1 Turing Machines

1.1 Introduction

Turing machines provide an answer to the question, What is a computer? It turns out that anything that is equivalent in power to a Turing machine is a general purpose computer.

Turing machines are a general model of computation.

- They are more powerful than push-down automata.
- For example, there is a Turing machine that recognizes the language $\{a^n b^n c^n : n \geq 0\}$.

Turing machines have

- a *finite control*,
- a *one-way infinite tape*, and
- a *read-write head* that can move in two directions on the tape.

This slight increase in power over push-down automata has dramatic consequences.

No more powerful model of computer is known that is also feasible to construct.

- This makes Turing machines very interesting because one can use them to prove problems *unsolvable*.
- Basically, if a problem can't be solved on a Turing machine, it can't be solved on any reasonable computer.

There are various models of Turing machines that differ in various details.

- The model in the text can *either* write a symbol *or* move the read-write head at each step.
- The tape is also one-way infinite to the right.

Other Turing machine models that are common have a two-way infinite tape and permit the machine to write and move on the same step.

In our model,

- the left end of the tape is marked with a special symbol $\triangleright$ that cannot be erased.
- The purpose of this symbol is to prevent the read-write head from falling off the end of the tape.

Conventions used in this course:

- The symbol $\leftarrow$ means move left; the symbol $\rightarrow$ means move to the right.
- The input to the Turing machine is written to the right of the $\triangleright$ marker on the tape, at the left end of the tape.
- Beyond this, at the start, there are infinitely many blanks on the tape. Blanks are indicated by $\sqcup$. There may be a blank between the left-end marker and the input.
- It is not specified where the read-write head starts in general, but frequently it is specified to be next to the left-end marker at the start.

So the tape looks something like this:

$\triangleright$	$\sqcup$	a_1	a_2	a_3	a_4	a_5	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	...

1.2 Formal Definition

Formally, a Turing machine is a quintuple

$$(K, \Sigma, \delta, s, H)$$

where

K	is a finite set of states
Σ	is an alphabet containing $\sqcup$ and $\triangleright$ but not $\leftarrow$ or $\rightarrow$
$s \in K$	is an initial state
$H \subseteq K$	is a set of halting states
δ	is a transition function from

$$\begin{array}{ccccccccc} (K-H) & \times & \Sigma & \text{to} & K & \times & (\Sigma & \cup & \{\leftarrow, \rightarrow\}) \\ & & & & & & & & \\ \text{non-halting} & & \text{scanned} & & \text{new} & & \text{symbol} & & \text{direction} \\ \text{state} & & \text{symbol} & & \text{state} & & \text{written} & & \text{moved} \end{array}$$

such that for all $q \in K - H$, if $\delta(q, \triangleright) = (p, b)$ then $b = \rightarrow$
(must move right when a $\triangleright$ is scanned)
for all $q \in K - H$ and $a \in \Sigma$, if $\delta(q, a) = (p, b)$ then $b \neq \triangleright$
(can't write a $\triangleright$)

1.3 Example Turing machines

$M = (K, \Sigma, \delta, s, \{h\})$, $K = \{q_0, q_1, h\}$, $\Sigma = \{a, \sqcup, \triangleright\}$, $s = q_0$.

δ:

q	σ	$\delta(q,\sigma)$		
q_0	a	$(q_1, \sqcup)$	see a,	write $\sqcup$
q_0	$\sqcup$	$(h, \sqcup)$	see $\sqcup$,	halt
q_0	$\triangleright$	$(q_0, \rightarrow)$	see $\triangleright$,	move right
q_1	a	(q_0, a)	see a,	switch to q_0
q_1	$\sqcup$	$(q_0, \rightarrow)$	read $\sqcup$,	move right
q_1	$\triangleright$	$(q_1, \rightarrow)$	read $\triangleright$,	move right

Here's an example computation:

$$\begin{array}{cccccccc} & q_o & & & & & & \\ \triangleright & a & a & a & a & \sqcup & \sqcup & \cdots \end{array}$$

$$\begin{array}{cccccccc} & q_1 & & & & & & \\ \triangleright & \sqcup & a & a & a & \sqcup & \sqcup & \cdots \end{array}$$

$$\begin{array}{cccccccc} & & q_0 & & & & & \\ \triangleright & \sqcup & a & a & a & \sqcup & \sqcup & \cdots \end{array}$$

$$
\begin{array}{ccccccccc}
 & & q_1 & & & & & \\
\rhd & \sqcup & \sqcup & a & a & \sqcup & \sqcup & \cdots
\end{array}
$$

$$
\begin{array}{ccccccccc}
 & & & q_0 & & & & \\
\rhd & \sqcup & \sqcup & a & a & \sqcup & \sqcup & \cdots
\end{array}
$$

This computation can also be written this way:

$$
\begin{array}{c}
(q_0, \rhd\underline{a}aaa \sqcup \sqcup), \\
(q_1, \rhd\underline{\sqcup}aaa \sqcup \sqcup), \\
(q_0, \rhd \sqcup \underline{a}aa \sqcup \sqcup), \\
(q_1, \rhd \sqcup \underline{\sqcup}aa \sqcup \sqcup), \\
(q_0, \rhd \sqcup \sqcup\underline{a}a \sqcup \sqcup)
\end{array}
$$

It is also possible to write it without even mentioning the state, like this:

$$
\begin{array}{c}
\rhd\underline{a}aaa \sqcup \sqcup, \\
\rhd\underline{\sqcup}aaa \sqcup \sqcup, \\
\rhd \sqcup \underline{a}aa \sqcup \sqcup, \\
\rhd \sqcup \underline{\sqcup}aa \sqcup \sqcup, \\
\rhd \sqcup \sqcup\underline{a}a \sqcup \sqcup
\end{array}
$$

1.4 Configurations and Computations

A *configuration* of a Turing machine $M = (K, \Sigma, \delta, s, H)$ is a member of

K	$\times$	$\rhd\Sigma^*$	$\times$	$(\Sigma^*(\Sigma - \{\sqcup\}) \cup \{\epsilon\})$
state		tape contents to left of read head, and scanned square		rest of tape, not ending with blank; all blanks indicated by ϵ

Configurations can be written as indicated above, with underlining to indicate the location of the read-write head.

- If C_1 and C_2 are configurations, then $C_1 \vdash_M C_2$ means that C_2 can be obtained from C_1 by one move of the Turing machine M.

- $\vdash_M^*$ is the transitive closure of $\vdash_M$, indicating zero or more moves of the Turing machine M.

- A *computation* by M is a sequence $C_0, C_1, C_2, \ldots, C_n$ of configurations such that $C_0 \vdash_M C_1 \vdash_M C_2 \ldots$. It is said to be of *length* n. One writes $C_0 \vdash_M^n C_n$.

- A *halting configuration* or *halted configuration* is a configuration whose state is in H.

1.5 Complex example Turing machines

It is convenient to introduce a programming language to describe complex Turing machines. For details about this, see Handout 8. Handout 7 gives details of a Turing machine to copy a string from one place on the tape to another.

We can also give the idea of a Turing machine to recognize $\{a^n b^n c^n : n \geq 0\}$ by showing a computation as follows:

$$
\begin{gathered}
\rhd \underline{\sqcup} aaabbbccc \vdash \\
\rhd \sqcup \underline{a} aabbbccc \vdash \\
\rhd \sqcup \underline{d} aabbbccc \vdash \\
\rhd \sqcup d\underline{a}abbbccc \vdash \\
\rhd \sqcup da\underline{a}bbbccc \vdash \\
\rhd \sqcup daa\underline{b}bbccc \vdash \\
\rhd \sqcup daa\underline{d}bbccc \vdash \\
\rhd \sqcup daad\underline{b}bccc \vdash \\
\rhd \sqcup daadb\underline{b}ccc \vdash \\
\rhd \sqcup daadbb\underline{c}cc \vdash \\
\rhd \sqcup daadbb\underline{d}cc \vdash \\
\ldots \\
\rhd \sqcup ddaddbd\underline{d}c \vdash \\
\ldots \\
\rhd \sqcup dddddddd\underline{d}
\end{gathered}
$$

Finally the Turing machine checks that all a, b, and c run out at the same time.