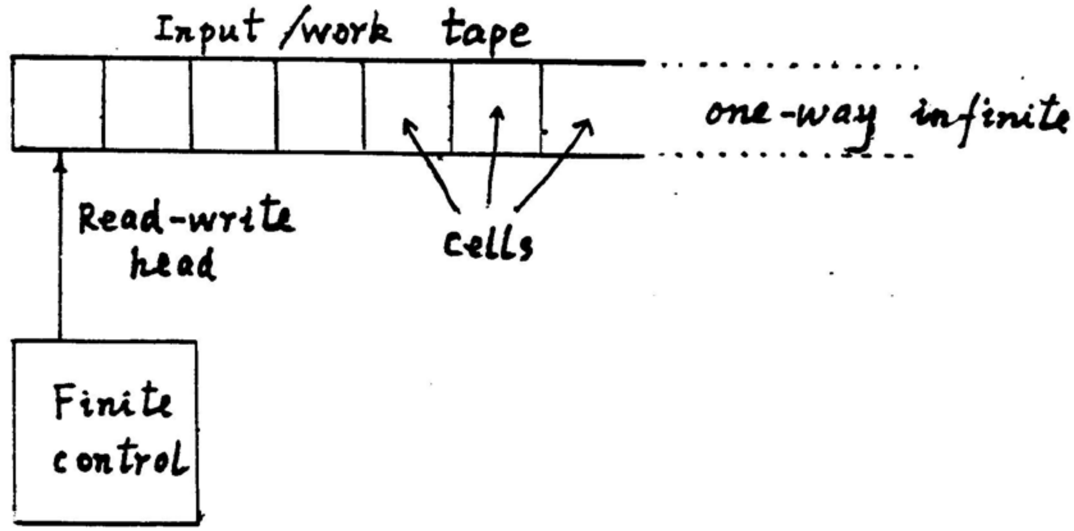
V. TURING MACHINES



Turing machines have more features than FA and PDA.

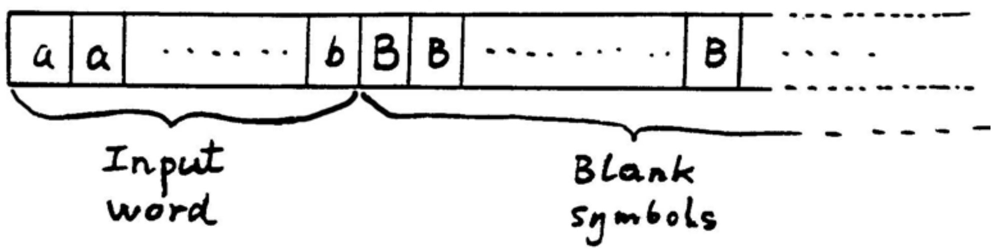
(1) The read-write head can move in either direction.

(2) It can write on the tape.

Turing machines are studied as a theoretical model of computers.

Some assumptions for TM's:

(1) At beginning, the input string (symbols) is placed at the left end of the input tape and followed by infinitely many blank symbols denoted by B's.



(2) There is only one final state denoted by "f".

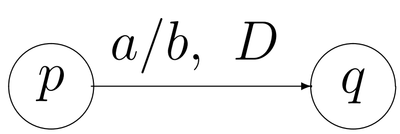
(3) A TM stops when it enters the final state "f".

<u>**Definition**</u> **A deterministic Turing Machine (DTM) is specified by a sextuple**
$(Q, \Sigma, \Gamma, \delta, s, f)$ **where**

- Q**: is a finite set of** <u>**states**</u>**;**
- Σ**: is an alphabet of** <u>**input symbols**</u>**;**
- Γ**: is an alphabet of** <u>**tape symbols**</u>**,**
  <u>$\Sigma \cup \{B\} \subseteq \Gamma$</u>
- $\delta :\ Q \times \Gamma \to Q \times \Gamma \times \{L, R, \varepsilon\}$ **is a** <u>**transition function**</u>**;**
- $s \in Q$ **is a** <u>**start state**</u>**;**
- $f \in Q$ **is a** <u>**final state**</u>**;**

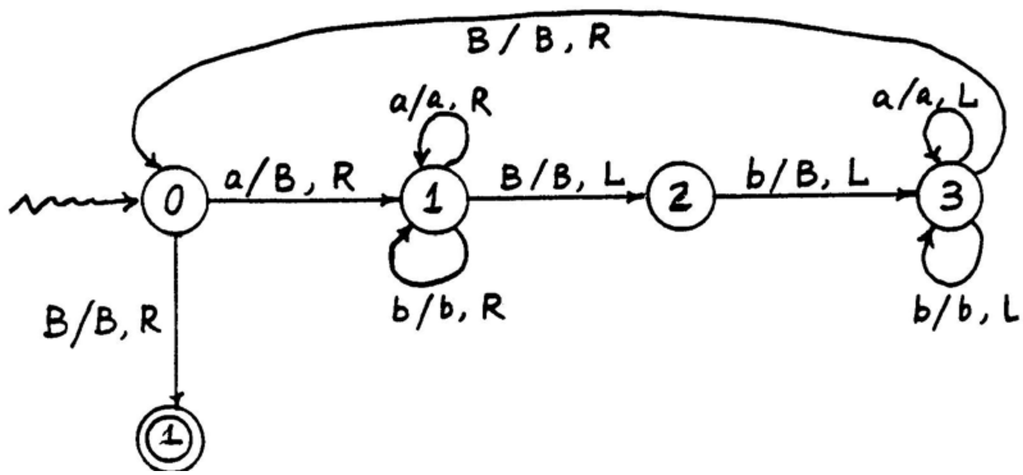
<u>**State diagram**</u>

$\delta(p, a) = (q, b, D)$ **is depicted graphically:**



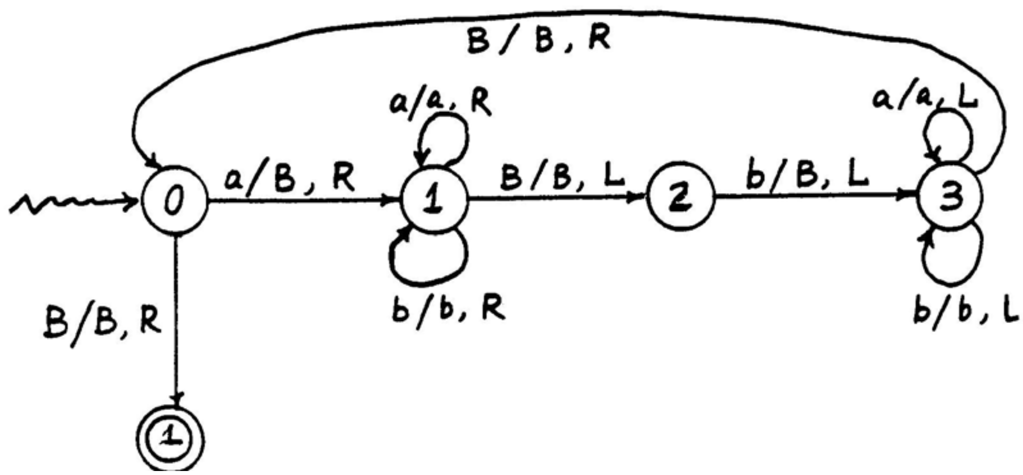
<u>Example</u> A DTM that accepts

$L = \{a^i b^i \mid i \geq 0\}$



Example A DTM that accepts

$L = \{a^i b^i \mid i \geq 0\}$



Configuration

A configuration is a word in

$$\Gamma^* Q \Gamma^*.$$

Strictly speaking, a configuration is a word in $\Gamma^* Q \Gamma^* (\Gamma - \{B\}) \cup \Gamma^* Q$

(Note: $Q \cap \Gamma = \emptyset$**)**

One move of a DTM

$$g_1 p h_1 \vdash g_2 q h_2 \qquad \textbf{if}$$

(i) either $h_1 = A h_1'$**, for some** A **in** Γ**,** h_1' **in** Γ^* **or** $h_1 = \varepsilon$**, then** $A = B$ **and** $h_1' = \varepsilon$**;**

(ii) $\delta(p, A)$ **is defined and** $p \neq f$**;**

(iii) $\delta(p, A) = (q, A', D)$

- **(a)** $D = L$**,** $g_1 = g_1' C$ **for some** $C \in \Gamma$**, and then** $h_2 = C A' h_1'$ **(if** $g_1 = \varepsilon$**, then** M **halts)**
- **(b)** $D = R$**,** $g_1 A' = g_2$ **and** $h_2 = h_1'$
- **(c)** $D = \varepsilon$**,** $g_2 = g_1$ **and** $h_2 = A' h_1'$ **(if** $A' = B,\ h_1' = \varepsilon,$ **then** $h_2 = \varepsilon$**)**

$\vdash^i, \ \vdash^+, \ \vdash^*$ **are defined as before.**

Language acceptance

$L(M) = \{x \mid sx \vdash^* yfz,$ **for some** $y, z \in \Gamma^*\}$
$\mathcal{L}_{DTM} = \{L \mid L = L(M)$ **for some DTM** $M\}$.

A DTM can be used

(i) as a language acceptor;

(ii) to compute a function:

$$f_M : \Sigma^* \to (\Gamma - \{B\})^*$$

$$f_M(x) = y \textbf{ in } (\Gamma - \{B\})^* \textbf{ iff}$$

$$sx \vdash^* y_1 f y_2, \textbf{ where } y = y_1 y_2$$

(iii) as a decision maker.

<u>Example</u> A right shift machine

Initial state:

B**: write** B**, move –, goto** f**;**

a**: write** B**, move right, goto** A**;**

b**: write** B**, move right, goto** B**;**

A**-state:**

a**: write** a**, move right, goto** A**;**

b**: write** a**, move right, goto** B**;**

B**: write** a**, move right, goto** f**;**

B**-state:**

a**: write** b**, move right, goto** A**;**

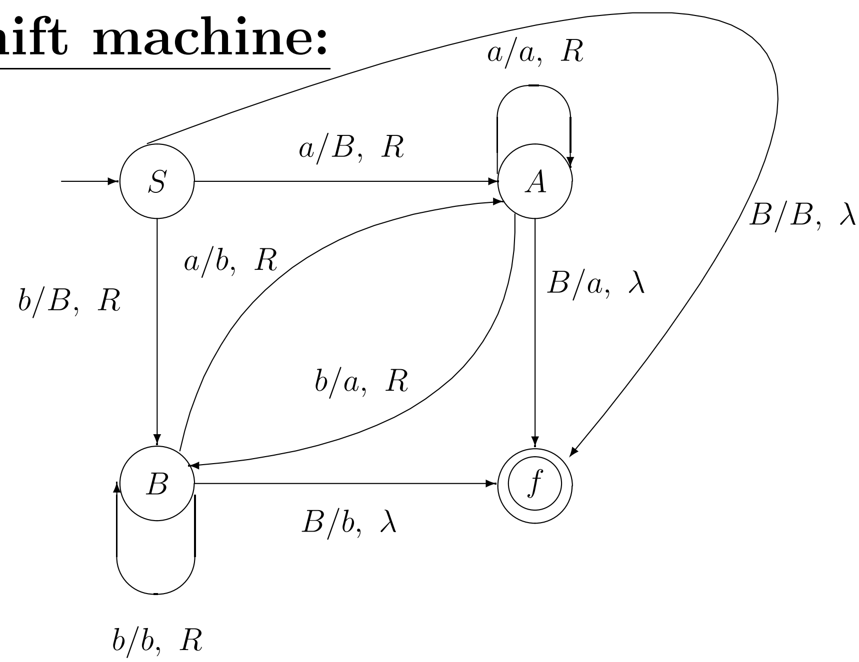
b**: write** b**, move right, goto** B**;**

B**: write** b**, move right, goto** f**;**

$$|a|a|b|b|b|a|a|a|B|B|$$

$$|B|a|a|b|b|b|a|a|a|B|$$

Right-shift machine:

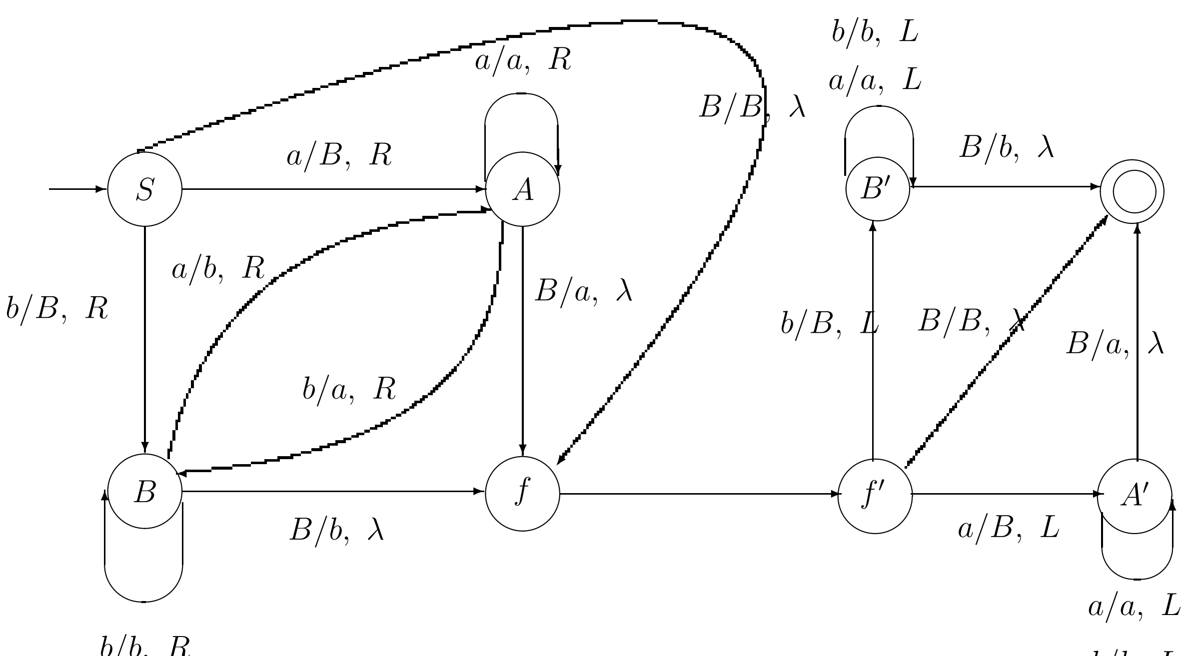


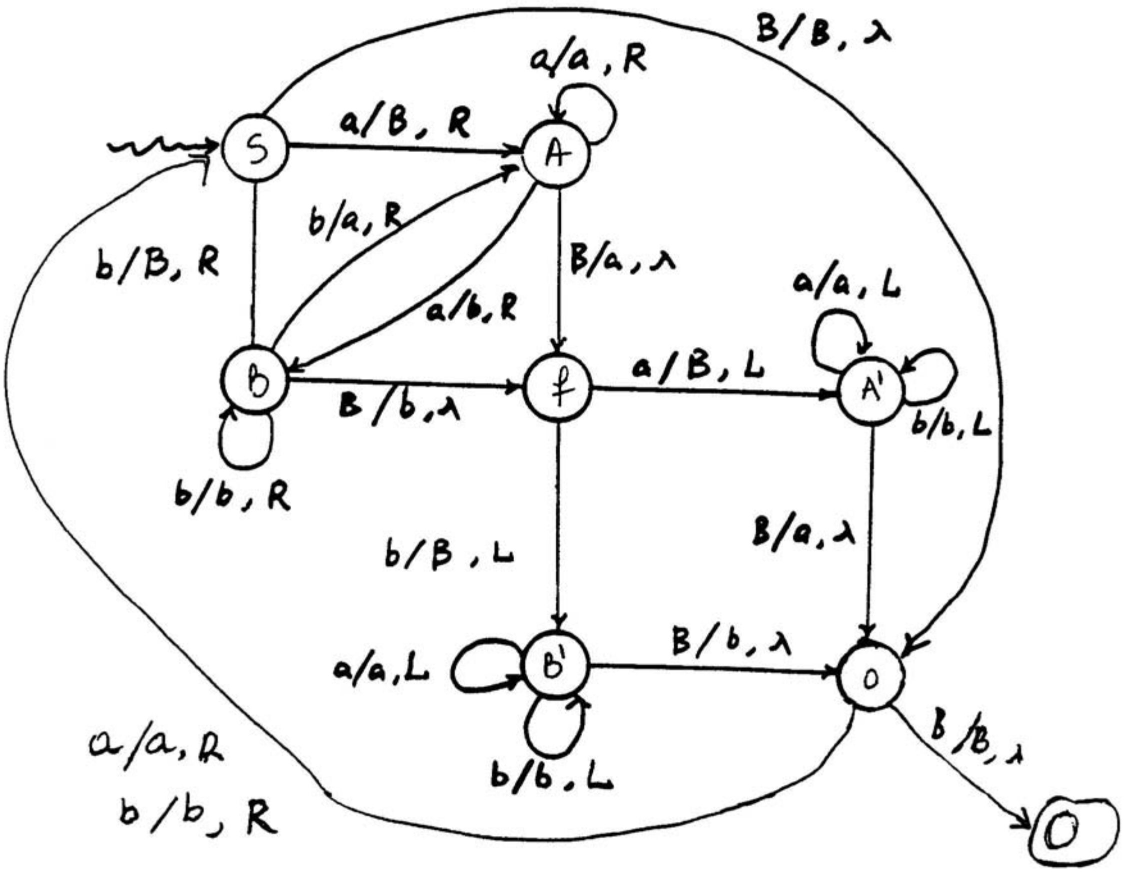
Example A cyclic right shift machine:

$|b|a|a|b|b|b|a|B|B|\ldots\ldots$

is transformed in:

$|a|b|a|a|b|b|b|B|B|\ldots\ldots$





Example A reversal machine

input: $aabba$

The Busy Beaver Problem

Consider a DTM with

— two-way infinite tape;

— a tape alphabet $\Gamma = \{1, B\}$;

— n states apart from f.

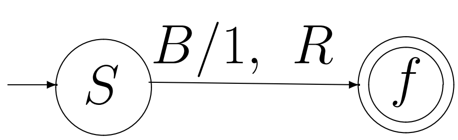
Question (by Tibor Rado)

> **How many 1's can there be on entering f, when given the empty word as input?**

(1's are like twigs. Beavers build busily with twigs.)

Define $\Sigma(n)$ **to be the maximum number of 1's that can be obtained by a DTM with n states.**

1-state DTM:



$$\Sigma(1) = 1$$

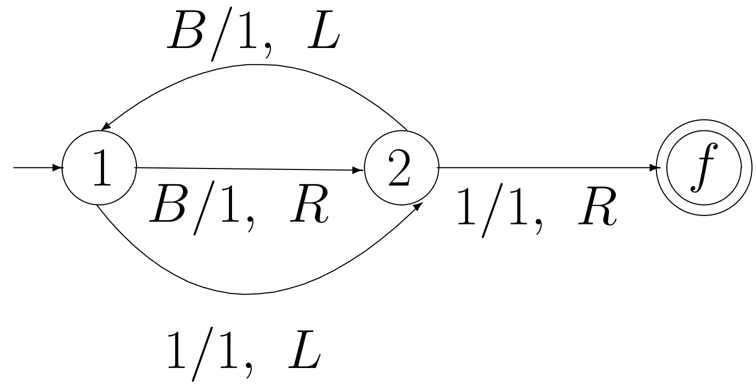
$$\Sigma(2) = 4$$

$$\Sigma(3) = 6$$

$$\Sigma(4) = 13$$

$$\Sigma(5) = ?, \qquad \geq 1915$$

2-state Machine:



$$\ldots\ldots|B|B|\underbrace{B}_{1}|B|\ldots\ldots$$

$$\ldots\ldots|B|B|1|\underbrace{B}_{2}|\ldots\ldots$$

$$\ldots\ldots|B|B|\underbrace{1}_{1}|1|\ldots\ldots$$

$$\ldots\ldots|B|\underbrace{B}_{2}|1|1|\ldots\ldots$$

$$\ldots\ldots|\underbrace{B}_{1}|1|1|1|\ldots\ldots$$

$$\ldots\ldots|1|\underbrace{1}_{2}|1|1|\ldots\ldots$$

$$\ldots\ldots|1|1|\underbrace{1}_{f}|1|\ldots\ldots$$

Definitions

Decision-making TM

A DTM $M = (Q, \Sigma, \Gamma, \delta, s, f)$ is said to be a decision making TM if y and n are in Γ and not in Σ, and for all $x \in \Sigma^*$, either $sx \vdash^* fy$ or $sx \vdash^* fn$.

Yes language of M

$$Y(M) = \{x \mid x \in \Sigma^* \textbf{ and } sx \vdash^* fy\}$$

No language of M

$$N(M) = \{x \mid x \in \Sigma^* \textbf{ and } sx \vdash^* fn\}$$

Decidability

Let $L \subseteq \Sigma^*$, $(B \notin \Sigma)$. L is decidable iff there is a decision-making TM M with $L = Y(M)$.

Recursive Languages

L **is recursive iff** L **is decidable.**

Notation

$\mathcal{L}_{REC}$ **denotes the family of recursive languages.**

Computability

Let $f : \Sigma^* \to \Delta^*$ **be a function, where** $B \notin \Sigma \cup \Delta$**.** f **is said to be** <u>**computable**</u> **iff there is a DTM** $M = (Q, \Sigma, \Gamma, \delta, s, f)$ **with** $\Delta \subseteq \Gamma$ **and for all** $x \in \Sigma^*$

$$\textbf{if } f(x) = y \textbf{ then}$$

$$sx \vdash^* y_1 f y_2 \quad \textbf{and} \quad y = y_1 y_2$$

for some $y_1, y_2 \in \Delta^*$**.**