Fundamental Computer Science
Sequence 1: Turing Machines

MoSIG-M1Info, 2021

February 1, 2021

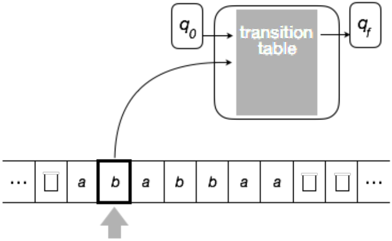
Aim and content

We present in detail the classical computational model of **Turing Machine**.

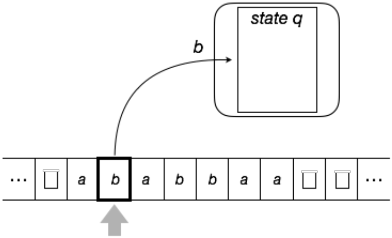
The objective is to understand the basic mechanisms and to learn the underlying formalism.

Description of the Turing Machine

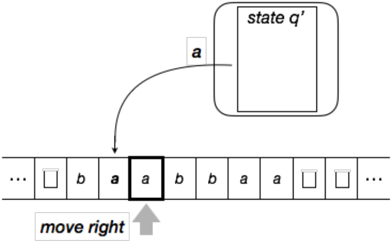
- memory: an infinite tape
  - initially, it contains the input string
  - move the head left or right
  - read and/or write to current cell
- control (transition table)
  - finite number of states
  - one current state
- At each step:
  – move from state to state
  – read/write or move Left/Right in the tape



q0
transition table
qf
…
a
b
a
b
b
a
a
…



state q
b
…
a
b
a
b
b
a
a
…


state q'
a
… b a a b b a a …
move right

Turing machine: formal definition

A Turing Machine (M) is a six-tuple $(K, \Sigma, \Gamma, \delta, s, H)$, where

- K is a finite set of states
- Σ is the input alphabet not containing the *blank* symbol $\sqcup$
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $s \in K$: the initial state
- $H \subseteq K$: the set of halting states
- δ: the transition *function* from $(K \setminus H) \times \Gamma$ to $K \times (\Gamma \cup \{\leftarrow, \rightarrow\})$

Turing machine: formal definition

A Turing Machine (M) is a six-tuple $(K, \Sigma, \Gamma, \delta, s, H)$, where

- K is a finite set of states
- Σ is the input alphabet not containing the *blank* symbol $\sqcup$
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $s \in K$: the initial state
- $H \subseteq K$: the set of halting states
- δ: the transition *function* from $(K \setminus H) \times \Gamma$ to $K \times (\Gamma \cup \{\leftarrow, \rightarrow\})$

In general, $\delta(q, a) = (p, b)$ means that when M is in the state q and reads a in the tape, it goes to the state p and

– if $b \in \Sigma$, writes b in the place of a
– if $b \in \{\leftarrow, \rightarrow\}$, moves the head either Left or Right

$$q \xrightarrow{a\,:\,b} p$$

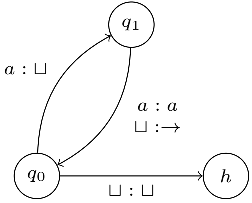
A first example (2 representations)

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q,\sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



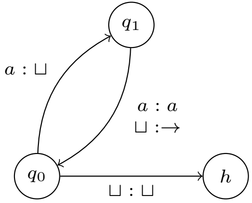
A first example (2 representations)

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q,\sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$(q_0, \underline{a}aa)$$

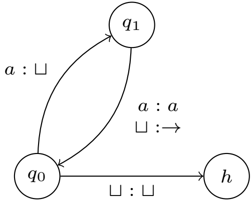
A first example (2 representations)

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q,\sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$(q_0, \underline{a}aa) \quad \vdash_M \quad (q_1, \underline{\sqcup}aa)$$

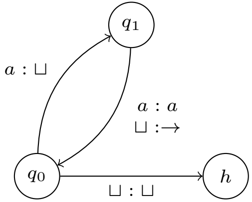
A first example (2 representations)

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$(q_0, \underline{a}aa) \quad \vdash_M \quad (q_1, \underline{\sqcup}aa) \ \vdash_M \ (q_0, \sqcup\underline{a}a)$$

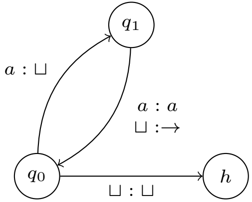
A first example (2 representations)

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$\begin{aligned} (q_0, \underline{a}aa) \quad & \vdash_M \quad (q_1, \underline{\sqcup}aa) \vdash_M (q_0, \sqcup\underline{a}a) \\ & \vdash_M \quad (q_1, \sqcup\underline{\sqcup}a) \end{aligned}$$

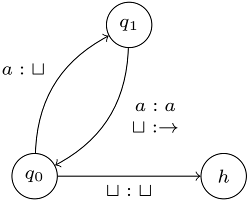
A first example (2 representations)

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q,\sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$\begin{aligned}(q_0, \underline{a}aa) &\vdash_M (q_1, \underline{\sqcup}aa) \vdash_M (q_0, \sqcup\underline{a}a)\\ &\vdash_M (q_1, \sqcup\underline{\sqcup}a) \vdash_M (q_0, \sqcup\sqcup\underline{a})\end{aligned}$$

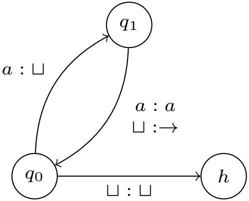
A first example (2 representations)

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$\begin{aligned}
(q_0, \underline{a}aa) \quad &\vdash_M \quad (q_1, \underline{\sqcup}aa) \vdash_M (q_0, \sqcup\underline{a}a) \\
&\vdash_M \quad (q_1, \sqcup\underline{\sqcup}a) \vdash_M (q_0, \sqcup \sqcup \underline{a}) \\
&\vdash_M \quad (q_1, \sqcup \sqcup \underline{\sqcup})
\end{aligned}$$

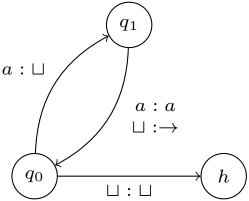
A first example (2 representations)

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$\begin{aligned}
(q_0, \underline{a}aa) \quad &\vdash_M \quad (q_1, \underline{\sqcup}aa) \vdash_M (q_0, \sqcup\underline{a}a) \\
&\vdash_M \quad (q_1, \sqcup\underline{\sqcup}a) \vdash_M (q_0, \sqcup\sqcup\underline{a}) \\
&\vdash_M \quad (q_1, \sqcup\sqcup\underline{\sqcup}) \vdash_M (q_0, \sqcup\sqcup\sqcup\underline{\sqcup})
\end{aligned}$$

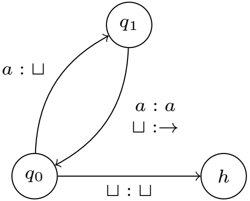
A first example (2 representations)

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$\begin{aligned}
(q_0, \underline{a}aa) \quad &\vdash_M \quad (q_1, \underline{\sqcup}aa) \ \vdash_M \ (q_0, \sqcup\underline{a}a) \\
&\vdash_M \quad (q_1, \sqcup\underline{\sqcup}a) \ \vdash_M \ (q_0, \sqcup\sqcup\underline{a}) \\
&\vdash_M \quad (q_1, \sqcup\sqcup\underline{\sqcup}) \ \vdash_M \ (q_0, \sqcup\sqcup\sqcup\underline{\sqcup}) \\
&\vdash_M \quad (h, \sqcup\sqcup\sqcup\underline{\sqcup})
\end{aligned}$$

Formalize the notation

Definition

A **configuration** of a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ is a member of $K \times \Gamma^* \times \Gamma^*((\Gamma \setminus \{\sqcup\}) \cup \{\epsilon\})$.

- informally: a triplet describing
  - the current state
  - the contents of the tape at the left of the head (including head's position)
  - the contents of the tape at the right of the head

Formalize the notation

Definition

A **configuration** of a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ is a member of $K \times \Gamma^* \times \Gamma^*((\Gamma \setminus \{\sqcup\}) \cup \{\epsilon\})$.

- informally: a triplet describing
  - the current state
  - the contents of the tape at the left of the head (including head's position)
  - the contents of the tape at the right of the head
- example: $(q_1, \sqcup a, a)$ or simply $(q_1, \sqcup\underline{a}a)$ or simply $(q_1, \underline{a}a)$

Formalize the notation

Definition

A **configuration** of a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ is a member of $K \times \Gamma^* \times \Gamma^*((\Gamma \setminus \{\sqcup\}) \cup \{\epsilon\})$.

- informally: a triplet describing
  - the current state
  - the contents of the tape at the left of the head (including head's position)
  - the contents of the tape at the right of the head
- example: $(q_1, \sqcup a, a)$ or simply $(q_1, \sqcup \underline{a} a)$ or simply $(q_1, \underline{a} a)$

Initial configuration: $(s, \underline{a} w)$ where $M = (K, \Sigma, \Gamma, \delta, s, H)$ is a Turing Machine, $a \in \Sigma$, $w \in \Sigma^*$ and aw is the *input string*

Formalize the notation

Definition

A **configuration** of a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ is a member of $K \times \Gamma^* \times \Gamma^*((\Gamma \setminus \{\sqcup\}) \cup \{\epsilon\})$.

- informally: a triplet describing
  - the current state
  - the contents of the tape at the left of the head (including head's position)
  - the contents of the tape at the right of the head
- example: $(q_1, \sqcup a, a)$ or simply $(q_1, \sqcup \underline{a} a)$ or simply $(q_1, \underline{a} a)$

Initial configuration: $(s, \underline{a} w)$ where $M = (K, \Sigma, \Gamma, \delta, s, H)$ is a Turing Machine, $a \in \Sigma$, $w \in \Sigma^*$ and aw is the *input string*

Halted configuration: a configuration whose state belongs to H

- example: $(h, \sqcup \sqcup \sqcup \sqcup, \epsilon)$ or simply $(h, \sqcup \sqcup \sqcup \underline{\sqcup})$ or simply $(h, \underline{\sqcup})$

Formalize the notation

Definition

Consider a Turing Machine M and two configurations C_1 and C_2 of M. If M can go from C_1 to C_2 in a *single step*, then we write

$$C_1 \vdash_M C_2$$

Formalize the notation

Definition

Consider a Turing Machine M and two configurations C_1 and C_2 of M. If M can go from C_1 to C_2 in a *single step*, then we write

$$C_1 \vdash_M C_2$$

Definition

Consider a Turing Machine M and two configurations C_1 and C_2 of M. If M can go from C_1 to C_2 using a *sequence* of configurations, then we say that C_1 **yields** C_2 and we write

$$C_1 \vdash_M^* C_2$$

Formalize the notation

Definition

Consider a Turing Machine M and two configurations C_1 and C_2 of M. If M can go from C_1 to C_2 in a *single step*, then we write

$$C_1 \vdash_M C_2$$

Definition

Consider a Turing Machine M and two configurations C_1 and C_2 of M. If M can go from C_1 to C_2 using a *sequence* of configurations, then we say that C_1 **yields** C_2 and we write

$$C_1 \vdash_M^* C_2$$

Definition

A **computation** of a Turing Machine M is a sequence of configurations $C_0, C_1, \ldots, C_n$, for some $n \geq 0$, such that

$$C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \ldots \vdash_M C_n$$

The **length** of the computation is n (or it performs n steps).

Turing Machines and automata

Turing Machines are (*augmented*) finite states automata.

Not detailed here
see the following link to get an idea of the powers.

http://www.jflap.org/

Determinism or not?

Implicitly, the transition δ is deterministic.

Determinism or not?

Implicitly, the transition δ is deterministic.

Non-deterministic Turing Machine

What happens if several outputs are allowed at each step?

The choice is among k fixed possibilities, random, round-robin, etc.

Determinism or not?

Implicitly, the transition δ is deterministic.

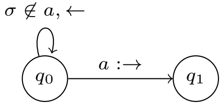
Non-deterministic Turing Machine

What happens if several outputs are allowed at each step?

The choice is among k fixed possibilities, random, round-robin, etc.

This point is important and it will be detailed in a separate lecture.

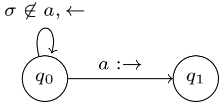
A more general notation for Turing Machines



Turing Machine $L_a = (K, \Sigma, \Gamma, \delta, s, H)$ where:

- $K = \{q_0, q_1\}$
- $a \in \Sigma$
- $s = q_0$
- $H = \{q_1\}$

A more general notation for Turing Machines

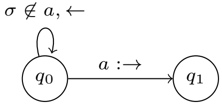


Turing Machine $L_a = (K, \Sigma, \Gamma, \delta, s, H)$ where:

- $K = \{q_0, q_1\}$
- $a \in \Sigma$
- $s = q_0$
- $H = \{q_1\}$

- Define similar simple Turing Machines
  - examples: L, R, L_a, R_a, L^2, R^2, a, $\sqcup$, etc.

A more general notation for Turing Machines



Turing Machine $L_a = (K, \Sigma, \Gamma, \delta, s, H)$ where:

- $K = \{q_0, q_1\}$
- $a \in \Sigma$
- $s = q_0$
- $H = \{q_1\}$

- Define similar simple Turing Machines
  - examples: L, R, L_a, R_a, L^2, R^2, a, $\sqcup$, etc.

- Combine simple machines to construct more complicated ones

$$\begin{tikzcd} M_3 & \\ M_1 \arrow[u, "b"'] \arrow[r, "a"] & M_2 \end{tikzcd}$$

1. Run M_1
2. If M_1 finishes and the head reads a then run M_2 starting from this a
3. Else run M_3 starting from this b

Example

What is the goal of the following Turing Machine?

$$
\begin{tikzcd}
> L_{\sqcup} \arrow[r] & R \arrow[rr, "a \neq \sqcup"] \arrow[d, "\sqcup"] & & \sqcup R^2_{\sqcup} a L^2_{\sqcup} a \arrow[ll, bend right] \\
& R_{\sqcup} & &
\end{tikzcd}
$$

Example

What is the goal of the following Turing Machine?

$$
\begin{tikzcd}
> L_\sqcup \arrow[r] & R \arrow[r, "a \neq \sqcup"] \arrow[d, "\sqcup"] & \sqcup R_\sqcup^2 a L_\sqcup^2 a \arrow[l, bend right] \\
& R_\sqcup &
\end{tikzcd}
$$

$$(\sqcup abc\underline{\sqcup}) \quad \vdash_M^* \quad (\underline{\sqcup}abc\sqcup) \qquad\qquad (L_\sqcup)$$

Example

What is the goal of the following Turing Machine?

$$
\begin{tikzcd}
> L_{\sqcup} \arrow[r] & R \arrow[r, "a \neq \sqcup"] \arrow[d, "\sqcup"] & \sqcup R_{\sqcup}^2 a L_{\sqcup}^2 a \arrow[l, bend right] \\
& R_{\sqcup} &
\end{tikzcd}
$$

$$
\begin{array}{llll}
(\sqcup abc\underline{\sqcup}) & \vdash_M^* & (\underline{\sqcup}abc\sqcup) & (L_{\sqcup}) \\
& \vdash_M & (\sqcup\underline{a}bc\sqcup) & (R)
\end{array}
$$

Example

What is the goal of the following Turing Machine?

$$
\begin{tikzcd}
> L_{\sqcup} \arrow[r] & R \arrow[r, "a \neq \sqcup"] \arrow[d, "\sqcup"] & \sqcup R^2_{\sqcup} a L^2_{\sqcup} a \arrow[l, bend right] \\
& R_{\sqcup} &
\end{tikzcd}
$$

$$
\begin{array}{llll}
(\sqcup abc\underline{\sqcup}) & \vdash_M^* & (\underline{\sqcup}abc\sqcup) & (L_{\sqcup}) \\
 & \vdash_M & (\sqcup\underline{a}bc\sqcup) & (R) \\
 & \vdash_M & (\sqcup\underline{\sqcup}bc\sqcup) & (\sqcup)
\end{array}
$$

Example

What is the goal of the following Turing Machine?

$$\begin{tikzcd} > L_{\sqcup} \arrow[r] & R \arrow[r, "a \neq \sqcup"] \arrow[d, "\sqcup"] & \sqcup R^2_{\sqcup} a L^2_{\sqcup} a \arrow[l, bend right] \\ & R_{\sqcup} & \end{tikzcd}$$

$$\begin{aligned}
(\sqcup abc\underline{\sqcup}) \quad & \vdash^*_M \quad (\underline{\sqcup}abc\sqcup) && (L_{\sqcup}) \\
& \vdash_M \quad (\sqcup\underline{a}bc\sqcup) && (R) \\
& \vdash_M \quad (\sqcup\underline{\sqcup}bc\sqcup) && (\sqcup) \\
& \vdash^*_M \quad (\sqcup \sqcup bc \sqcup \underline{\sqcup}) && (R^2_{\sqcup})
\end{aligned}$$

Example

What is the goal of the following Turing Machine?

$$\begin{tikzcd}
> L_{\sqcup} \arrow[r] & R \arrow[r, "a \neq \sqcup"] \arrow[d, "\sqcup"] & \sqcup R^2_{\sqcup} a L^2_{\sqcup} a \arrow[l, bend right] \\
& R_{\sqcup} &
\end{tikzcd}$$

$$\begin{array}{llll}
(\sqcup abc\underline{\sqcup}) & \vdash^*_M & (\underline{\sqcup}abc\sqcup) & (L_{\sqcup}) \\
 & \vdash_M & (\sqcup\underline{a}bc\sqcup) & (R) \\
 & \vdash_M & (\sqcup\underline{\sqcup}bc\sqcup) & (\sqcup) \\
 & \vdash^*_M & (\sqcup \sqcup bc \sqcup \underline{\sqcup}) & (R^2_{\sqcup}) \\
 & \vdash_M & (\sqcup \sqcup bc \sqcup \underline{a}) & (a)
\end{array}$$

Example

What is the goal of the following Turing Machine?

$$
\begin{tikzcd}
> L_{\sqcup} \arrow[r] & R \arrow[r, "a \neq \sqcup"] \arrow[d, "\sqcup"] & \sqcup R^2_{\sqcup} a L^2_{\sqcup} a \arrow[l, bend right] \\
 & R_{\sqcup} &
\end{tikzcd}
$$

$$
\begin{aligned}
(\sqcup abc\underline{\sqcup}) \quad & \vdash^*_M \quad (\underline{\sqcup}abc\sqcup) && (L_{\sqcup}) \\
& \vdash_M \quad (\sqcup\underline{a}bc\sqcup) && (R) \\
& \vdash_M \quad (\sqcup\underline{\sqcup}bc\sqcup) && (\sqcup) \\
& \vdash^*_M \quad (\sqcup \sqcup bc \sqcup \underline{\sqcup}) && (R^2_{\sqcup}) \\
& \vdash_M \quad (\sqcup \sqcup bc \sqcup \underline{a}) && (a) \\
& \vdash^*_M \quad (\sqcup\underline{\sqcup}bc \sqcup a) && (L^2_{\sqcup})
\end{aligned}
$$

Example

What is the goal of the following Turing Machine?

$$
\begin{tikzcd}
> L_{\sqcup} \arrow[r] & R \arrow[r, "a \neq \sqcup"] \arrow[d, "\sqcup"] & \sqcup R^2_{\sqcup} a L^2_{\sqcup} a \arrow[l, bend right] \\
& R_{\sqcup} &
\end{tikzcd}
$$

$$
\begin{array}{llll}
(\sqcup abc\underline{\sqcup}) & \vdash_M^* & (\underline{\sqcup}abc\sqcup) & (L_{\sqcup}) \\
 & \vdash_M & (\sqcup\underline{a}bc\sqcup) & (R) \\
 & \vdash_M & (\sqcup\underline{\sqcup}bc\sqcup) & (\sqcup) \\
 & \vdash_M^* & (\sqcup \sqcup bc \sqcup \underline{\sqcup}) & (R^2_{\sqcup}) \\
 & \vdash_M & (\sqcup \sqcup bc \sqcup \underline{a}) & (a) \\
 & \vdash_M^* & (\sqcup\underline{\sqcup}bc \sqcup a) & (L^2_{\sqcup}) \\
 & \vdash_M & (\sqcup\underline{a}bc \sqcup a) & (a)
\end{array}
$$

Example

What is the goal of the following Turing Machine?

$$
\begin{tikzcd}
> L_{\sqcup} \arrow[r] & R \arrow[r, "a \neq \sqcup"] \arrow[d, "\sqcup"] & \sqcup R_{\sqcup}^{2} a L_{\sqcup}^{2} a \arrow[l, bend right] \\
 & R_{\sqcup} &
\end{tikzcd}
$$

$$
\begin{array}{llll}
(\sqcup abc\underline{\sqcup}) & \vdash_M^* & (\underline{\sqcup}abc\sqcup) & (L_{\sqcup}) \\
 & \vdash_M & (\sqcup\underline{a}bc\sqcup) & (R) \\
 & \vdash_M & (\sqcup\underline{\sqcup}bc\sqcup) & (\sqcup) \\
 & \vdash_M^* & (\sqcup \sqcup bc \sqcup \underline{\sqcup}) & (R_{\sqcup}^{2}) \\
 & \vdash_M & (\sqcup \sqcup bc \sqcup \underline{a}) & (a) \\
 & \vdash_M^* & (\sqcup\underline{\sqcup}bc \sqcup a) & (L_{\sqcup}^{2}) \\
 & \vdash_M & (\sqcup\underline{a}bc \sqcup a) & (a) \\
 & \vdash_M & (\sqcup a\underline{b}c \sqcup a) & (R)
\end{array}
$$

Example

What is the goal of the following Turing Machine?

$$\begin{tikzcd}
> L_{\sqcup} \arrow[r] & R \arrow[r, "a \neq \sqcup"] \arrow[d, "\sqcup"] & \sqcup R^2_{\sqcup} a L^2_{\sqcup} a \arrow[l, bend right] \\
& R_{\sqcup} &
\end{tikzcd}$$

$$\begin{array}{llll}
(\sqcup abc\underline{\sqcup}) & \vdash^*_M & (\underline{\sqcup}abc\sqcup) & (L_{\sqcup}) \\
& \vdash_M & (\sqcup\underline{a}bc\sqcup) & (R) \\
& \vdash_M & (\sqcup\underline{\sqcup}bc\sqcup) & (\sqcup) \\
& \vdash^*_M & (\sqcup \sqcup bc \sqcup \underline{\sqcup}) & (R^2_{\sqcup}) \\
& \vdash_M & (\sqcup \sqcup bc \sqcup \underline{a}) & (a) \\
& \vdash^*_M & (\sqcup\underline{\sqcup}bc \sqcup a) & (L^2_{\sqcup}) \\
& \vdash_M & (\sqcup\underline{a}bc \sqcup a) & (a) \\
& \vdash_M & (\sqcup a\underline{b}c \sqcup a) & (R)
\end{array}$$

Solution:

transforms $\sqcup w\underline{\sqcup}$ to $\sqcup w \sqcup w\underline{\sqcup}$

Behind the power...

Why do we need so formal descriptions?

- Precision avoids ambiguity
- The finest grain is required
- The hierarchical decomposition is useful

Generalize more the notation ...

High-level description

- give an algorithmic description of how the Turing Machine works in finite and discrete steps
- what is allowed?

Generalize more the notation ...

High-level description

- give an algorithmic description of how the Turing Machine works in finite and discrete steps
- what is allowed? **almost everything!**

Generalize more the notation ...

High-level description

- give an algorithmic description of how the Turing Machine works in finite and discrete steps
- what is allowed? **almost everything!**

Example

$M =$ "On input w:

1. scan the input from left to right to be sure that it is of the form $a^*b^*c^*$ and *reject* if not
2. find the leftmost a in the tape and if such an a does not exist, then
   - scan the input for a c and if such a c exists then *reject* else *accept*
3. replace a by $\hat{a}$
4. scan the input for the leftmost b and if such a b does not exist, then restore all b's (replace all $\hat{b}$ by b) and goto 2
5. replace b by $\hat{b}$
6. scan to the right for the first c and if such a c does not exist, then *reject*
7. replace c by $\hat{c}$ and goto 4"

Generalize more the notation ...

High-level description

- give an algorithmic description of how the Turing Machine works in finite and discrete steps
- what is allowed? **almost everything!**

Example

$L = \{a^i b^j c^k : i \times j = k\}$

$M =$ "On input w:

1. scan the input from left to right to be sure that it is of the form $a^* b^* c^*$ and *reject* if not
2. find the leftmost a in the tape and if such an a does not exist, then
   - scan the input for a c and if such a c exists then *reject* else *accept*
3. replace a by $\hat{a}$
4. scan the input for the leftmost b and if such a b does not exist, then restore all b's (replace all $\hat{b}$ by b) and goto 2
5. replace b by $\hat{b}$
6. scan to the right for the first c and if such a c does not exist, then *reject*
7. replace c by $\hat{c}$ and goto 4"

Definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ such that $H = \{y, n\}$.

Any halting configuration whose state component is y (for "*yes*") is called an **accepting configuration**, while a halting configuration whose state component is n (for "*no*") is called a **rejecting configuration**.

Definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ such that $H = \{y, n\}$.

Any halting configuration whose state component is y (for "*yes*") is called an **accepting configuration**, while a halting configuration whose state component is n (for "*no*") is called a **rejecting configuration**.

We say that M **accepts** a string $w \in \Sigma^*$ if starting from an initial configuration yields an accepting configuration.
We say that M **rejects** a string $w \in \Sigma^*$ if starting from an initial configuration yields an rejecting configuration.

Definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ such that $H = \{y, n\}$.

Any halting configuration whose state component is y (for "*yes*") is called an **accepting configuration**, while a halting configuration whose state component is n (for "*no*") is called a **rejecting configuration**.

We say that M **accepts** a string $w \in \Sigma^*$ if starting from an initial configuration yields an accepting configuration.
We say that M **rejects** a string $w \in \Sigma^*$ if starting from an initial configuration yields an rejecting configuration.

We say that M **decides** a language $L \subseteq \Sigma^*$ if for any string $w \in \Sigma^*$: if $w \in L$ then M accepts w; and if $w \notin L$ then M rejects w.

Definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ such that $H = \{y, n\}$.

Any halting configuration whose state component is y (for "*yes*") is called an **accepting configuration**, while a halting configuration whose state component is n (for "*no*") is called a **rejecting configuration**.

We say that M **accepts** a string $w \in \Sigma^*$ if starting from an initial configuration yields an accepting configuration.
We say that M **rejects** a string $w \in \Sigma^*$ if starting from an initial configuration yields an rejecting configuration.

We say that M **decides** a language $L \subseteq \Sigma^*$ if for any string $w \in \Sigma^*$: if $w \in L$ then M accepts w; and if $w \notin L$ then M rejects w.

We say that M **recognizes** (or **semidecides**) a language $L \subseteq \Sigma^*$ if for any string $w \in \Sigma^*$: $w \in L$ if and only if M accepts w.

Definitions

A language L is called **decidable** (or **Turing-decidable** or **recursive**) if there is a Turing Machine that decides it.

Definitions

A language L is called **decidable** (or **Turing-decidable** or **recursive**) if there is a Turing Machine that decides it.

A language L is called **Turing-recognizable** (or **recursively enumerable**) if there is a Turing Machine that recognizes it.

Basic theorems

Theorem

If a language L is decidable, then it is Turing-recognizable.

Basic theorems

Theorem

If a language L is decidable, then it is Turing-recognizable.

Theorem

If a language L is decidable, then its complement $\bar{L}$ is also.

Proof.

$$\delta'(q,a) = \begin{cases} n & \text{if } \delta(q,a) = y \\ y & \text{if } \delta(q,a) = n \\ \delta(q,a) & \text{otherwise} \end{cases}$$

$\square$

More definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, \{h\})$ and a string $w \in \Sigma^*$. Suppose that M halts on input w and for some $y \in \Sigma^*$ we have

$$(s, \underline{\sqcup} w) \vdash_M^* (h, \underline{\sqcup} y)$$

Then, y is the **output** of M on input w and is denoted by $M(w)$.

More definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, \{h\})$ and a string $w \in \Sigma^*$. Suppose that M halts on input w and for some $y \in \Sigma^*$ we have

$$(s, \underline{\sqcup} w) \vdash_M^* (h, \underline{\sqcup} y)$$

Then, y is the **output** of M on input w and is denoted by $M(w)$.

Consider a function $f : \Sigma^* \to \Sigma^*$. We say that M **computes** the function f if $M(w) = f(w)$ for all $w \in \Sigma^*$.

More definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, \{h\})$ and a string $w \in \Sigma^*$. Suppose that M halts on input w and for some $y \in \Sigma^*$ we have

$$(s, \underline{\sqcup}w) \vdash_M^* (h, \underline{\sqcup}y)$$

Then, y is the **output** of M on input w and is denoted by $M(w)$.

Consider a function $f : \Sigma^* \to \Sigma^*$. We say that M **computes** the function f if $M(w) = f(w)$ for all $w \in \Sigma^*$.

A function f is called **decidable** (or **recursive**) if there is a Turing Machine that computes it.

Extension of the Turing Machine

The natural extension:

- write in the tape and move left or right at the same time
- modify the definition of the transition function

  initial: from $(K \setminus H) \times \Gamma$ to $K \times (\Gamma \cup \{\leftarrow, \rightarrow\})$

  extended: from $(K \setminus H) \times \Gamma$ to $K \times \Gamma \times \{\leftarrow, \rightarrow\}$

Extension of the Turing Machine

The natural extension:

- write in the tape and move left or right at the same time
- modify the definition of the transition function

  initial: from $(K \setminus H) \times \Gamma$ to $K \times (\Gamma \cup \{\leftarrow, \rightarrow\})$

  extended: from $(K \setminus H) \times \Gamma$ to $K \times \Gamma \times \{\leftarrow, \rightarrow\}$

- if the extended Turing Machine halts on input w after t steps, then the initial Turing Machine halts on input w after at most $2t$ steps

Discussion

- We can even combine some extensions and still not get a *stronger* (more powerful) model

Discussion

- We can even combine some extensions and still not get a *stronger* (more powerful) model
- Observation: a computation in the prototype Turing Machine needs a number of steps which is bounded by a polynomial of the size of the input and of the number of steps in any of the extended model