Lesson 6. Turing machines and the notion of algorithms

CSIE 3110 – Formal Languages and Automata Theory

Tony Tan
Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Table of contents

Table of contents

Multi-tape Turing machines

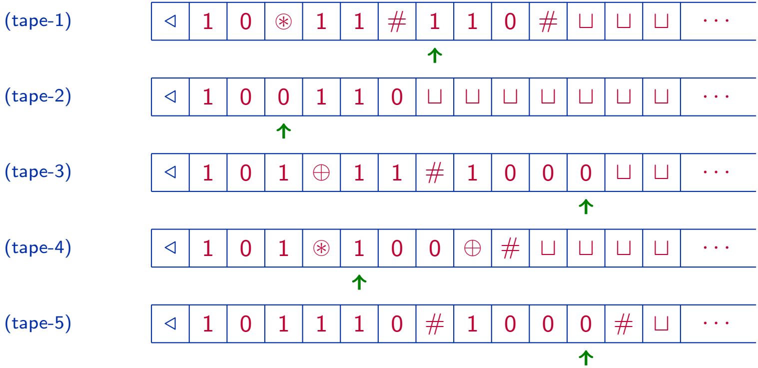
Recall that a TM has one tape (with infinitely many cells).

We can view the tape as a "scrap" paper for the TM to do its computation.

In this lesson we will extend TM with multiple tapes

Example: 5-tape TM

On input w:



To help with computation, the TM has five tapes and one head on each tape.

Example: 5-tape TM

On input w:

(tape-1)	$\triangleleft$	1	0	$\circledast$	1	1	$\#$	1	1	0	$\#$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$
								↑							
(tape-2)	$\triangleleft$	1	0	0	1	1	0	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$
				↑											
(tape-3)	$\triangleleft$	1	0	1	$\oplus$	1	1	$\#$	1	0	0	0	$\sqcup$	$\sqcup$	$\cdots$
												↑			
(tape-4)	$\triangleleft$	1	0	1	$\circledast$	1	0	0	$\oplus$	$\#$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$
						↑									
(tape-5)	$\triangleleft$	1	0	1	1	1	0	$\#$	1	0	0	0	$\#$	$\sqcup$	$\cdots$
												↑			

To help with computation, the TM has five tapes and one head on each tape.

(Note) The number of tapes is fixed, i.e., 5. On whatever input word w, the TM has 5 tapes to do the computation.

Multi-tape Turing machines

We can talk about 10-tape Turing machine, 10^{10}-tape Turing machine or 10^{20}-tape Turing machine.

Multi-tape Turing machines

We can talk about 10-tape Turing machine, 10^{10}-tape Turing machine or 10^{20}-tape Turing machine.

In general we can talk about a *k-tape TM* for any integer $k \geqslant 0$, where k is a fixed number like 10, 10^{10} or 10^{20}.

Multi-tape Turing machines

We can talk about 10-tape Turing machine, 10^{10}-tape Turing machine or 10^{20}-tape Turing machine.

In general we can talk about a k-tape TM for any integer $k \geqslant 0$, where k is a fixed number like 10, 10^{10} or 10^{20}.

Theorem 6.1 (intuitive version)
Every k-tape TM $\mathcal{M}$, where $k \geqslant 2$, is "equivalent" to a 1-tape TM $\mathcal{M}'$, i.e., $\mathcal{M}$ and $\mathcal{M}'$ compute the same thing.

Multi-tape Turing machines

We can talk about 10-tape Turing machine, 10^{10}-tape Turing machine or 10^{20}-tape Turing machine.

In general we can talk about a k-tape TM for any integer $k \geqslant 0$, where k is a fixed number like 10, 10^{10} or 10^{20}.

Theorem 6.1 (intuitive version)
Every k-tape TM $\mathcal{M}$, where $k \geqslant 2$, is "equivalent" to a 1-tape TM $\mathcal{M}'$, i.e., $\mathcal{M}$ and $\mathcal{M}'$ compute the same thing.

Intuitively Theorem 6.1 is correct since a tape has infinitely many cells.

Multi-tape Turing machines

We can talk about 10-tape Turing machine, 10^{10}-tape Turing machine or 10^{20}-tape Turing machine.

In general we can talk about a k-tape TM for any integer $k \geqslant 0$, where k is a fixed number like 10, 10^{10} or 10^{20}.

Theorem 6.1 (intuitive version)
Every k-tape TM $\mathcal{M}$, where $k \geqslant 2$, is "equivalent" to a 1-tape TM $\mathcal{M}'$, i.e., $\mathcal{M}$ and $\mathcal{M}'$ compute the same thing.

Intuitively Theorem 6.1 is correct since a tape has infinitely many cells.

So the amount of information that can be stored in, say 10^{10} tapes, can also be stored in a single tape.

The formal definition of k-tape Turing machines

(Def.) A *k-tape Turing machine* is a system $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}}, \delta \rangle$:

- Σ, Γ, Q, q_0, q_{acc} and q_{rej} are the same as in the 1-tape TM.
- δ is the transition function:

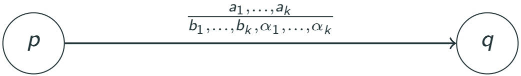
$$\delta \; : \; (Q - \{q_{\mathrm{acc}}, q_{\mathrm{rej}}\}) \times \Gamma^k \; \to \; Q \times \Gamma^k \times \{\texttt{Left}, \texttt{Right}\}^k$$

whose elements are written in the form:

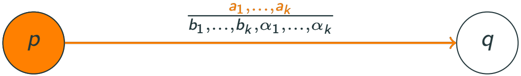
$$(p, a_1, \ldots, a_k) \; \to \; (q, b_1, \ldots, b_k, \alpha_1, \ldots, \alpha_k)$$

where $p, q \in Q$, $a_1, \ldots, a_k, b_1, \ldots, b_k \in \Gamma$ and $\alpha_1, \ldots, \alpha_k \in \{\texttt{Left}, \texttt{Right}\}$.

The intuitive meaning of $(p, a_1, \ldots, a_k) \to (q, b_1, \ldots, b_k, \alpha_1, \ldots, \alpha_k)$



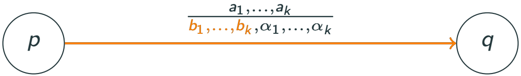
The intuitive meaning of $(p, a_1, \dots, a_k) \to (q, b_1, \dots, b_k, \alpha_1, \dots, \alpha_k)$



If:

- the TM is in state p,
- for each $i = 1, \dots, k$, the head on tape i is reading symbol a_i,

The intuitive meaning of $(p, a_1, \ldots, a_k) \to (q, b_1, \ldots, b_k, \alpha_1, \ldots, \alpha_k)$



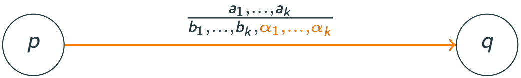
If:

- the TM is in state p,
- for each $i = 1, \ldots, k$, the head on tape i is reading symbol a_i,

then:

- for each $i = 1, \ldots, k$, the head on tape i writes symbol b_i,

The intuitive meaning of $(p, a_1, \ldots, a_k) \to (q, b_1, \ldots, b_k, \alpha_1, \ldots, \alpha_k)$



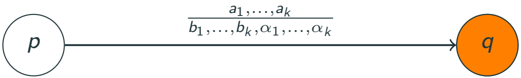
If:

- the TM is in state p,
- for each $i = 1, \ldots, k$, the head on tape i is reading symbol a_i,

then:

- for each $i = 1, \ldots, k$, the head on tape i writes symbol b_i,
- for each $i = 1, \ldots, k$, the head moves α_i where $\alpha_i \in \{\texttt{Left}, \texttt{Right}\}$,

The intuitive meaning of $(p, a_1, \ldots, a_k) \to (q, b_1, \ldots, b_k, \alpha_1, \ldots, \alpha_k)$



If:

- the TM is in state p,
- for each $i = 1, \ldots, k$, the head on tape i is reading symbol a_i,

then:

- for each $i = 1, \ldots, k$, the head on tape i writes symbol b_i,
- for each $i = 1, \ldots, k$, the head moves α_i where $\alpha_i \in \{\texttt{Left}, \texttt{Right}\}$,
- the TM enters state q.

Configuration of a k-tape Turing machine

Let $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}}, \delta \rangle$ be a k-tape TM.

(Def.) A *configuration* of $\mathcal{M}$ is a string of the form:

$$(q, \triangleleft u_1, \ldots, \triangleleft u_k)$$

where $q \in Q$, each u_i is a string over $\Gamma \cup \{\bullet\}$ and the symbol $\bullet$ appears exactly once in each u_i.

Configuration of a k-tape Turing machine

Let $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}}, \delta \rangle$ be a k-tape TM.

(Def.) A *configuration* of $\mathcal{M}$ is a string of the form:

$$(q, \triangleleft u_1, \ldots, \triangleleft u_k)$$

where $q \in Q$, each u_i is a string over $\Gamma \cup \{\bullet\}$ and the symbol $\bullet$ appears exactly once in each u_i.

The symbol $\bullet$ denotes the position of the head. As before, the symbol $\triangleleft$ is the left-end marker of each tape.

Configuration of a k-tape Turing machine

Let $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}}, \delta \rangle$ be a k-tape TM.

(Def.) A *configuration* of $\mathcal{M}$ is a string of the form:

$$(q, \triangleleft u_1, \ldots, \triangleleft u_k)$$

where $q \in Q$, each u_i is a string over $\Gamma \cup \{\bullet\}$ and the symbol $\bullet$ appears exactly once in each u_i.

The symbol $\bullet$ denotes the position of the head. As before, the symbol $\triangleleft$ is the left-end marker of each tape.

(Recall) In 1-tape TM a configuration is a string of the form:

$$\triangleleft a_1 \cdots a_{i-1} \ p \ a_i \cdots a_m$$

where we use the state p to indicate the position of the head.

Acceptance and rejection by a k-tape TM

(Def.) The *initial configuration* of $\mathcal{M}$ on input w is

$$(q_0, \triangleleft \bullet w, \triangleleft\bullet, \ldots, \triangleleft\bullet)$$

That is, the first tape initially contains the input word and all the other tapes are initially blank.

Acceptance and rejection by a k-tape TM

(Def.) The *initial configuration* of $\mathcal{M}$ on input w is

$$(q_0, \lhd \bullet w, \lhd\bullet, \ldots, \lhd\bullet)$$

That is, the first tape initially contains the input word and all the other tapes are initially blank.

The notion of "one step computation" $C \vdash C'$ is defined as in 1-tape TM.

Acceptance and rejection by a k-tape TM

(Def.) The *initial configuration* of $\mathcal{M}$ on input w is

$$(q_0, \triangleleft \bullet w, \triangleleft\bullet, \ldots, \triangleleft\bullet)$$

That is, the first tape initially contains the input word and all the other tapes are initially blank.

The notion of "one step computation" $C \vdash C'$ is defined as in 1-tape TM.

(Def.) The run of $\mathcal{M}$ on input word w:

$$C_0 \ \vdash \ C_1 \ \vdash \ \cdots$$

where C_0 is the initial configuration of $\mathcal{M}$ on w.

Acceptance and rejection by a k-tape TM

(Def.) The *initial configuration* of $\mathcal{M}$ on input w is

$$(q_0, \triangleleft \bullet w, \triangleleft\bullet, \ldots, \triangleleft\bullet)$$

That is, the first tape initially contains the input word and all the other tapes are initially blank.

The notion of "one step computation" $C \vdash C'$ is defined as in 1-tape TM.

(Def.) The run of $\mathcal{M}$ on input word w:

$$C_0 \ \vdash \ C_1 \ \vdash \ \cdots$$

where C_0 is the initial configuration of $\mathcal{M}$ on w.

$\mathcal{M}$ accepts w, if the run is accepting. $\mathcal{M}$ rejects w, if the run is rejecting.

The equivalence between k-tape TM and 1-tape TM

Theorem 6.1

For every k-tape TM $\mathcal{M}$, where $k \geqslant 2$, there is a 1-tape TM $\mathcal{M}'$ such that for every input word w, the following holds.

- *If $\mathcal{M}$ accepts w, then $\mathcal{M}'$ accepts w.*
- *If $\mathcal{M}$ rejects w, then $\mathcal{M}'$ rejects w.*
- *If $\mathcal{M}$ does not halt on w, then $\mathcal{M}'$ does not halt on w.*

The equivalence between k-tape TM and 1-tape TM

Theorem 6.1

For every k-tape TM $\mathcal{M}$, where $k \geqslant 2$, there is a 1-tape TM $\mathcal{M}'$ such that for every input word w, the following holds.

- *If $\mathcal{M}$ accepts w, then $\mathcal{M}'$ accepts w.*
- *If $\mathcal{M}$ rejects w, then $\mathcal{M}'$ rejects w.*
- *If $\mathcal{M}$ does not halt on w, then $\mathcal{M}'$ does not halt on w.*

(Proof) Let $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}}, \delta \rangle$ be a k-tape TM.

On input w, the TM $\mathcal{M}'$ simulates the run of $\mathcal{M}$ on w, i.e., computing the run:

$$C_0 \ \vdash \ C_1 \ \vdash \ \cdots$$

From each C_i, it computes the next configuration C_{i+1}.

Some details on the proof of Theorem 6.1, part. 1

A configuration (of $\mathcal{M}$):

$$(q, \triangleleft u_1, \ldots\ldots\ldots, \triangleleft u_k)$$

is viewed as a string over the alphabet $Q \cup \Gamma \cup \{\tilde{\triangleleft}, \bullet\}$:

$$q\tilde{\triangleleft}u_1 \cdots\cdots\cdots \tilde{\triangleleft}u_k$$

One tape is sufficient to store this string.

Some details on the proof of Theorem 6.1, part. 1

A configuration (of $\mathcal{M}$):

$$(q, \triangleleft u_1, \ldots\ldots\ldots, \triangleleft u_k)$$

is viewed as a string over the alphabet $Q \cup \Gamma \cup \{\tilde{\triangleleft}, \bullet\}$:

$$q\tilde{\triangleleft}u_1 \cdots\cdots\cdots \tilde{\triangleleft}u_k$$

One tape is sufficient to store this string.

The symbol $\tilde{\triangleleft}$ is used to represent the left-end marker of $\mathcal{M}$.

Some details on the proof of Theorem 6.1, part. 2

(The algorithm/TM $\mathcal{M}'$) On input word w, do the following.

- Let C be the initial configuration of $\mathcal{M}$ on w.
- While (C is not a halting configuration of $\mathcal{M}$):

  $C :=$ the next configuration of C.

- If C is an accepting configuration, ACCEPT.
  If C is a rejecting configuration, REJECT.

Some details on the proof of Theorem 6.1, part. 2

(The algorithm/TM $\mathcal{M}'$) On input word w, do the following.

- Let C be the initial configuration of $\mathcal{M}$ on w.
- While (C is not a halting configuration of $\mathcal{M}$):

  $C :=$ the next configuration of C.

  - Scan the string C from left to right to find out the symbol read by each "head."

- If C is an accepting configuration, ACCEPT.

  If C is a rejecting configuration, REJECT.

Some details on the proof of Theorem 6.1, part. 2

(The algorithm/TM $\mathcal{M}'$) On input word w, do the following.

- Let C be the initial configuration of $\mathcal{M}$ on w.
- While (C is not a halting configuration of $\mathcal{M}$):

  $C :=$ the next configuration of C.

  - Scan the string C from left to right to find out the symbol read by each "head."
  - Move the head back to the beginning of the tape.

- If C is an accepting configuration, ACCEPT.
  If C is a rejecting configuration, REJECT.

Some details on the proof of Theorem 6.1, part. 2

(The algorithm/TM $\mathcal{M}'$) On input word w, do the following.

- Let C be the initial configuration of $\mathcal{M}$ on w.
- While (C is not a halting configuration of $\mathcal{M}$):

  $C :=$ the next configuration of C.

  - Scan the string C from left to right to find out the symbol read by each "head."
  - Move the head back to the beginning of the tape.
  - While moving back, change the state and the position of each head in C according to the transition function δ.
- If C is an accepting configuration, ACCEPT.
  If C is a rejecting configuration, REJECT.

Some details on the proof of Theorem 6.1, part. 2

(The algorithm/TM $\mathcal{M}'$) On input word w, do the following.

- Let C be the initial configuration of $\mathcal{M}$ on w.
- While (C is not a halting configuration of $\mathcal{M}$):

  $C :=$ the next configuration of C.

  - Scan the string C from left to right to find out the symbol read by each "head."
  - Move the head back to the beginning of the tape.
  - While moving back, change the state and the position of each head in C according to the transition function δ.
- If C is an accepting configuration, ACCEPT.
  If C is a rejecting configuration, REJECT.

(Note) $\mathcal{M}'$ uses only one "variable" C which can be stored in one tape.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\ \cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Proof of Theorem 6.1: Illustration

On input:

$\lhd$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\lhd$	q_0	$\tilde{\lhd}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\lhd}$	$\bullet$	$\tilde{\lhd}$	$\bullet$	$\cdots\ \cdots$	$\tilde{\lhd}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\lhd$	p	$\tilde{\lhd}$	$\cdots$	$\bullet$	a_1	$\cdots\ \cdots\ \cdots$	$\tilde{\lhd}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\ \cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\ \cdots\ \cdots$	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$
$\uparrow$													

Scan the string from left to right.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\ \cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\ \cdots\ \cdots$	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$
	$\uparrow$												

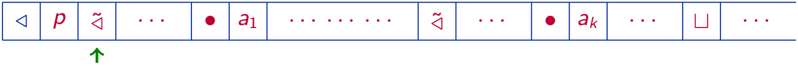
Scan the string from left to right.

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$)

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$
			$\uparrow$										

Scan the string from left to right.

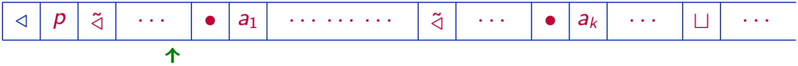
Remember p in the state (of $\mathcal{M}'$)

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$)

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$
				$\uparrow$									

Scan the string from left to right.

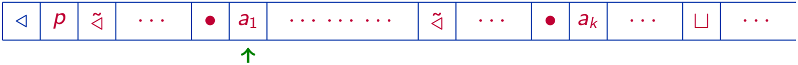
Remember p in the state (of $\mathcal{M}'$)

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1

Proof of Theorem 6.1: Illustration

On input:

$\lhd$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\lhd$	q_0	$\tilde{\lhd}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\lhd}$	$\bullet$	$\tilde{\lhd}$	$\bullet$	$\cdots\ \cdots$	$\tilde{\lhd}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\lhd$	p	$\tilde{\lhd}$	$\cdots$	$\bullet$	a_1	$\cdots\ \cdots\ \cdots$	$\tilde{\lhd}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$

Scan the string from left to right.

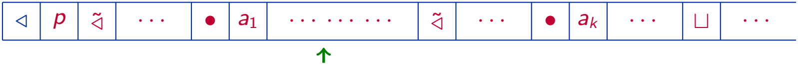
Remember p in the state (of $\mathcal{M}'$) and a_1

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\ \cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\ \cdots\ \cdots$	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$
						$\uparrow$							

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$ ↑	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$

Scan the string from left to right.

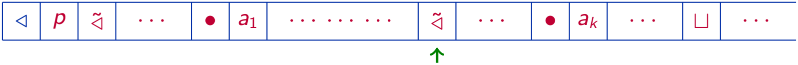
Remember p in the state (of $\mathcal{M}'$) and a_1

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

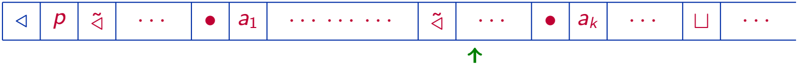
Remember p in the state (of $\mathcal{M}'$) and a_1 and so on

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

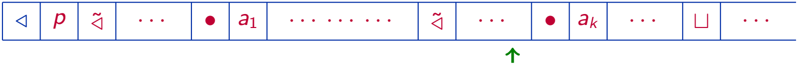
Remember p in the state (of $\mathcal{M}'$) and a_1 and so on

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$
									$\uparrow$				

Scan the string from left to right.

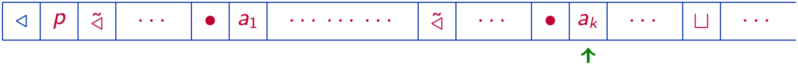
Remember p in the state (of $\mathcal{M}'$) and a_1 and so on

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$

Scan the string from left to right.

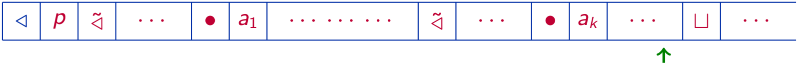
Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

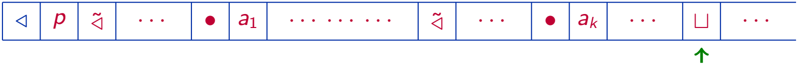
Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$
											$\uparrow$		

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$
											$\uparrow$		

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

$\lhd$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\lhd$	q_0	$\tilde{\lhd}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\lhd}$	$\bullet$	$\tilde{\lhd}$	$\bullet$	$\cdots\cdots$	$\tilde{\lhd}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\lhd$	p	$\tilde{\lhd}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$	$\tilde{\lhd}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$
										$\uparrow$			

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

$\lhd$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\lhd$	q_0	$\tilde{\lhd}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\lhd}$	$\bullet$	$\tilde{\lhd}$	$\bullet$	$\cdots\cdots$	$\tilde{\lhd}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\lhd$	p	$\tilde{\lhd}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$	$\tilde{\lhd}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$
									$\uparrow$				

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow$ w $\longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow$ w $\longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\ \cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\ \cdots\ \cdots$	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_k	$\cdots$	$\sqcup$	$\cdots$
										$\uparrow$			

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

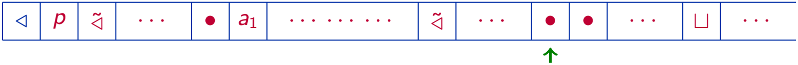
Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$	$\tilde{\triangleleft}$	$\cdots$	b	$\bullet$	$\cdots$	$\sqcup$	$\cdots$
								$\uparrow$					

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

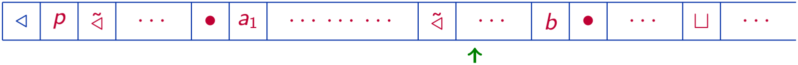
Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each • along the way.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$	$\tilde{\triangleleft}$	$\cdots$	b	$\bullet$	$\cdots$	$\sqcup$	$\cdots$
							↑						

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$	$\tilde{\triangleleft}$	$\cdots$	b	$\bullet$	$\cdots$	$\sqcup$	$\cdots$

↑

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$	$\tilde{\triangleleft}$	$\cdots$	b	$\bullet$	$\cdots$	$\sqcup$	$\cdots$
						$\uparrow$							

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

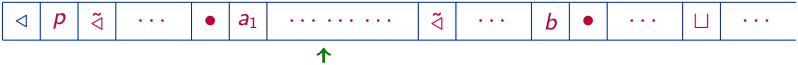
Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

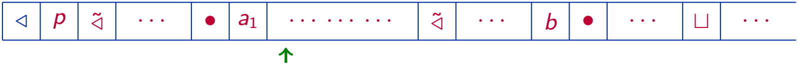
Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each • along the way.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\;\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\;\cdots\;\cdots$	$\tilde{\triangleleft}$	$\cdots$	b	$\bullet$	$\cdots$	$\sqcup$	$\cdots$
					$\uparrow$								

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$	$\tilde{\triangleleft}$	$\cdots$	b	$\bullet$	$\cdots$	$\sqcup$	$\cdots$
				$\uparrow$									

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots$	$\bullet$	a_1	$\cdots\cdots\cdots$	$\tilde{\triangleleft}$	$\cdots$	b	$\bullet$	$\cdots$	$\sqcup$	$\cdots$
			$\uparrow$										

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\ \cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots\bullet$	$\bullet$	a_1	$\cdots\ \cdots\ \cdots$	$\tilde{\triangleleft}$	$\cdots$	b	$\bullet$	$\cdots$	$\sqcup$	$\cdots$
				$\uparrow$									

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots\bullet$	b	a_1	$\cdots\cdots\cdots$	$\tilde{\triangleleft}$	$\cdots$	b	$\bullet$	$\cdots$	$\sqcup$	$\cdots$
					$\uparrow$								

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each • along the way.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\ \cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	p	$\tilde{\triangleleft}$	$\cdots\bullet$ ↑	b	c	$\cdots\ \cdots\ \cdots$	$\tilde{\triangleleft}$	$\cdots$	b	$\bullet$	$\cdots$	$\sqcup$	$\cdots$

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each • along the way.

Proof of Theorem 6.1: Illustration

On input:

$\lhd$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\lhd$	q_0	$\tilde{\lhd}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\lhd}$	$\bullet$	$\tilde{\lhd}$	$\bullet$	$\cdots\cdots$	$\tilde{\lhd}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\lhd$	p	$\tilde{\lhd}$	$\cdots\bullet$	b	c	$\cdots\cdots\cdots$	$\tilde{\lhd}$	$\cdots$	b	$\bullet$	$\cdots$	$\sqcup$	$\cdots$
		$\uparrow$											

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

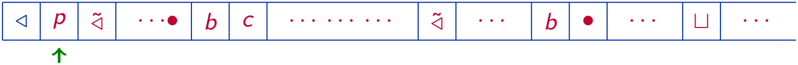
Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

Write the initial configuration of $\mathcal{M}$ on the tape:

Updating the current configuration:



Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

Proof of Theorem 6.1: Illustration

On input:

$\triangleleft$	$\longleftarrow w \longrightarrow$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

Write the initial configuration of $\mathcal{M}$ on the tape:

$\triangleleft$	q_0	$\tilde{\triangleleft}$	$\bullet$	$\longleftarrow w \longrightarrow$	$\tilde{\triangleleft}$	$\bullet$	$\tilde{\triangleleft}$	$\bullet$	$\cdots\ \cdots$	$\tilde{\triangleleft}$	$\bullet$	$\sqcup$	$\cdots$

Updating the current configuration:

$\triangleleft$	q	$\tilde{\triangleleft}$	$\cdots\bullet$	b	c	$\cdots\ \cdots\ \cdots$	$\tilde{\triangleleft}$	$\cdots$	b	$\bullet$	$\cdots$	$\sqcup$	$\cdots$
$\uparrow$													

Scan the string from left to right.

Remember p in the state (of $\mathcal{M}'$) and a_1 and so on until a_k.

Scan from right to left and update the position of each $\bullet$ along the way.

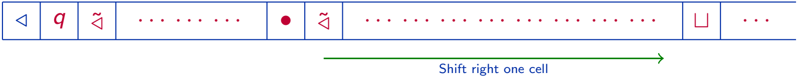
Proof: Illustration

Sometimes $\mathcal{M}'$ needs to shift right while updating the position of each $\bullet$:

$\triangleleft$	q	$\tilde{\triangleleft}$	$\cdots\ \cdots\ \cdots$	$\bullet$	$\tilde{\triangleleft}$	$\cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots\ \cdots$	$\sqcup$	$\cdots$

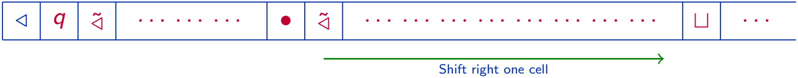
Proof: Illustration

Sometimes $\mathcal{M}'$ needs to shift right while updating the position of each $\bullet$:



Proof: Illustration

Sometimes $\mathcal{M}'$ needs to shift right while updating the position of each $\bullet$:



(Remark) Since the number of states in $\mathcal{M}$ is already fixed, it is not necessary to store the state q in the string C. The Turing machine $\mathcal{M}'$ can "remember" q in its states.

So it is sufficient to just store the content of each tape, i.e., the string C is of the form:

$$\tilde{\triangleleft} u_1 \cdots\cdots\cdots \tilde{\triangleleft} u_k$$

The equivalence between k-tape TM and 1-tape TM

Theorem 6.1

For every k-tape TM $\mathcal{M}$, where $k \geqslant 2$, there is a 1-tape TM $\mathcal{M}'$ such that for every input word w, the following holds.

- *If $\mathcal{M}$ accepts w, then $\mathcal{M}'$ accepts w.*
- *If $\mathcal{M}$ rejects w, then $\mathcal{M}'$ rejects w.*
- *If $\mathcal{M}$ does not halt on w, then $\mathcal{M}'$ does not halt on w.*

Table of contents

An informal definition of algorithm: A C++ like pseudo-code

We define an algorithm (informally) as a program of the form:

```
Boolean main (w)
{       statement;
          ⋮
        statement;   }
```

The input w is always a string.

An informal definition of algorithm: A C++ like pseudo-code

We define an algorithm (informally) as a program of the form:

```
Boolean main (w)
{       statement;
          ⋮
        statement;   }
```

The input w is always a string.

It also has some (finite number of) functions of the form:

```
Boolean/string function ⟨name⟩ (⟨var-name⟩,...,⟨var-name⟩)
{       statement;
          ⋮
        statement;   }
```

Note that functions always return Boolean or String values.

What is a “statement”?

What is a "statement"?

- $\langle$*var-name*$\rangle$:= $\langle$*``expression''*$\rangle$;

What is a "statement"?

- $\langle$*var-name*$\rangle$:= $\langle$*``expression''*$\rangle$;
- $\langle$*var-name*$\rangle$:= $\langle$*function-name*$\rangle$($\langle$*var-name*$\rangle$,...,$\langle$*var-name*$\rangle$);

What is a “statement”?

- $\langle$*var-name*$\rangle$:= $\langle$*‘‘expression’’*$\rangle$;
- $\langle$*var-name*$\rangle$:= $\langle$*function-name*$\rangle$($\langle$*var-name*$\rangle$,. . . ,$\langle$*var-name*$\rangle$);
- **return** $\langle$*var-name*$\rangle$; *or* **return** $\langle$*some-value*$\rangle$;

What is a “statement”?

- $\langle$*var-name*$\rangle$:= $\langle$*‘‘expression’’*$\rangle$;
- $\langle$*var-name*$\rangle$:= $\langle$*function-name*$\rangle$($\langle$*var-name*$\rangle$,...,$\langle$*var-name*$\rangle$);
- **return** $\langle$*var-name*$\rangle$; *or* **return** $\langle$*some-value*$\rangle$;
- **if** $\langle$*condition*$\rangle$

  { *statement;*

  ⋮

  statement; }

  else

  { *statement;*

  ⋮

  statement; }

What is a "statement"?

- $\langle \mathit{var\text{-}name} \rangle$:= $\langle$ ``*expression*'' $\rangle$;
- $\langle \mathit{var\text{-}name} \rangle$:= $\langle \mathit{function\text{-}name} \rangle$ ($\langle \mathit{var\text{-}name} \rangle$, ..., $\langle \mathit{var\text{-}name} \rangle$);
- **return** $\langle \mathit{var\text{-}name} \rangle$; *or* **return** $\langle \mathit{some\text{-}value} \rangle$;
- **if** $\langle \mathit{condition} \rangle$
  { *statement;*
  ⋮
  statement; }
  else
  { *statement;*
  ⋮
  statement; }

Variables can only store Boolean or string values. Of course, Boolean values can be viewed as string values.

There is no `while`-loop, since it can be implemented as a recursive function.

What is an “expression”?

An “expression” is loosely defined as any reasonable “basic” computation:

What is an “expression”?

An “expression” is loosely defined as any reasonable “basic” computation:

- Concatenating two strings.

What is an “expression”?

An “expression” is loosely defined as any reasonable “basic” computation:

- Concatenating two strings.
- Shift left/right of a string.

What is an "expression"?

An "expression" is loosely defined as any reasonable "basic" computation:

- Concatenating two strings.
- Shift left/right of a string.
- Change the symbol in a position in a string.

What is an “expression”?

An “expression” is loosely defined as any reasonable “basic” computation:

- Concatenating two strings.
- Shift left/right of a string.
- Change the symbol in a position in a string.

0-1 strings can be used to represent numbers, so “basic” computation includes:

What is an “expression”?

An “expression” is loosely defined as any reasonable “basic” computation:

- Concatenating two strings.
- Shift left/right of a string.
- Change the symbol in a position in a string.

0-1 strings can be used to represent numbers, so “basic” computation includes:

- Adding/subtracting/multiplying/dividing two numbers.

What is an "expression"?

An "expression" is loosely defined as any reasonable "basic" computation:

- Concatenating two strings.
- Shift left/right of a string.
- Change the symbol in a position in a string.

0-1 strings can be used to represent numbers, so "basic" computation includes:

- Adding/subtracting/multiplying/dividing two numbers.
- Enumerating all the numbers between 1 and some number n.

What is an "expression"?

An "expression" is loosely defined as any reasonable "basic" computation:

- Concatenating two strings.
- Shift left/right of a string.
- Change the symbol in a position in a string.

0-1 strings can be used to represent numbers, so "basic" computation includes:

- Adding/subtracting/multiplying/dividing two numbers.
- Enumerating all the numbers between 1 and some number n.
- Measuring the length of a 0-1 string.

What is an "expression"?

An "expression" is loosely defined as any reasonable "basic" computation:

- Concatenating two strings.
- Shift left/right of a string.
- Change the symbol in a position in a string.

0-1 strings can be used to represent numbers, so "basic" computation includes:

- Adding/subtracting/multiplying/dividing two numbers.
- Enumerating all the numbers between 1 and some number n.
- Measuring the length of a 0-1 string.
- Enumerating all the 0-1 strings with length between 1 and some number n.

What is an "expression"?

An "expression" is loosely defined as any reasonable "basic" computation:

- Concatenating two strings.
- Shift left/right of a string.
- Change the symbol in a position in a string.

0-1 strings can be used to represent numbers, so "basic" computation includes:

- Adding/subtracting/multiplying/dividing two numbers.
- Enumerating all the numbers between 1 and some number n.
- Measuring the length of a 0-1 string.
- Enumerating all the 0-1 strings with length between 1 and some number n.

(Note) Of course, we can add some other basic instructions/expressions. The point here is that we want to be convinced that any "algorithm" can be written in our pseudo-code.

Our pseudo-code and the Turing machines

```
Boolean main (w)
{        statement;
              .
              .
              .
         statement;     }
string function F1 (x,y,z)
{        statement;
              .
              .
              .
         statement;     }
.
.
.
Boolean function F200 (z)
{        statement;
              .
              .
              .
         statement;     }
```

Our pseudo-code and the Turing machines

```
Boolean main (w)
{       statement;
             .
             .
             .
        statement;    }
string function F1 (x,y,z)
{       statement;
             .
             .
             .
        statement;    }

   .
   .
   .
Boolean function F200 (z)
{       statement;
             .
             .
             .
        statement;    }
```

This pseudo-code can be translated into a Turing machine:

Our pseudo-code and the Turing machines

```
1:      Boolean main (w)
2:      {       statement;
  ⋮               ⋮
20:             statement;   }
21:     string function F1 (x,y,z)
22:     {       statement;
  ⋮               ⋮
45:             statement;   }
  ⋮
9536:   Boolean function F200 (z)
9537:   {       statement;
  ⋮               ⋮
9553:           statement;   }
```

This pseudo-code can be translated into a Turing machine:

- The line numbers are the states of the TM.

Our pseudo-code and the Turing machines

```
1:      Boolean main (w)
2:      {       statement;
  .                .
  .                .
  .                .
20:             statement;   }
21:     string function F1 (x,y,z)
22:     {       statement;
  .                .
  .                .
  .                .
45:             statement;   }
  .
  .
  .
9536:   Boolean function F200 (z)
9537:   {       statement;
  .                .
  .                .
  .                .
9553:           statement;   }
```

This pseudo-code can be translated into a Turing machine:

- The line numbers are the states of the TM.
- The variables are the tapes, i.e., one tape is used to represent one variable.

Our pseudo-code and the Turing machines

```
1:      Boolean main (w)
2:      {       statement;
 .                .
 .                .
 .                .
20:             statement;    }
21:     string function F1 (x,y,z)
22:     {       statement;
 .                .
 .                .
 .                .
45:             statement;    }
 .
 .
 .
9536:   Boolean function F200 (z)
9537:   {       statement;
 .                .
 .                .
 .                .
9553:           statement;    }
```

This pseudo-code can be translated into a Turing machine:

- The line numbers are the states of the TM.
- The variables are the tapes, i.e., one tape is used to represent one variable.
- When the `main` function returns `True` on input w, the TM accepts w.
  When the `main` function returns `False` on input w, the TM rejects w.

Our pseudo-code and the Turing machines

That every Turing machine can be translated to some form of algorithm is pretty obvious.

Our pseudo-code and the Turing machines

That every Turing machine can be translated to some form of algorithm is pretty obvious.

Theorem

Our C++-like pseudo-codes and Turing machines are equivalent.

Our pseudo-code and the Turing machines

That every Turing machine can be translated to some form of algorithm is pretty obvious.

Theorem
Our C++-like pseudo-codes and Turing machines are equivalent.

(Question) Does the Theorem establish Church-Turing thesis?

Church-Turing thesis
Every "algorithm" is equivalent to a Turing machine.

Our pseudo-code and the Turing machines

That every Turing machine can be translated to some form of algorithm is pretty obvious.

Theorem
Our C++-like pseudo-codes and Turing machines are equivalent.

(Question) Does the Theorem establish Church-Turing thesis?

Church-Turing thesis
Every "algorithm" is equivalent to a Turing machine.

(Hint) There is nothing wrong with our conversion of pseudo-codes to Turing machines. To spell it out exactly is not difficult, but it will be long and tedious.

The convention in this course

When we describe a Turing machine:

- We will describe it in some acceptable algorithm form.
- We will write ACCEPT to mean that the TM enters q_{acc} and REJECT to mean that the TM enters q_{rej}.
- In some cases when we need to be more precise, we will use our C++-like pseudo-code as the representation of a TM.

When do we use the formal definition of Turing machines?

We usually only use the formal definition of Turing machines (as defined in Lesson 5 and 6) when:

- we want to prove that some languages are undecidable,
- we want to prove that some languages are NP-complete,
- we want to construct a Turing machine that simulates other Turing machines.

Describing the simulation of a transition function (of a TM) is much easier than describing the simulation of a C++-like algorithm.

An example when we use the formal definition of Turing machines

In the proof of Theorem 6.1 we describe $\mathcal{M}'$ as an algorithm:

An example when we use the formal definition of Turing machines

In the proof of Theorem 6.1 we describe $\mathcal{M}'$ as an algorithm:

On input word w, $\mathcal{M}'$ does the following.
- Let C be the initial configuration of $\mathcal{M}$ on w.
- While (C is not a halting configuration of $\mathcal{M}$):
  $C :=$ the next configuration of C.
- If C is an accepting configuration, ACCEPT.
  If C is a rejecting configuration, REJECT.

An example when we use the formal definition of Turing machines

In the proof of Theorem 6.1 we describe $\mathcal{M}'$ as an algorithm:

On input word w, $\mathcal{M}'$ does the following.
- Let C be the initial configuration of $\mathcal{M}$ on w.
- While (C is not a halting configuration of $\mathcal{M}$):
  $C :=$ the next configuration of C.
- If C is an accepting configuration, ACCEPT.
  If C is a rejecting configuration, REJECT.

But we use the formal definition of TM for $\mathcal{M}$.

Recall

(Def.) We say that $\mathcal{M}$ *recognizes* a language L, if for every input word w:

- if $w \in L$, then $\mathcal{M}$ accepts w;
- if $w \notin L$, then $\mathcal{M}$ does not accept w, i.e., either it does not halt on w or rejects w.

A language L is recognizable, if there is a TM that recognizes L.

Recall

(Def.) We say that $\mathcal{M}$ *recognizes* a language L, if for every input word w:

- if $w \in L$, then $\mathcal{M}$ accepts w;
- if $w \notin L$, then $\mathcal{M}$ does not accept w, i.e., either it does not halt on w or rejects w.

A language L is recognizable, if there is a TM that recognizes L.

(Def.) We say that $\mathcal{M}$ *decides* a language L, if for every input word w:

- if $w \in L$, then $\mathcal{M}$ accepts w;
- if $w \notin L$, then $\mathcal{M}$ rejects w.

A language L is decidable, if there is a TM that decides L.

Recall

(Def.) We say that $\mathcal{M}$ *recognizes* a language L, if for every input word w:

- if $w \in L$, then $\mathcal{M}$ accepts w;
- if $w \notin L$, then $\mathcal{M}$ does not accept w, i.e., either it does not halt on w or rejects w.

A language L is recognizable, if there is a TM that recognizes L.

(Def.) We say that $\mathcal{M}$ *decides* a language L, if for every input word w:

- if $w \in L$, then $\mathcal{M}$ accepts w;
- if $w \notin L$, then $\mathcal{M}$ rejects w.

A language L is decidable, if there is a TM that decides L.

(Note) To prove the existence of $\mathcal{M}$, we usually describe $\mathcal{M}$ as an algorithm.

Example of algorithms that recognize and decide a language

Consider:

$$L = \{w \mid \text{the number of 1 in } w \text{ is even}\}$$

Example of algorithms that recognize and decide a language

Consider:

$$L = \{w \mid \text{the number of 1 in } w \text{ is even}\}$$

The following algorithm decides L:

On input word w:

- Count the number of 1 in w.
- If it is even, ACCEPT.
- If it is odd, REJECT.

Example of algorithms that recognize and decide a language

Consider:

$$L = \{w \mid \text{the number of 1 in } w \text{ is even}\}$$

The following algorithm decides L:

On input word w:
- Count the number of 1 in w.
- If it is even, ACCEPT.
- If it is odd, REJECT.

The following algorithm recognizes L:

On input word w:
- Count the number of 1 in w.
- If it is even, ACCEPT.
- If it is odd, enter an infinite loop.

Table of contents

Decidable and recognizable languages

Theorem 6.4

- *If a language L (over the alphabet Σ) is decidable, so is its complement $\Sigma^* - L$.*
- *If both a language L and its complement $\Sigma^* - L$ are recognizable, then L is decidable.*

Decidable and recognizable languages

Theorem 6.4

- *If a language L (over the alphabet Σ) 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 6.5

Decidable languages are closed under union, intersection, concatenation and Kleene star.

Decidable and recognizable languages

Theorem 6.4

- *If a language L (over the alphabet Σ) 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 6.5

Decidable languages are closed under union, intersection, concatenation and Kleene star.

Theorem 6.6

Recognizable languages are closed under union and intersection.

Decidable languages — Proof of Theorem 6.4: The first item

Theorem 6.4 (The first item)

- *If a language L (over the alphabet Σ) is decidable, so is its complement $\Sigma^* - L$.*

(Proof) The first item is trivial.

Let $\mathcal{M}$ be a TM that decides L. By switching its accept and reject states, we get a TM that decides its complement.

Decidable languages — Proof of Theorem 6.4: The second item

Theorem 6.4 (The second item)

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

Decidable languages — Proof of Theorem 6.4: The second item

Theorem 6.4 (The second item)

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

(Proof) Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be 1-tape TM that recognize L and $\Sigma^* - L$, respectively. We describe 2-tape TM $\mathcal{M}$ that decides L. On input w:

- Copy the input word onto the second tape.
- Run $\mathcal{M}_1$ on the first tape and $\mathcal{M}_2$ on the second tape "simultaneously."
- If $\mathcal{M}_1$ accepts, then ACCEPT. If $\mathcal{M}_2$ accepts, then REJECT.

Decidable languages — Proof of Theorem 6.4: The second item

Theorem 6.4 (The second item)

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

(Proof) Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be 1-tape TM that recognize L and $\Sigma^* - L$, respectively. We describe 2-tape TM $\mathcal{M}$ that decides L. On input w:

- Copy the input word onto the second tape.
- Run $\mathcal{M}_1$ on the first tape and $\mathcal{M}_2$ on the second tape "simultaneously."
- If $\mathcal{M}_1$ accepts, then ACCEPT. If $\mathcal{M}_2$ accepts, then REJECT.

For every input word $w \in \Sigma^*$, either $w \in L$ or $w \in \Sigma^* - L$, and hence, w is accepted either by $\mathcal{M}_1$ or by $\mathcal{M}_2$.

Decidable languages — Proof of Theorem 6.4: The second item

Theorem 6.4 (The second item)

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

(Proof) Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be 1-tape TM that recognize L and $\Sigma^* - L$, respectively. We describe 2-tape TM $\mathcal{M}$ that decides L. On input w:

- Copy the input word onto the second tape.
- Run $\mathcal{M}_1$ on the first tape and $\mathcal{M}_2$ on the second tape "simultaneously."
- If $\mathcal{M}_1$ accepts, then ACCEPT. If $\mathcal{M}_2$ accepts, then REJECT.

For every input word $w \in \Sigma^*$, either $w \in L$ or $w \in \Sigma^* - L$, and hence, w is accepted either by $\mathcal{M}_1$ or by $\mathcal{M}_2$.

Therefore, for every input word w, the TM $\mathcal{M}$ halts, and accepts if and only if $w \in L$.

Theorem 6.4 (The second item)

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

(Proof) Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be 1-tape TM that recognize L and $\Sigma^* - L$, respectively. We describe 2-tape TM $\mathcal{M}$ that decides L. On input w:

- Copy the input word onto the second tape.
- Run $\mathcal{M}_1$ on the first tape and $\mathcal{M}_2$ on the second tape "simultaneously."
- If $\mathcal{M}_1$ accepts, then ACCEPT. If $\mathcal{M}_2$ accepts, then REJECT.

For every input word $w \in \Sigma^*$, either $w \in L$ or $w \in \Sigma^* - L$, and hence, w is accepted either by $\mathcal{M}_1$ or by $\mathcal{M}_2$.

Therefore, for every input word w, the TM $\mathcal{M}$ halts, and accepts if and only if $w \in L$.

(See Note 6 for more details on running $\mathcal{M}_1$ and $\mathcal{M}_2$ "simultaneously.")

Closure properties of decidable languages — Proof of Theorem 6.5

Theorem 6.5

Decidable languages are closed under union, intersection, concatenation and Kleene star.

Closure properties of decidable languages — Proof of Theorem 6.5

Theorem 6.5

Decidable languages are closed under union, intersection, concatenation and Kleene star.

(Proof) Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be the TM that decide languages L_1 and L_2, respectively.

Closure properties of decidable languages — Proof of Theorem 6.5

Theorem 6.5
Decidable languages are closed under union, intersection, concatenation and Kleene star.

(Proof) Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be the TM that decide languages L_1 and L_2, respectively.

(Closure under union) The TM decides $L_1 \cup L_2$ works as follows. On input word w, it runs $\mathcal{M}_1$ on w and then $\mathcal{M}_2$ on w. It accepts if and only if at least one of $\mathcal{M}_1$ or $\mathcal{M}_2$ accepts.

Theorem 6.5
Decidable languages are closed under union, intersection, concatenation and Kleene star.

(Proof) Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be the TM that decide languages L_1 and L_2, respectively.

(Closure under union) The TM decides $L_1 \cup L_2$ works as follows. On input word w, it runs $\mathcal{M}_1$ on w and then $\mathcal{M}_2$ on w. It accepts if and only if at least one of $\mathcal{M}_1$ or $\mathcal{M}_2$ accepts.

(Closure under intersection) Similar to the above.

Closure properties of decidable languages — Proof of Theorem 6.5

(Closure under concatenation) The TM that decides $L_1 \cdot L_2$ works as follows.
On input word w:

- For all possible pairs (v_1, v_2) such that $v_1 v_2 = w$:
  Check if $\mathcal{M}_1$ accepts v_1 and $\mathcal{M}_2$ accepts v_2.
- ACCEPT, if there is a pair (v_1, v_2) where v_1 is accepted by $\mathcal{M}_1$ and v_2 is accepted by $\mathcal{M}_2$.
  REJECT, otherwise.

Closure properties of decidable languages — Proof of Theorem 6.5

(Closure under concatenation) The TM that decides $L_1 \cdot L_2$ works as follows. On input word w:

- For all possible pairs (v_1, v_2) such that $v_1 v_2 = w$:
  Check if $\mathcal{M}_1$ accepts v_1 and $\mathcal{M}_2$ accepts v_2.
- ACCEPT, if there is a pair (v_1, v_2) where v_1 is accepted by $\mathcal{M}_1$ and v_2 is accepted by $\mathcal{M}_2$.
  REJECT, otherwise.

(Closure under Kleene star) Similar to the above. See Note 6.

Closure properties of recognizable languages — Proof of Theorem 6.6

Theorem 6.6

Recognizable languages are closed under union and intersection.

Closure properties of recognizable languages — Proof of Theorem 6.6

Theorem 6.6
Recognizable languages are closed under union and intersection.

(Proof) Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be 1-tape TM that recognize languages L_1 and L_2, respectively.

Closure properties of recognizable languages — Proof of Theorem 6.6

Theorem 6.6
Recognizable languages are closed under union and intersection.

(Proof) Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be 1-tape TM that recognize languages L_1 and L_2, respectively.

(Closure under union) The TM that recognizes $L_1 \cup L_2$ works as follows. It has two tapes. On input word w:

- Copy the input word onto the second tape.
- Run $\mathcal{M}_1$ on the first tape and $\mathcal{M}_2$ on the second tape "simultaneously."
- ACCEPT, if at least one of $\mathcal{M}_1$ or $\mathcal{M}_2$ accepts.

Every word $w \in L_1 \cup L_2$ is accepted by at least one of $\mathcal{M}_1$ or $\mathcal{M}_2$. Thus, the TM above recognizes the language $L_1 \cup L_2$ correctly.
(What happens to the TM when $w \notin L_1 \cup L_2$?)

Closure properties of recognizable languages — Proof of Theorem 6.6

Theorem 6.6
Recognizable languages are closed under union and intersection.

(Proof) Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be 1-tape TM that recognize languages L_1 and L_2, respectively.

(Closure under union) The TM that recognizes $L_1 \cup L_2$ works as follows. It has two tapes. On input word w:

- Copy the input word onto the second tape.
- Run $\mathcal{M}_1$ on the first tape and $\mathcal{M}_2$ on the second tape "simultaneously."
- ACCEPT, if at least one of $\mathcal{M}_1$ or $\mathcal{M}_2$ accepts.

Every word $w \in L_1 \cup L_2$ is accepted by at least one of $\mathcal{M}_1$ or $\mathcal{M}_2$. Thus, the TM above recognizes the language $L_1 \cup L_2$ correctly.
(What happens to the TM when $w \notin L_1 \cup L_2$?)

(Closure under intersection) Similar to the above.

Some properties of decidable and recognizable languages

Theorem 6.4

- *If a language L (over the alphabet Σ) is decidable, so is its complement $\Sigma^* - L$.*
- *If both a language L and its complement $\Sigma^* - L$ are recognizable, then L is decidable.*

Some properties of decidable and recognizable languages

Theorem 6.4

- *If a language L (over the alphabet Σ) 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 6.5

Decidable languages are closed under union, intersection, concatenation and Kleene star.

Some properties of decidable and recognizable languages

Theorem 6.4

- *If a language L (over the alphabet Σ) 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 6.5

Decidable languages are closed under union, intersection, concatenation and Kleene star.

Theorem 6.6

Recognizable languages are closed under union and intersection.

Some remarks

(Remark) Recognizable languages are also closed under concatenation and Kleene star.

Some remarks

(Remark) Recognizable languages are also closed under concatenation and Kleene star.

We can already prove it in this lesson, but the proof is a bit technical.

Some remarks

(Remark) Recognizable languages are also closed under concatenation and Kleene star.

We can already prove it in this lesson, but the proof is a bit technical.

So we postpone the proof until Lesson 9, where we will use “non-deterministic” TM to obtain a neater and clearer proof.

Some remarks

(Remark) Recognizable languages are also closed under concatenation and Kleene star.

We can already prove it in this lesson, but the proof is a bit technical.

So we postpone the proof until Lesson 9, where we will use “non-deterministic” TM to obtain a neater and clearer proof.

(Remark) Recognizable languages are *not!* closed under complement. We will see this in Lesson 7.

End of Lesson 6