w \mid w \in \{0, 1\}^*\}$
The Main Loop

First non-\textit{X}
The Main Loop

\[
\begin{array}{cccccc}
X & X & 1 & 0 & \# & X & 0 & 1 & 0 \\
\end{array}
\]

0 memorized
The Main Loop

X X 1 0 # X 0 1 0

0 memorized
The Main Loop

0 memorized

```
X X 1 0 # X 0 1 0
```
The Main Loop

0 memorized and # found
The Main Loop

0 memorized and # found
The Main Loop

\[ \begin{array}{cccccc}
X & X & 1 & 0 & \# & X & 0 & 1 & 0 \\
\end{array} \]

0 memorized
# found
and
0 found
The Main Loop

```
XX10 # XX10
```

matched
The Main Loop

matched
# found
The Main Loop

matched
# found
The Main Loop

X X 1 0 # X X 1 0

matched
# found
The Main Loop

```
X X 1 0 # X X 1 0
```

matched
# found
X found
The Main Loop

Start of new round

matched
# found
X found

CSC527, Chapter 3, Part 1 © 2012 Mitsunori Ogihara
Example 2: \( \{ w\#w\#w \mid w \in \{0, 1\}^* \} \)

- Instead of \( x \), use special symbols \( x_0 \) and \( x_1 \) used for crossing out a 0 and a 1, respectively.

- Execute the algorithm for the previous example with these substitutions and treating the second \( \# \) as \( \square \).

- The algorithm halts in the accept state if and only if the input is in the form \( w\#w\#y \) for some \( w, y \in \{0, 1\}^* \).

- If accept, move on to the next stage.
Example 2: \( \{w#w#w \mid w \in \{0, 1\}^*\} \)

- Move the head to the symbol immediately to the right of the first 
  \#.
- Execute the algorithm for the previous example by treating \( x_0 \)
  as 0 and \( x_1 \) as 1.
- Accept if and only if this simulation accepts.
A **configuration** of a Turing machine is the setting of its tape, head, and state.

If the tape contents are $a_1, \ldots, a_m, \sqcup, \ldots$, the head is located on the $k$th square, and the state is $q$, then write

$$a_1 \ldots a_{k-1} qa_k \ldots a_m$$

to represent the configuration as a word.

Note that $Q$ is treated as an alphabet.

The infinite stretch of $\sqcup$’s to the right of the head position is omitted from configuration. However, at least one symbol must appear after the state symbol.

Thus a configuration is a word in $\Gamma^*Q\Gamma(\Gamma - \{\sqcup\})^*$. 
Examples

The configuration is $abq_1abc$.

The configuration is $aq_2bbbc$. 
The action of a TM can be viewed as rewriting of the configuration.

A configuration \( C_1 \) \textbf{yields} \( C_2 \) if the Turing machine can go from \( C_1 \) to \( C_2 \) in a single step.

- \( uaq_i bv \) yields \( uq_j acv \) if \( \delta(q_i, b) = (q_j, c, L) \).
- \( q_i bv \) yields \( q_j cv \) if \( \delta(q_i, b) = (q_j, c, L) \) (the set cannot move to the left).
- \( uq_i bv \) yields \( ucq_j v' \) if \( \delta(q_i, b) = (q_j, c, R) \).

Here \( v' = v \) if \( v \neq \epsilon \) and \( v = \square \) if \( v = \epsilon \).
Special Configurations

An accepting configuration (A rejecting configuration) is one in which the state is $q_{\text{accept}}$ ($q_{\text{reject}}$).

Both accepting configuration and rejecting configuration are halting configurations.