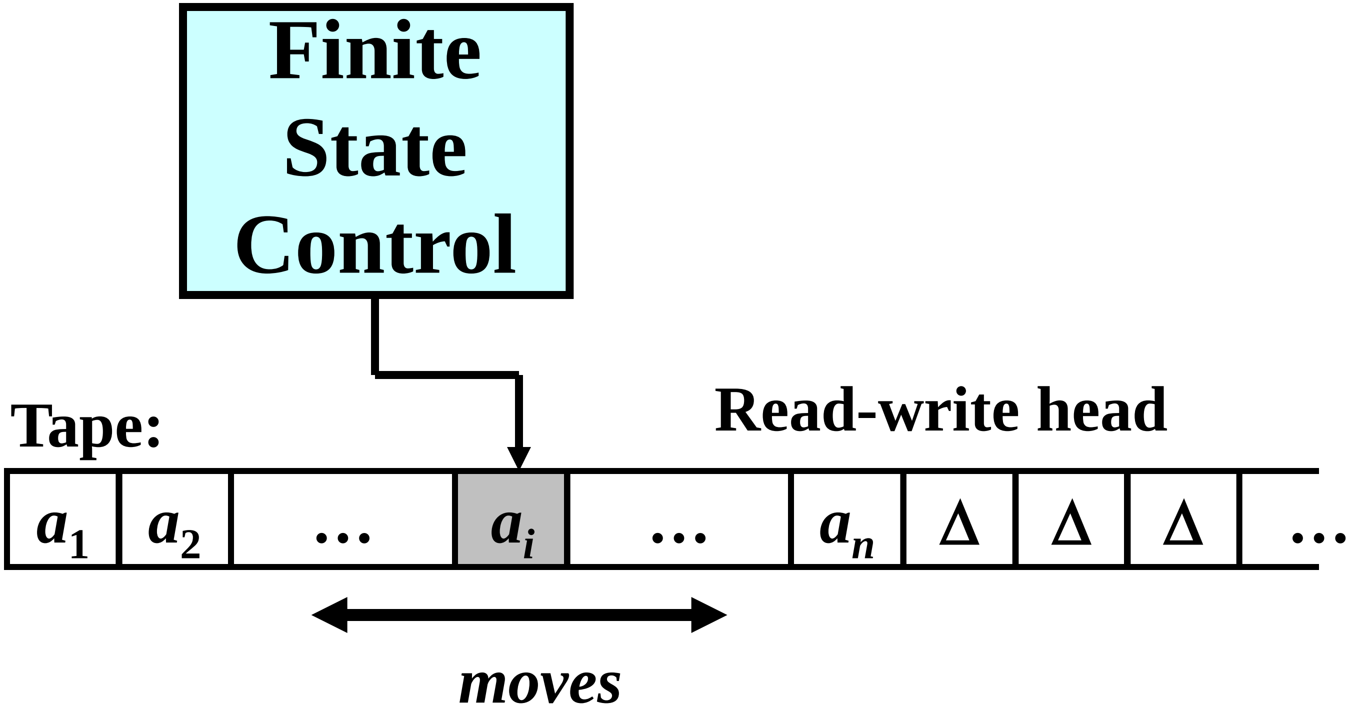
Turing Machines and General Grammars

Alan Turing (1912 – 1954)

Turing Machines (TM)

Gist: The most powerful computational model.



Note: Δ = blank

Turing Machines: Definition

Definition: *A Turing machine* (TM) is a 6-tuple $M = (Q, \Sigma, \Gamma, R, s, F)$, where

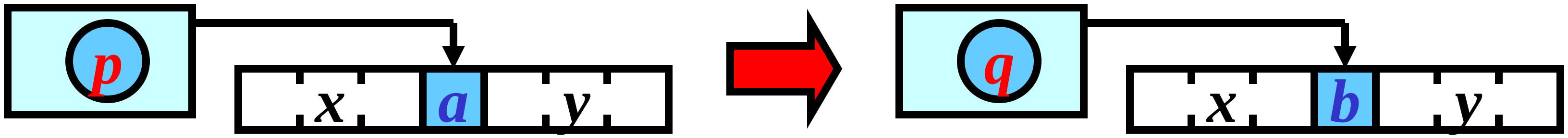
- Q is a *finite set of states*
- Σ is an *input alphabet*
- Γ is a *tape alphabet*; $\Delta \in \Gamma$; $\Sigma \subseteq \Gamma$
- R is a *finite set of rules* of the form: $pa \to qbt$, where $p, q \in Q$, $a, b \in \Gamma$, $t \in \{S, R, L\}$
- $s \in Q$ is the *start state*
- $F \subseteq Q$ is a set of *final states*

Mathematical note:

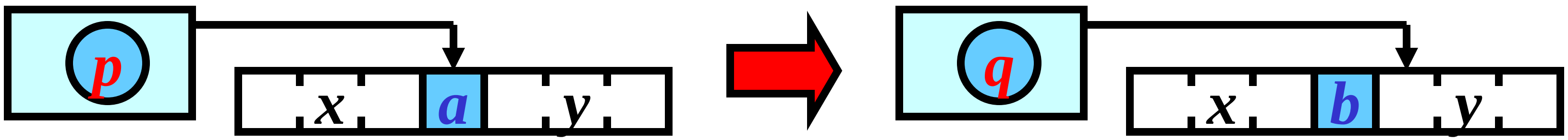
- Mathematically, R is a relation from $Q \times \Gamma$ to $Q \times \Gamma \times \{S, R, L\}$
- Instead of $(\boldsymbol{pa}, \boldsymbol{qbt})$, we write $\boldsymbol{pa \to qbt}$

Interpretation of Rules

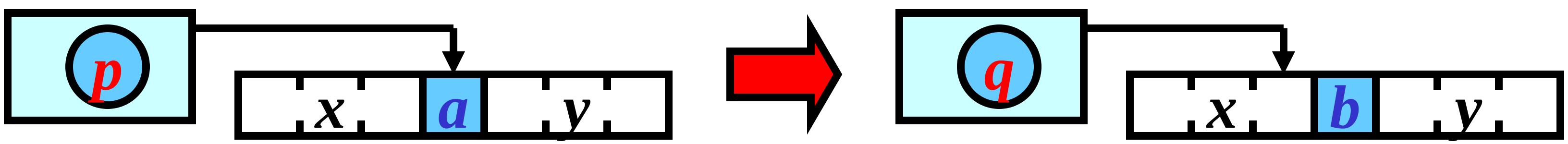
- $pa \rightarrow qbS$: If the current state and tape symbol are p and a, respectively, then replace a with b, change p to q, and keep the head Stationary.



- $pa \rightarrow qbR$: If the current state and tape symbol are p and a, respectively, then replace a with b, shift the head a square Right, and change p to q.

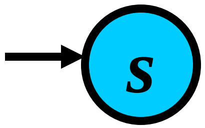


- $pa \rightarrow qbL$: If the current state and tape symbol are p and a, respectively, then replace a with b, shift the head a square Left, and change p to q.



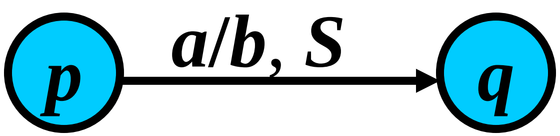
Graphical Representation

represents $q \in Q$

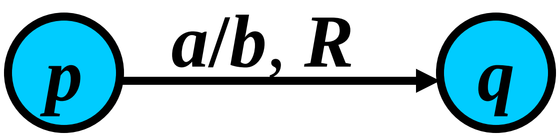


represents the initial state $s \in Q$

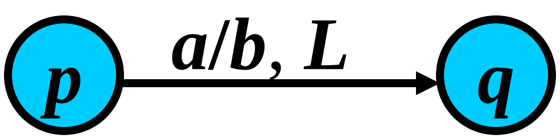
represents a final state $f \in F$



denotes $pa \to qbS \in R$



denotes $pa \to qbR \in R$



denotes $pa \to qbL \in R$

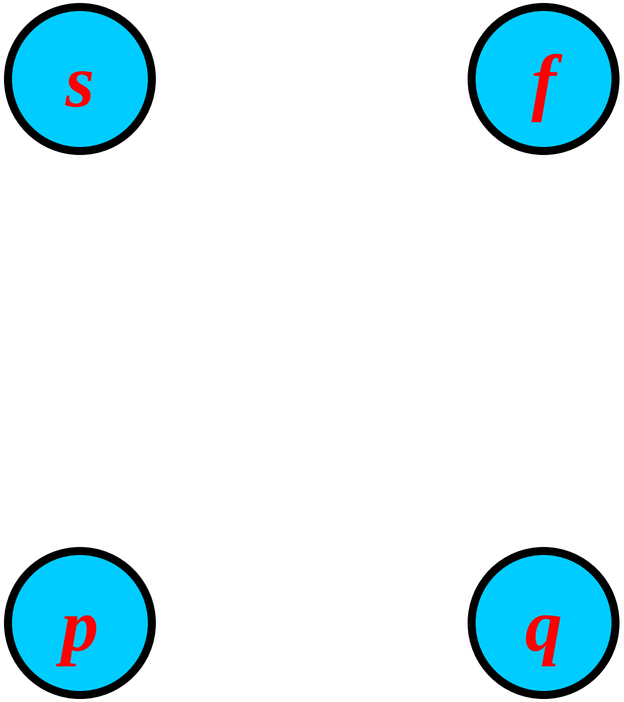
Turing Machine: Example 1/2

$M = (Q, \Sigma, \Gamma, R, s, F)$
where:

Turing Machine: Example 1/2

$M = (Q, \Sigma, \Gamma, R, s, F)$ where:

- $Q = \{s, p, q, f\}$;

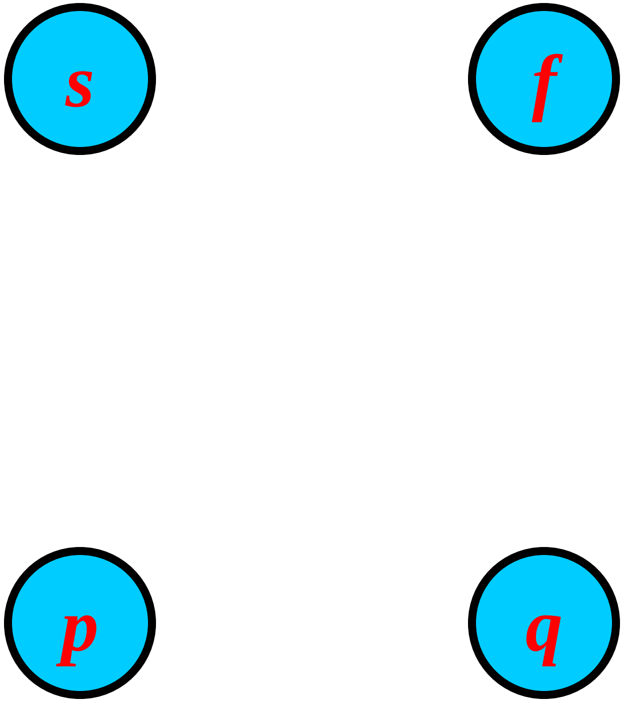


Turing Machine: Example 1/2

$M = (Q, \Sigma, \Gamma, R, s, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;

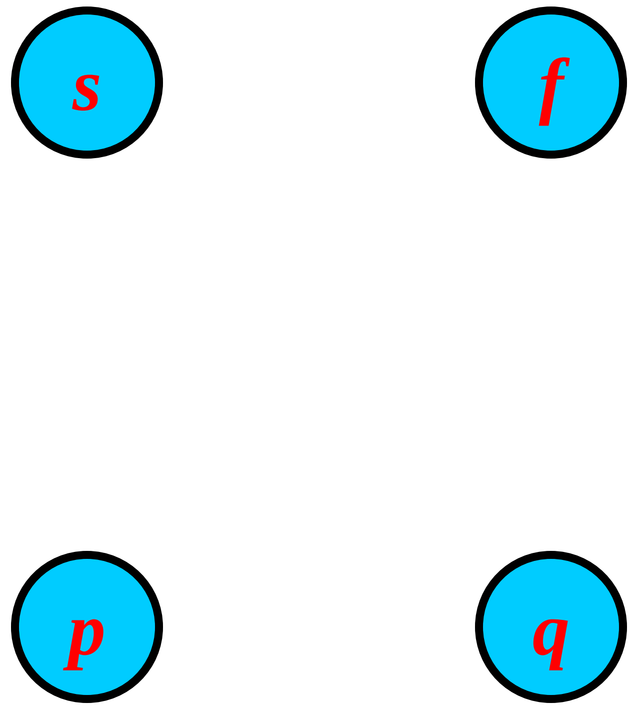


Turing Machine: Example 1/2

$M = (Q, \Sigma, \Gamma, R, s, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, b, \Delta\}$;

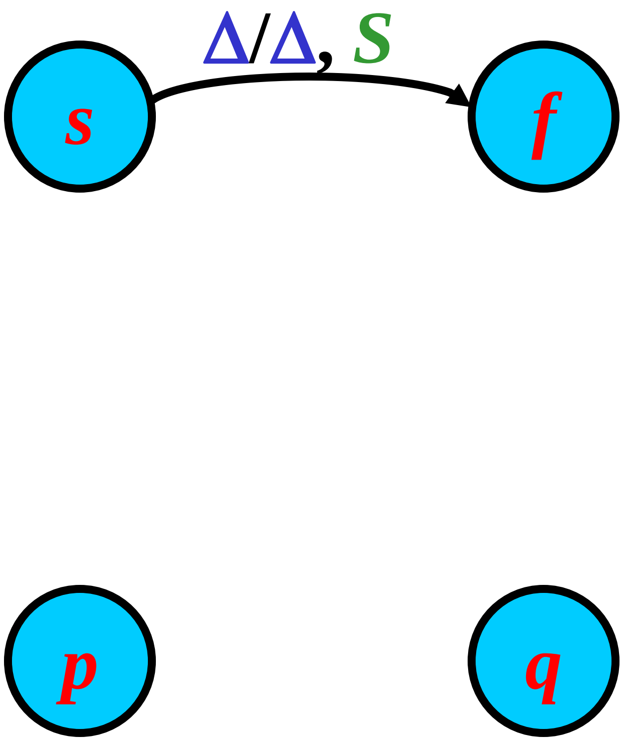


Turing Machine: Example 1/2

$M = (Q, \Sigma, \Gamma, R, s, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, b, \Delta\}$;
- $R = \{s\Delta \rightarrow f\Delta S,$



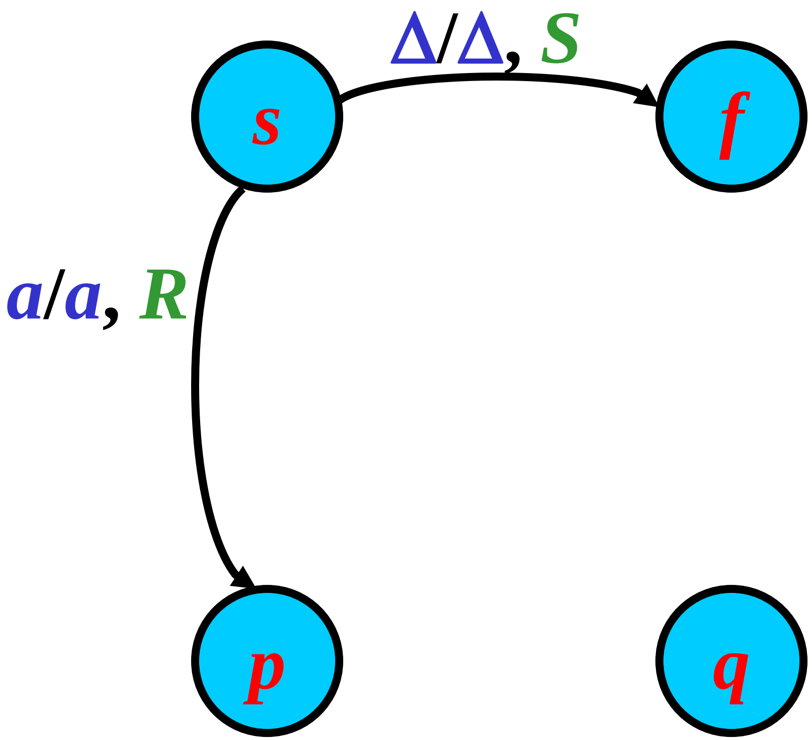
Turing Machine: Example 1/2

$M = (Q, \Sigma, \Gamma, R, s, F)$

where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, b, \Delta\}$;
- $R = \{s\Delta \to f\Delta S,$

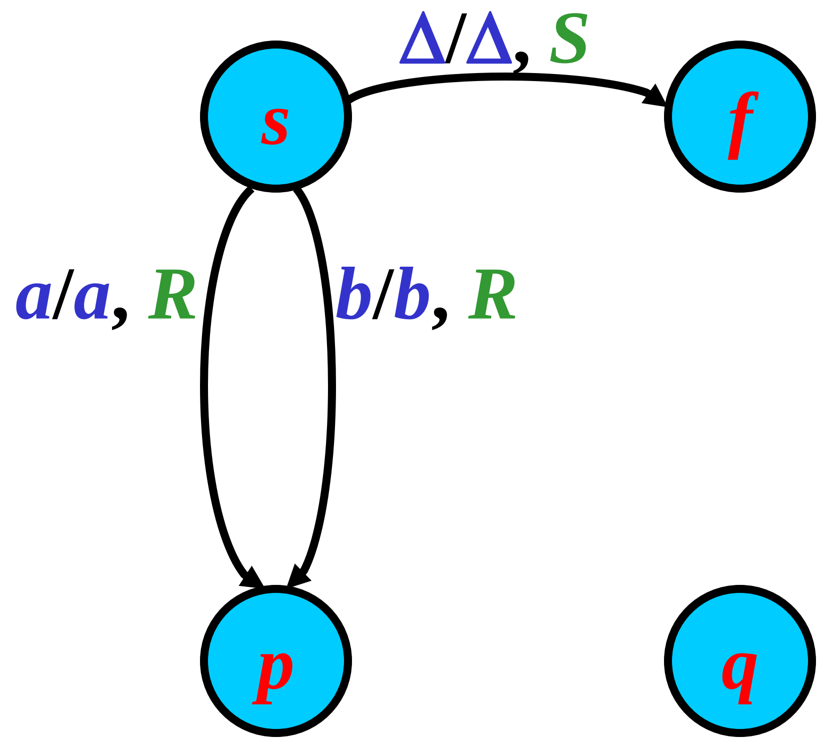
  $sa \to paR,$



Turing Machine: Example 1/2

$M = (Q, \Sigma, \Gamma, R, s, F)$
where:

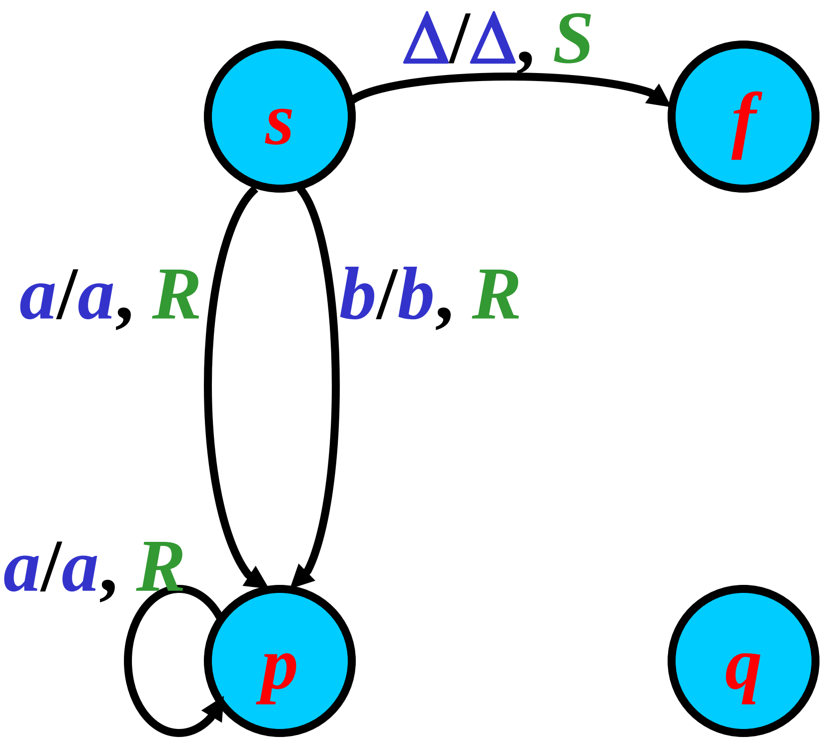
- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, b, \Delta\}$;
- $R = \{s\Delta \rightarrow f\Delta S,$
  $sa \rightarrow paR,$
  $sb \rightarrow pbR,$



Turing Machine: Example 1/2

$M = (Q, \Sigma, \Gamma, R, s, F)$
where:

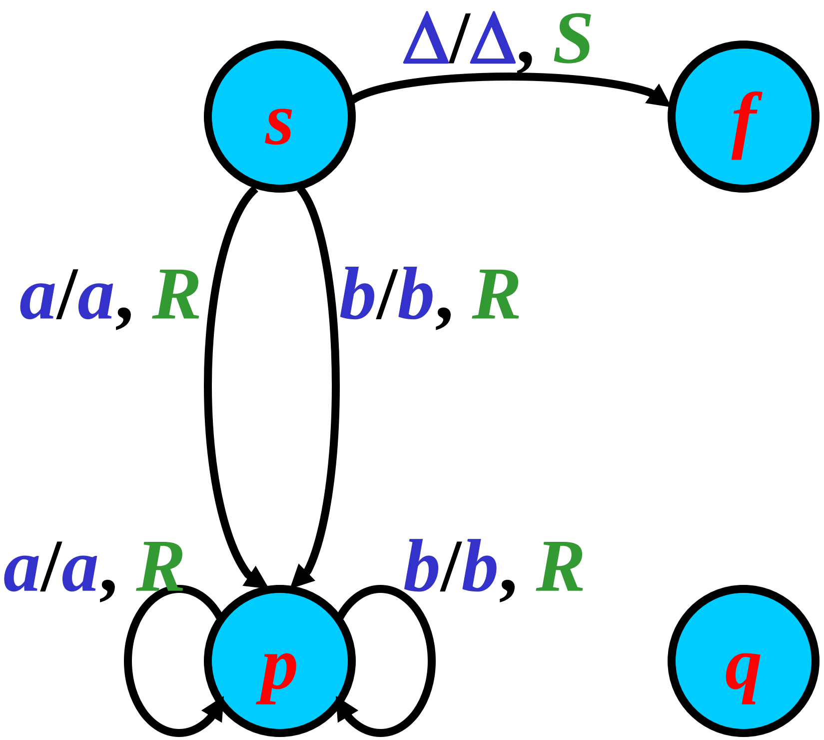
- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, b, \Delta\}$;
- $R = \{s\Delta \rightarrow f\Delta S,$
  $sa \rightarrow paR,$
  $sb \rightarrow pbR,$
  $pa \rightarrow paR,$



Turing Machine: Example 1/2

$M = (Q, \Sigma, \Gamma, R, s, F)$
where:

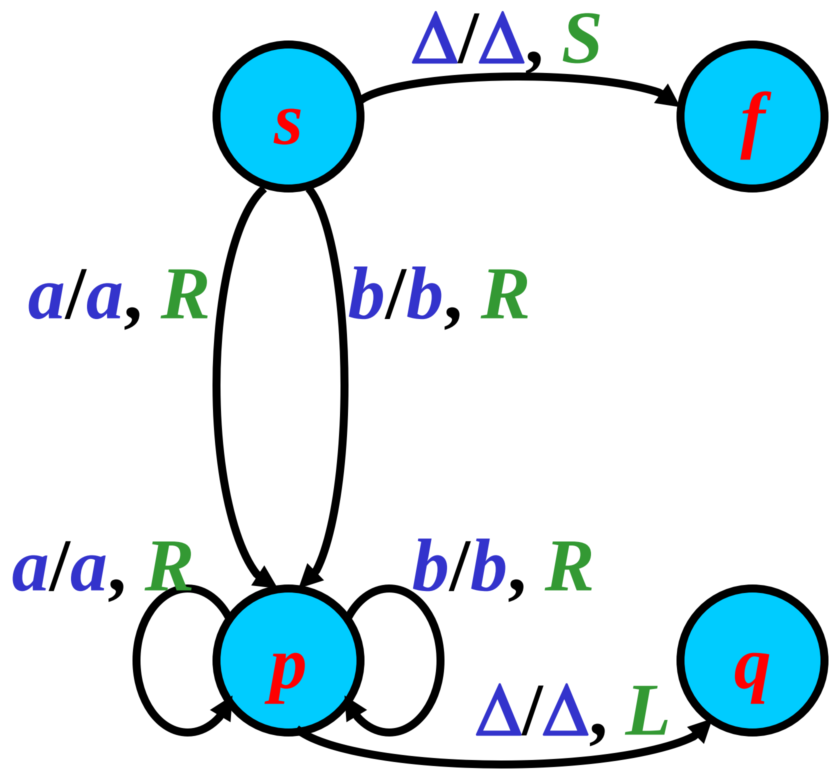
- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, b, \Delta\}$;
- $R = \{s\Delta \rightarrow f\Delta S,$
  $sa \rightarrow paR,$
  $sb \rightarrow pbR,$
  $pa \rightarrow paR,$
  $pb \rightarrow pbR,$



Turing Machine: Example 1/2

$M = (Q, \Sigma, \Gamma, R, s, F)$
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, b, \Delta\}$;
- $R = \{s\Delta \rightarrow f\Delta S,$
  $sa \rightarrow paR,$
  $sb \rightarrow pbR,$
  $pa \rightarrow paR,$
  $pb \rightarrow pbR,$
  $p\Delta \rightarrow q\Delta L,$

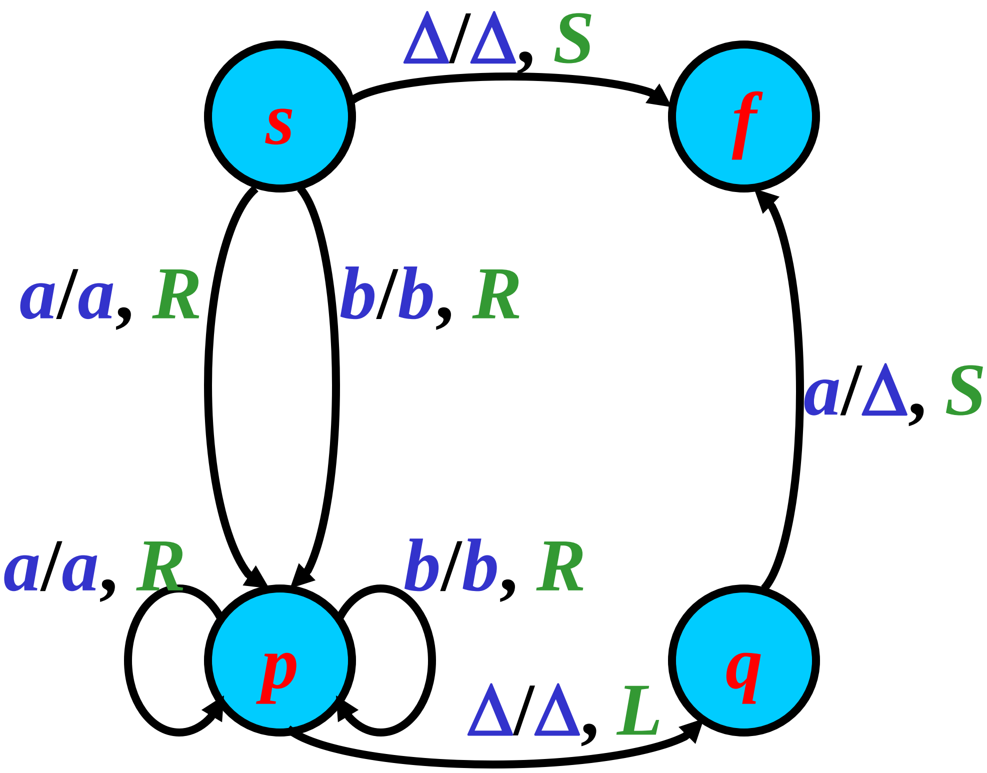


Turing Machine: Example 1/2

$M = (Q, \Sigma, \Gamma, R, s, F)$

where:

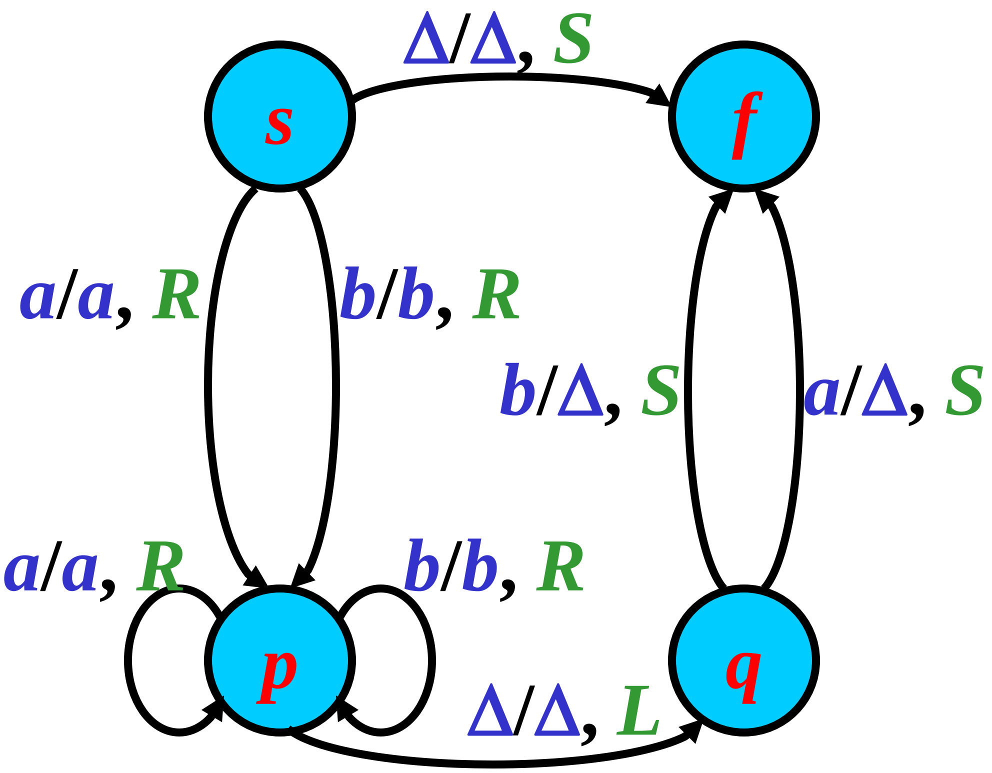
- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, b, \Delta\}$;
- $R = \{s\Delta \to f\Delta S,$
  $sa \to paR,$
  $sb \to pbR,$
  $pa \to paR,$
  $pb \to pbR,$
  $p\Delta \to q\Delta L,$
  $qa \to f\Delta S,$



Turing Machine: Example 1/2

$M = (Q, \Sigma, \Gamma, R, s, F)$
where:

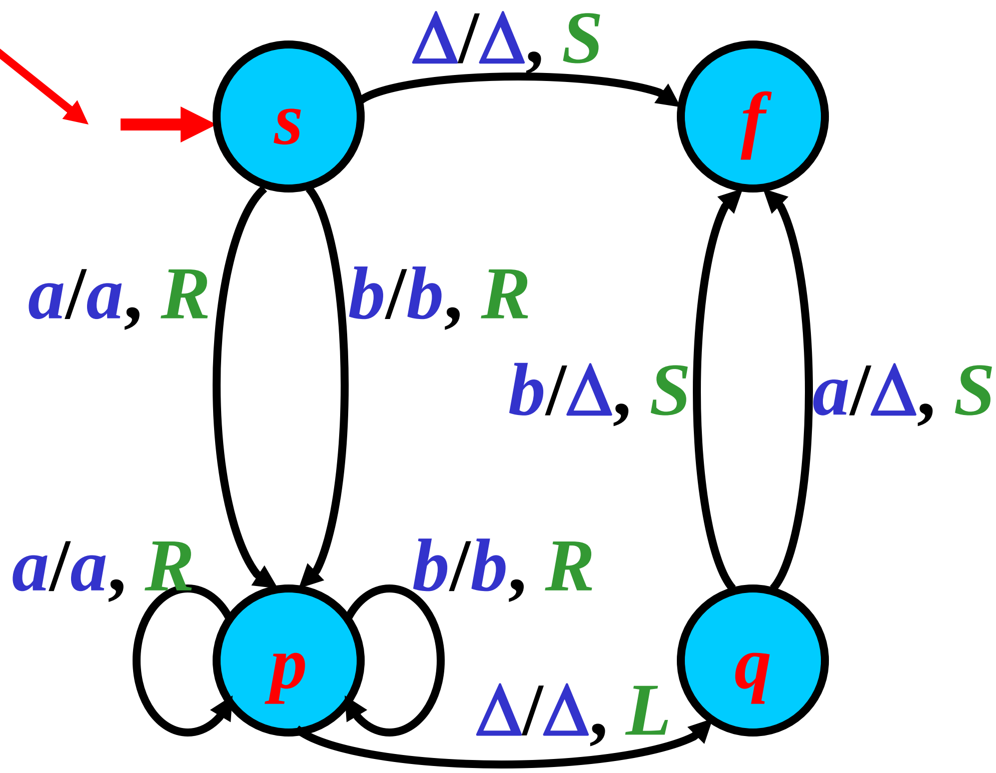
- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, b, \Delta\}$;
- $R = \{s\Delta \rightarrow f\Delta S,$
  $sa \rightarrow paR,$
  $sb \rightarrow pbR,$
  $pa \rightarrow paR,$
  $pb \rightarrow pbR,$
  $p\Delta \rightarrow q\Delta L,$
  $qa \rightarrow f\Delta S,$
  $qb \rightarrow f\Delta S\}$



Turing Machine: Example 1/2

$M = (Q, \Sigma, \Gamma, R, s, F)$
where:

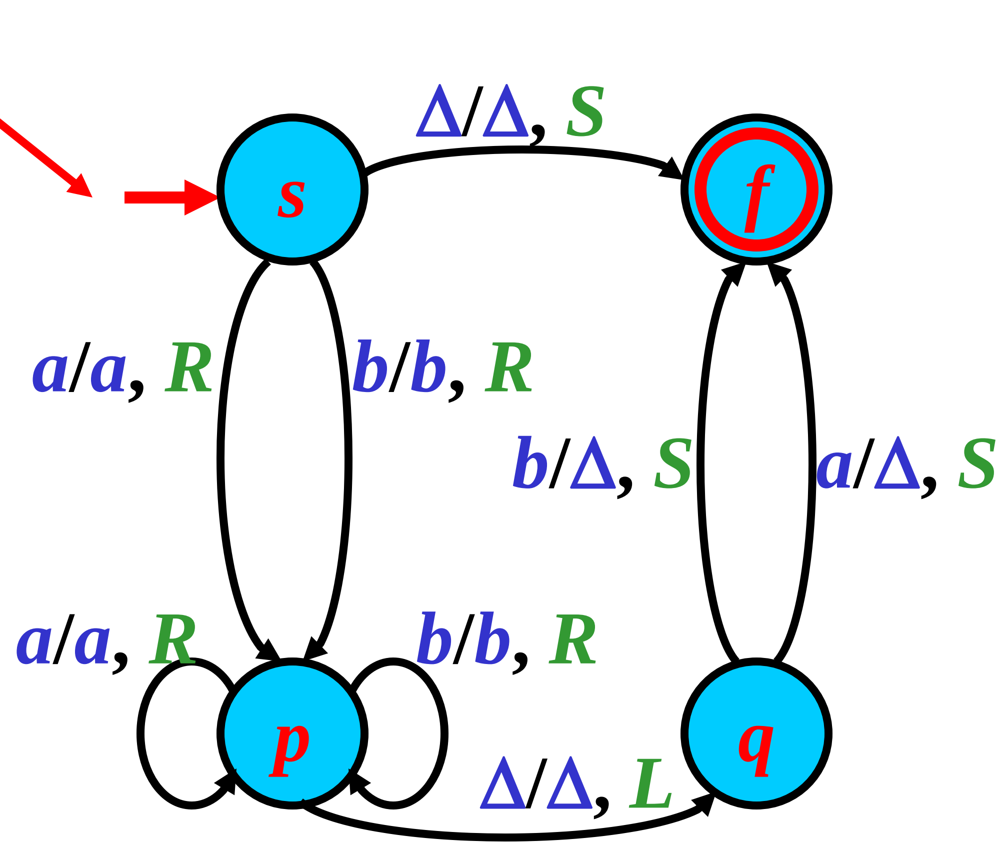
- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, b, \Delta\}$;
- $R = \{s\Delta \rightarrow f\Delta S,$
  $sa \rightarrow paR,$
  $sb \rightarrow pbR,$
  $pa \rightarrow paR,$
  $pb \rightarrow pbR,$
  $p\Delta \rightarrow q\Delta L,$
  $qa \rightarrow f\Delta S,$
  $qb \rightarrow f\Delta S\}$



Turing Machine: Example 1/2

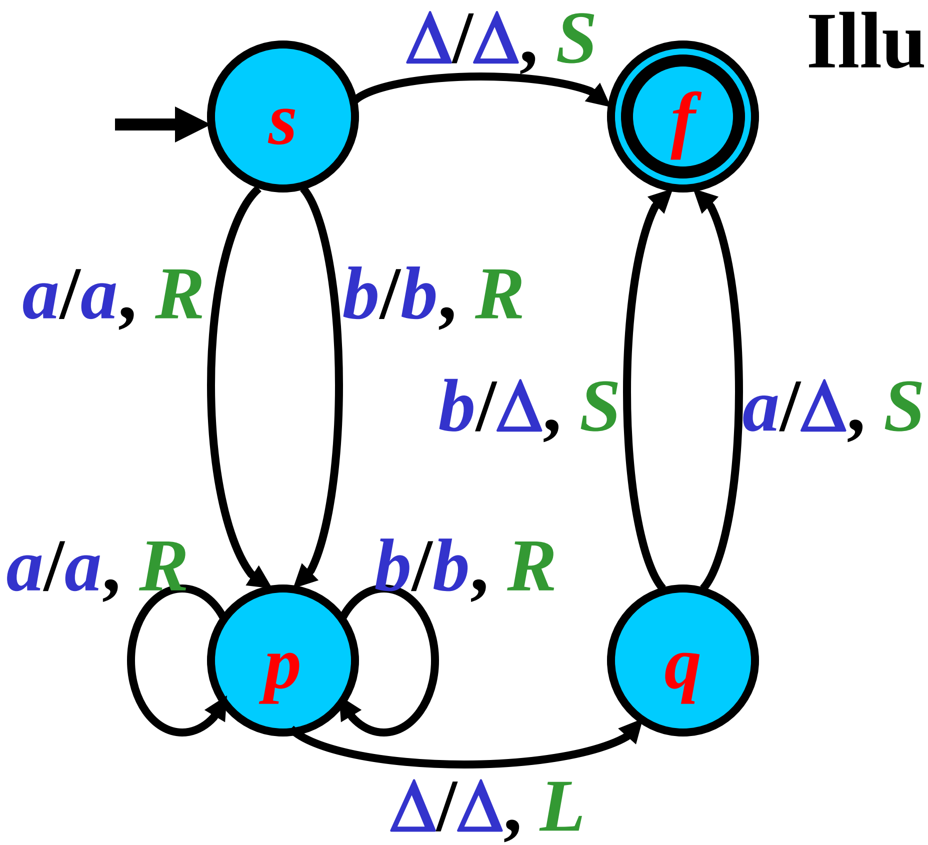
$M = (Q, \Sigma, \Gamma, R, s, F)$
where:

- $Q = \{s, p, q, f\}$;
- $\Sigma = \{a, b\}$;
- $\Gamma = \{a, b, \Delta\}$;
- $R = \{s\Delta \rightarrow f\Delta S,$
  $sa \rightarrow paR,$
  $sb \rightarrow pbR,$
  $pa \rightarrow paR,$
  $pb \rightarrow pbR,$
  $p\Delta \rightarrow q\Delta L,$
  $qa \rightarrow f\Delta S,$
  $qb \rightarrow f\Delta S\}$
- $F = \{f\}$



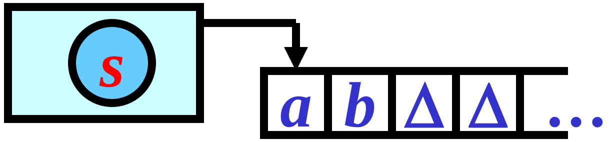
Turing Machine: Example 2/2

TM ***M***:



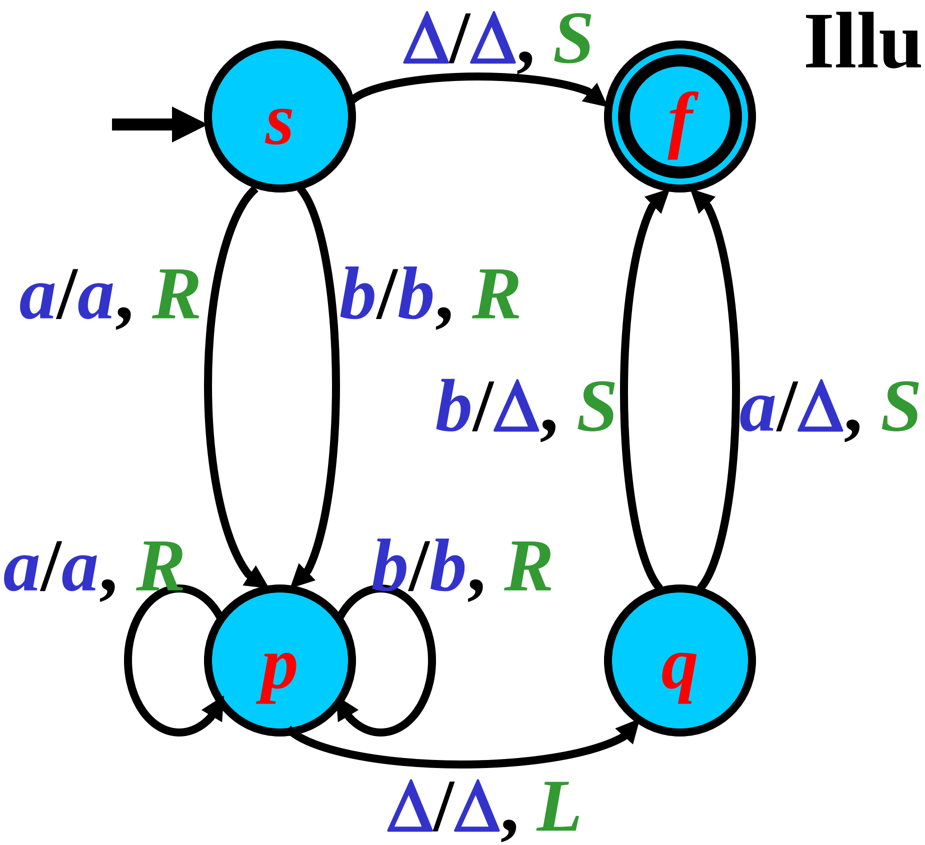
Note: *M* deletes a symbol before the first occurrence of Δ:

Illustration:



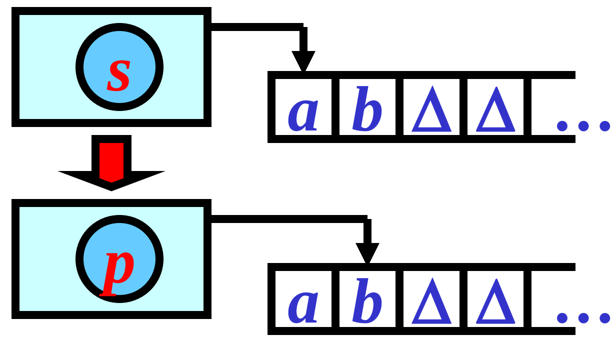
Turing Machine: Example 2/2

TM ***M***:



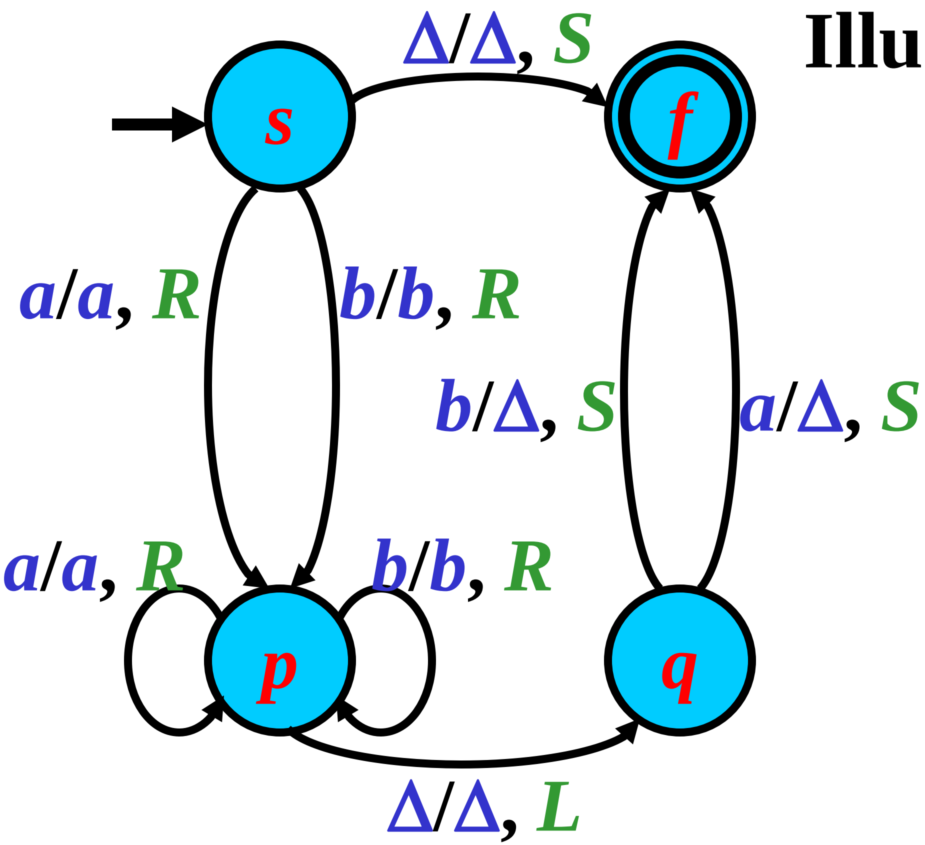
Note: *M* deletes a symbol before the first occurrence of Δ:

Illustration:



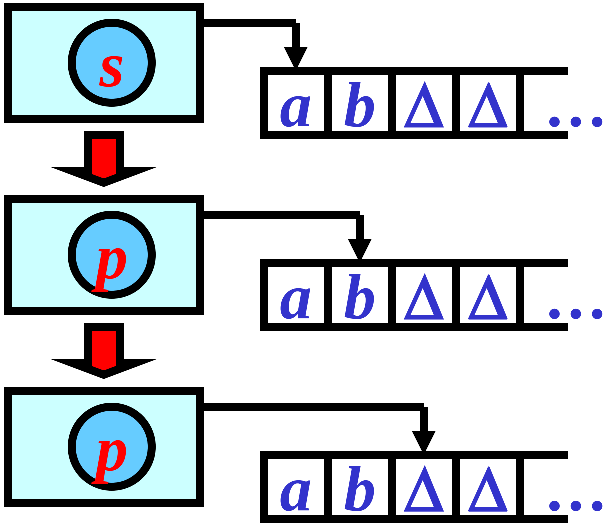
Turing Machine: Example 2/2

TM ***M***:



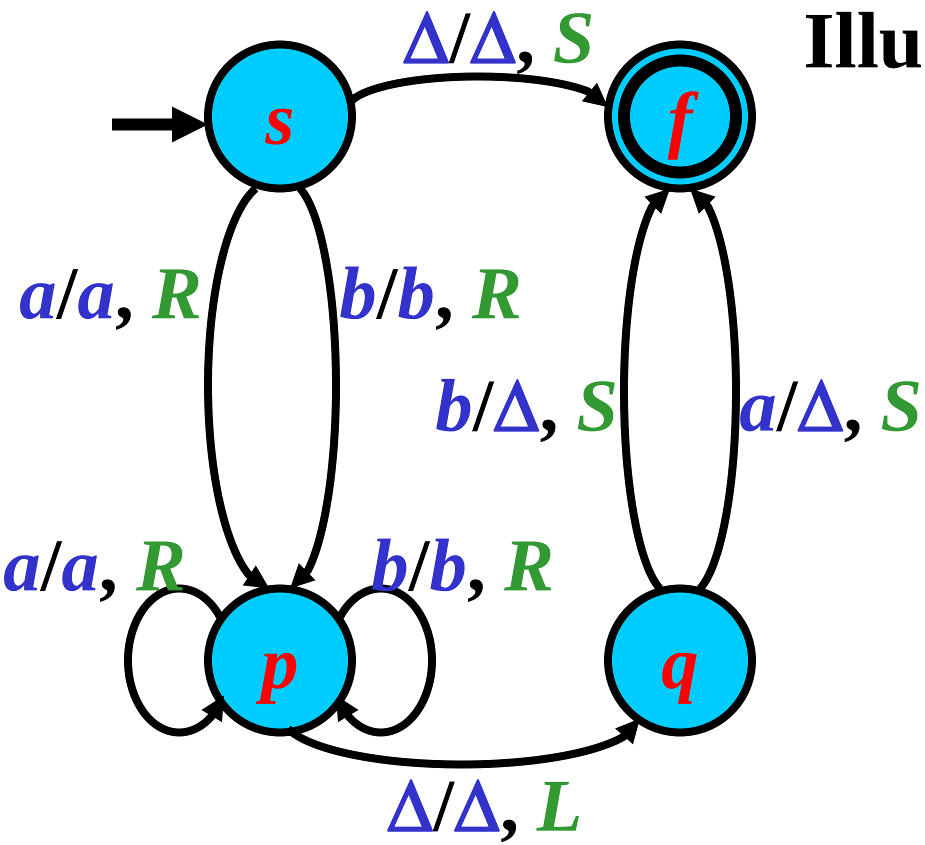
Note: *M* deletes a symbol before the first occurrence of Δ:

Illustration:



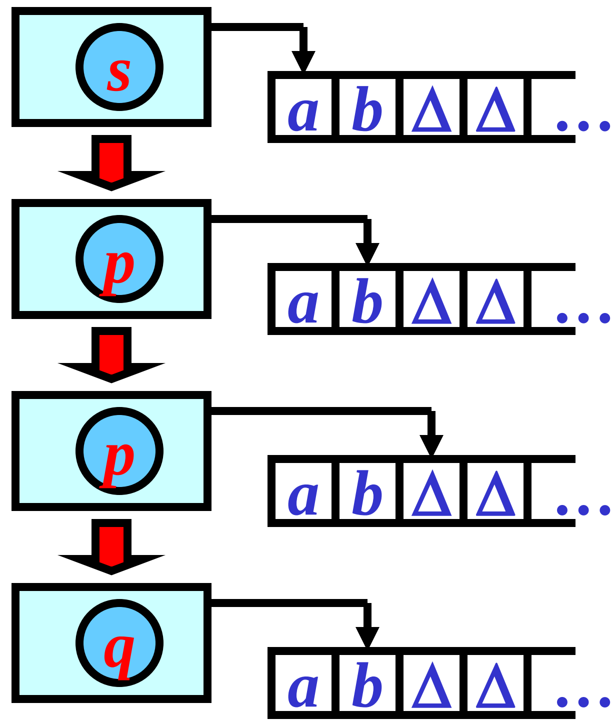
Turing Machine: Example 2/2

TM ***M***:



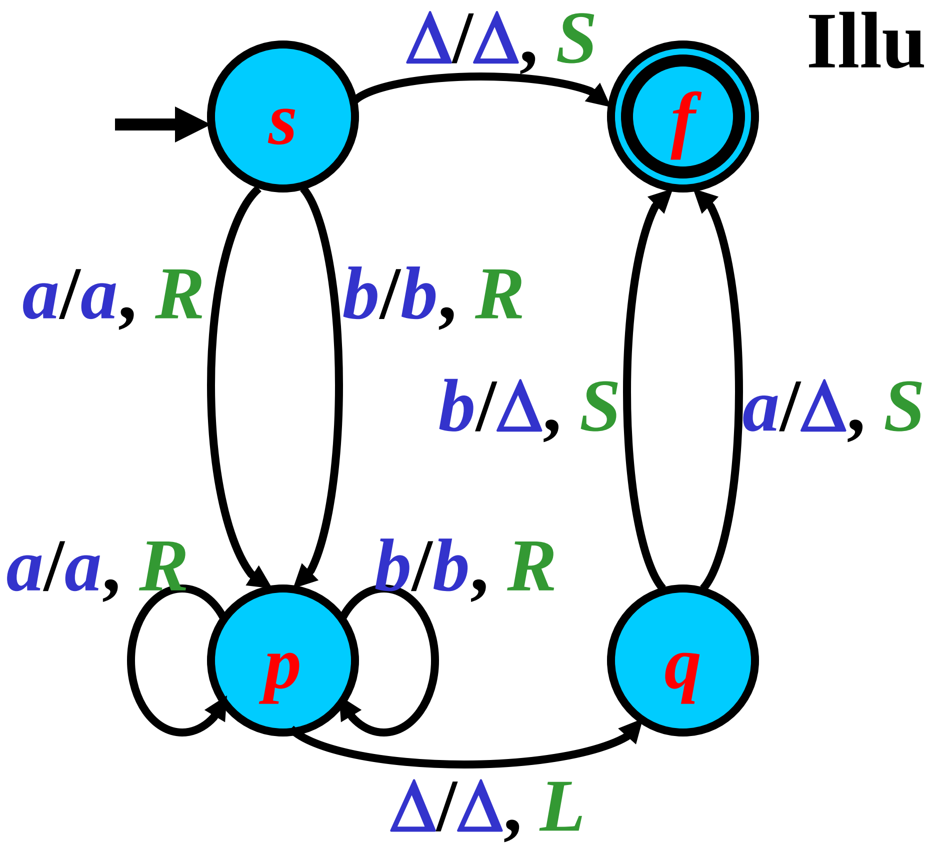
Note: *M* deletes a symbol before the first occurrence of Δ:

Illustration:



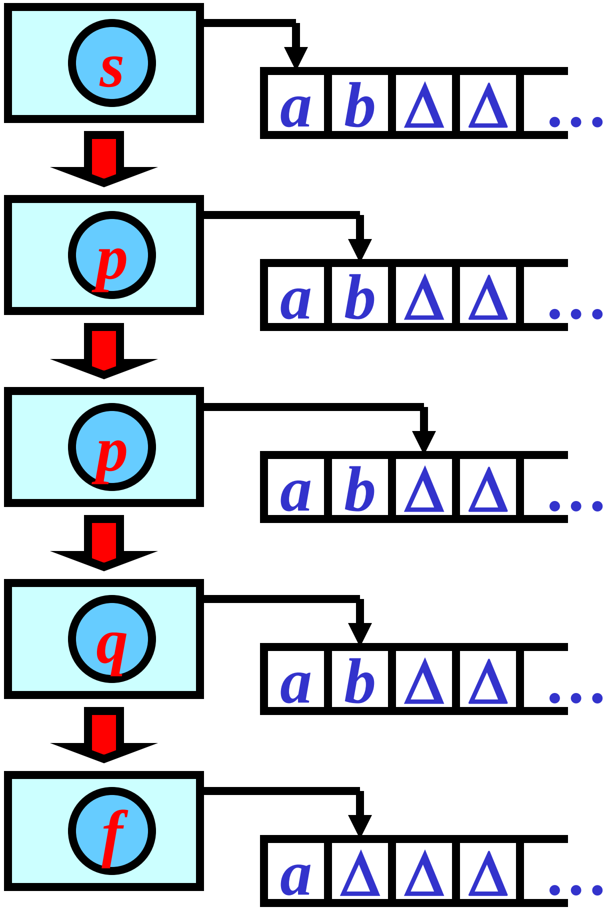
Turing Machine: Example 2/2

TM ***M***:



Note: *M* deletes a symbol before the first occurrence of Δ:

Illustration:

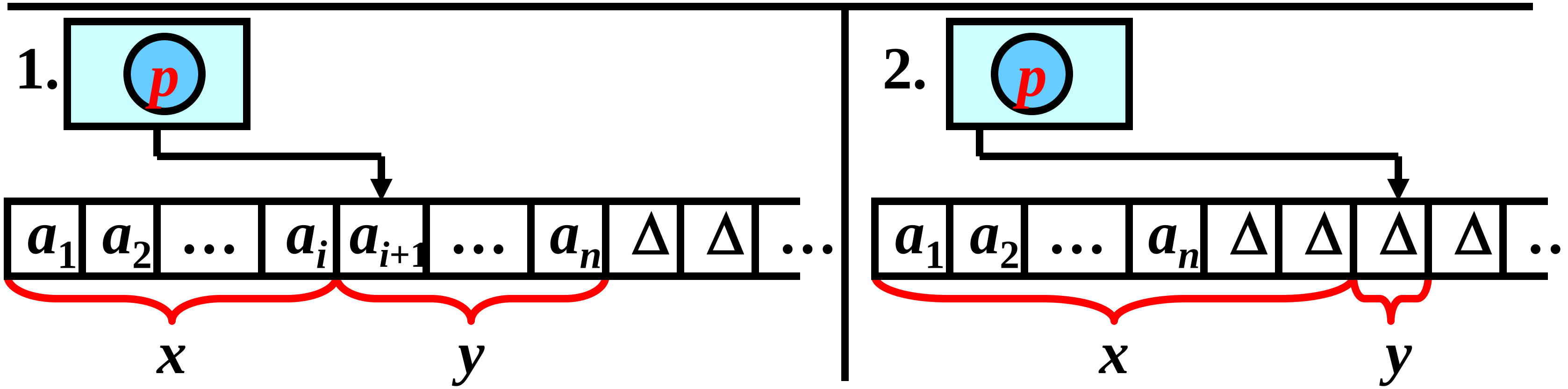


TM Configuration

Gist: Instantaneous description of TM

What does a configuration describes?

1) Current state **2)** Tape Contents **3)** Position of the head



Configuration ***xpy***

Definition: Let $M = (Q, \Sigma, \Gamma, R, s, F)$ be a TM.

A configuration of M is a string $\chi = xpy$, where

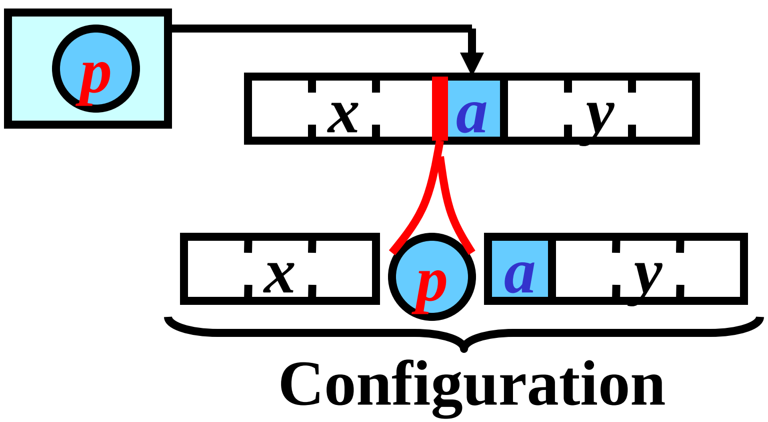
$x \in \Gamma^*$, $p \in Q$, $y \in \Gamma^*(\Gamma - \{\Delta\}) \cup \{\Delta\}$.

Stationary Move

Definition: Let χ, χ' be two configurations of *M*. Then, *M* makes a *stationary move* from χ to χ' according to *r*, written as $\chi \vdash_S \chi'$ [*r*] or, simply, $\chi \vdash_S \chi'$ if

$$\chi = xpay,\ \chi' = xqby \text{ and } r: pa \rightarrow qbS \in R$$

Illustration:



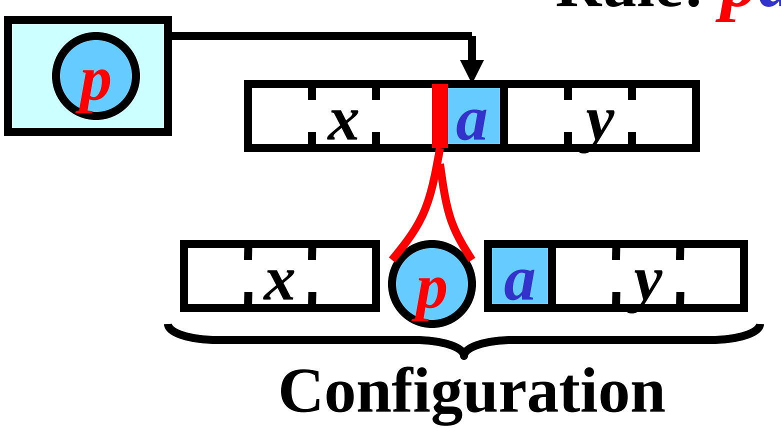
Stationary Move

Definition: Let χ, χ' be two configurations of M. Then, M makes a *stationary move* from χ to χ' according to r, written as $\chi \vdash_S \chi'$ [r] or, simply, $\chi \vdash_S \chi'$ if

$$\chi = xpay,\ \ \chi' = xqby \text{ and } r: pa \rightarrow qbS \in R$$

Illustration:

Rule: $pa \rightarrow qbS$

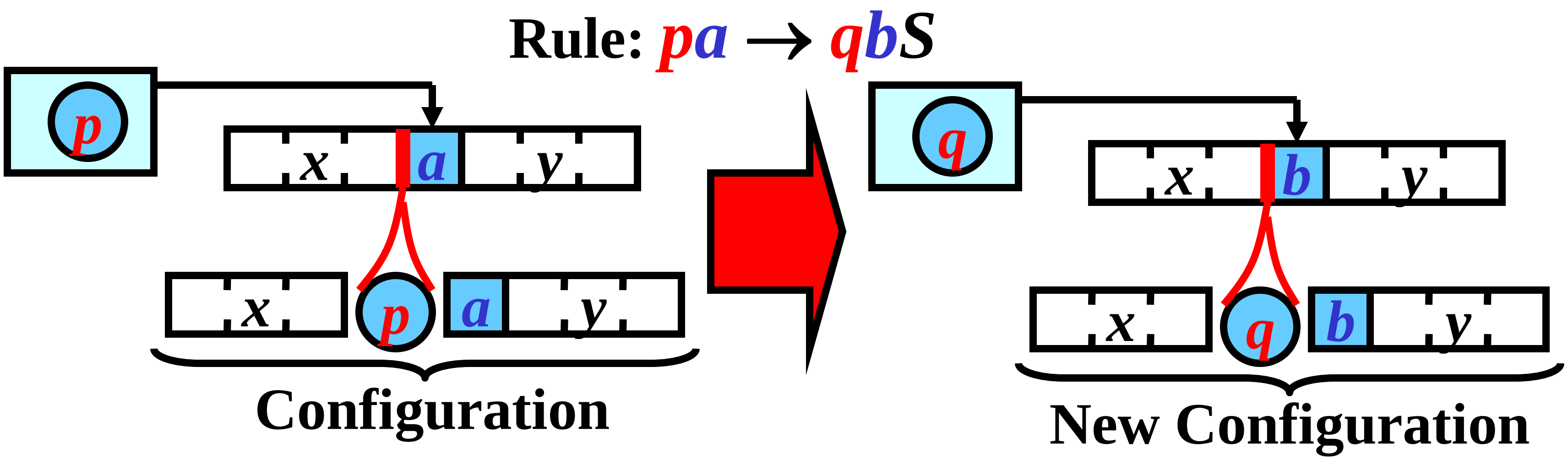


Stationary Move

Definition: Let χ, χ' be two configurations of M. Then, M makes a *stationary move* from χ to χ' according to *r*, written as $\chi \vdash_S \chi'$ [*r*] or, simply, $\chi \vdash_S \chi'$ if

$$\chi = xpay,\ \ \chi' = xqby \text{ and } r: pa \rightarrow qbS \in R$$

Illustration:

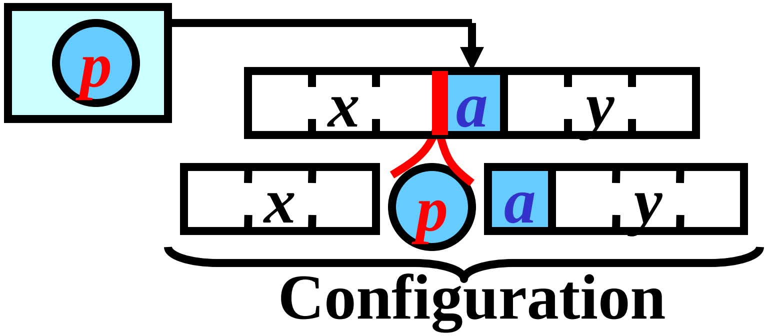


Right Move

Definition: Let χ, χ' be two configurations of *M*. Then, *M* makes a *right move* from χ to χ' according to *r*, written as $\chi \vdash_R \chi'$ [*r*] or, simply, $\chi \vdash_R \chi'$ if $\chi = xpay$, $r: pa \rightarrow qbR \in R$ and

(1) $\chi' = xbqy$, $y \neq \varepsilon$ **or**

(2) $\chi' = xbq\Delta$, $y = \varepsilon$

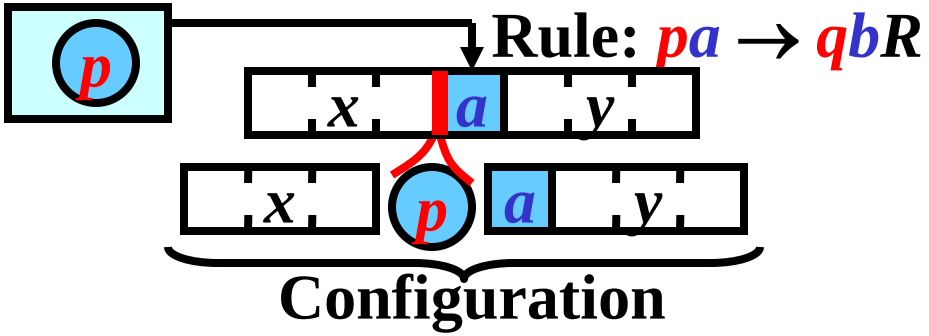


Right Move

Definition: Let χ, χ' be two configurations of M. Then, M makes a *right move* from χ to χ' according to r, written as $\chi \vdash_R \chi'$ [r] or, simply, $\chi \vdash_R \chi'$ if $\chi = \boldsymbol{xpay}$, r: $\boldsymbol{pa} \rightarrow \boldsymbol{qbR} \in R$ and

(1) $\chi' = \boldsymbol{xbqy}$, $y \neq \varepsilon$ **or**

(2) $\chi' = \boldsymbol{xbq}\Delta$, $y = \varepsilon$

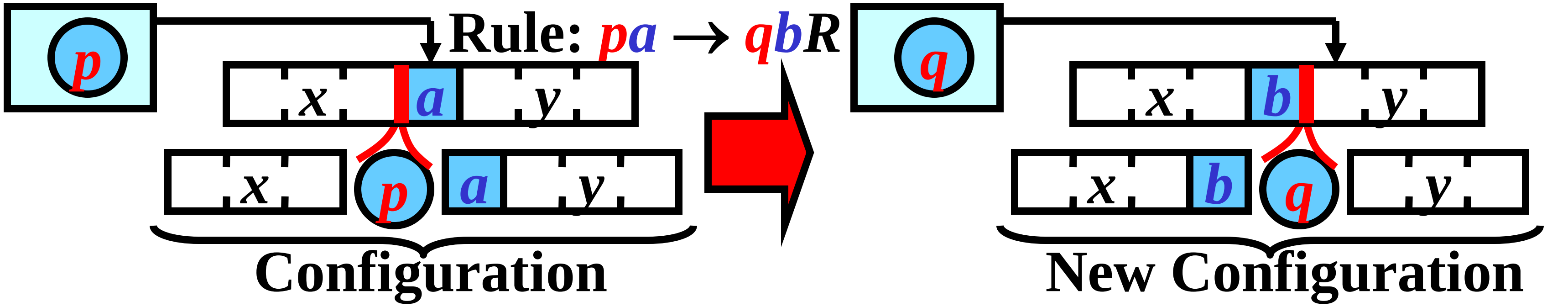


Right Move

Definition: Let χ, χ' be two configurations of M. Then, M makes a *right move* from χ to χ' according to r, written as $\chi \vdash_R \chi'$ [r] or, simply, $\chi \vdash_R \chi'$ if $\chi = \boldsymbol{xpay}$, r: $\boldsymbol{pa} \rightarrow \boldsymbol{qbR} \in R$ and

(1) $\chi' = \boldsymbol{xbqy}$, $y \neq \varepsilon$ **or**

(2) $\chi' = \boldsymbol{xbq}\Delta$, $y = \varepsilon$

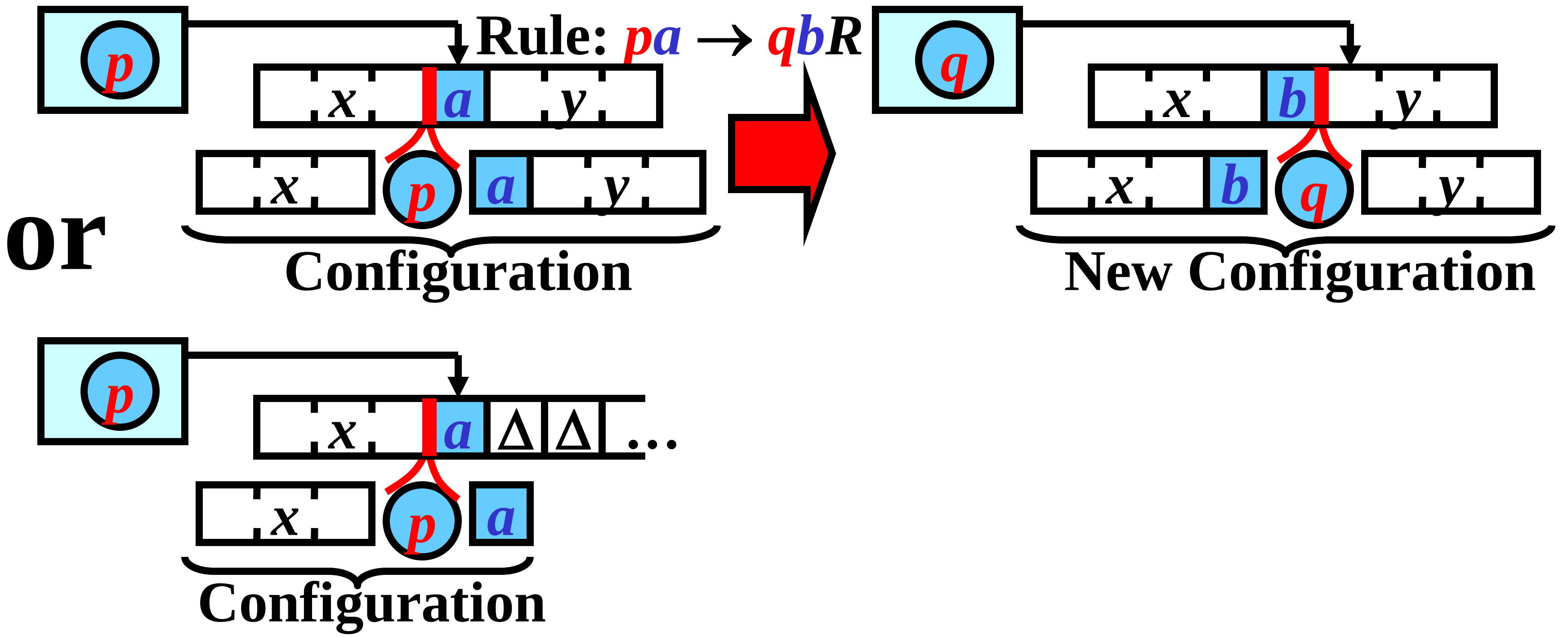


Right Move

Definition: Let χ, χ' be two configurations of *M*. Then, *M* makes a *right move* from χ to χ' according to *r*, written as $\chi \vdash_R \chi'$ [*r*] or, simply, $\chi \vdash_R \chi'$ if χ = ***xpay***, *r*: ***pa*** → ***qbR*** ∈ *R* and

(1) χ' = ***xbqy***, *y* ≠ ε **or**

(2) χ' = ***xbq***Δ, *y* = ε

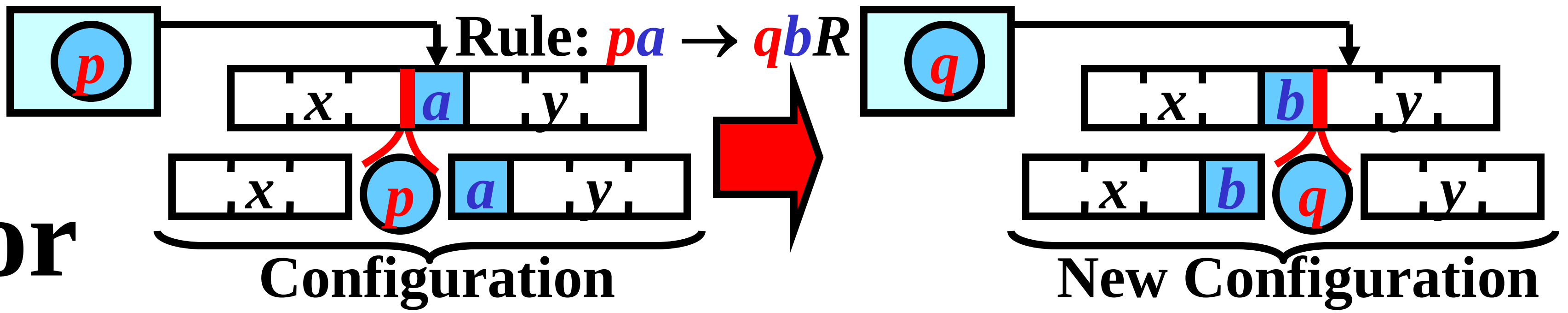


Right Move

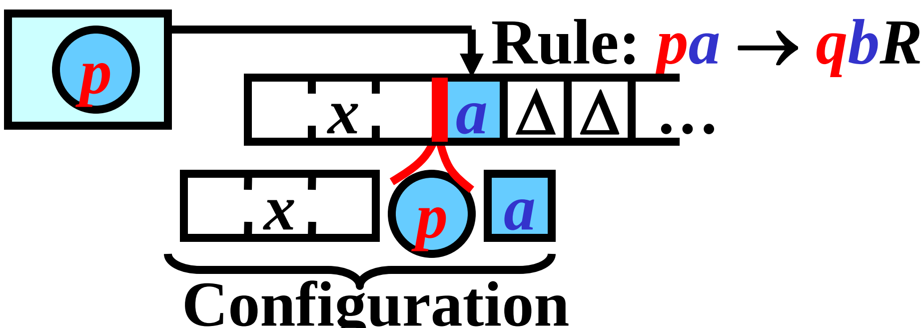
Definition: Let χ, χ' be two configurations of M. Then, M makes a *right move* from χ to χ' according to r, written as $\chi \vdash_R \chi'$ [r] or, simply, $\chi \vdash_R \chi'$ if $\chi = \boldsymbol{xpay}$, r: $\boldsymbol{pa} \rightarrow \boldsymbol{qbR} \in R$ and

(1) $\chi' = \boldsymbol{xbqy}$, $y \neq \varepsilon$ **or**

(2) $\chi' = \boldsymbol{xbq}\Delta$, $y = \varepsilon$



or

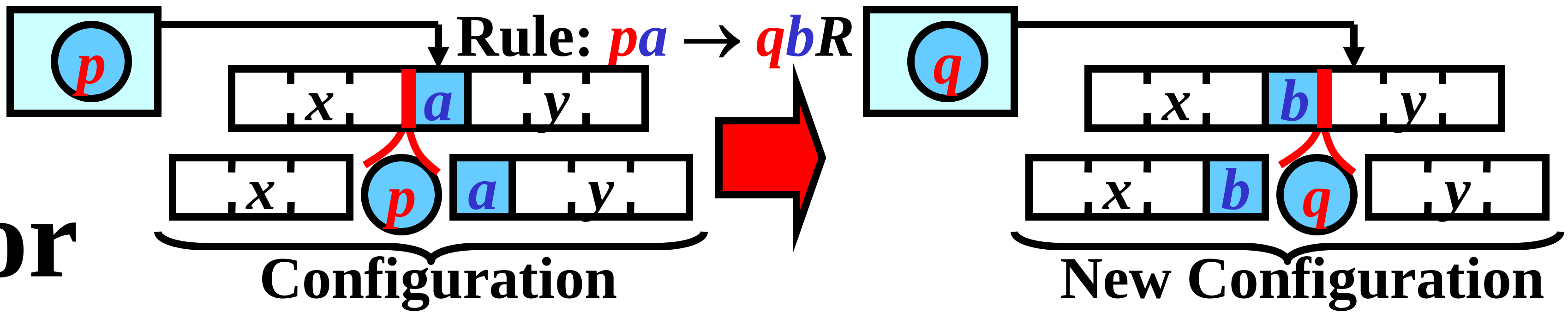


Right Move

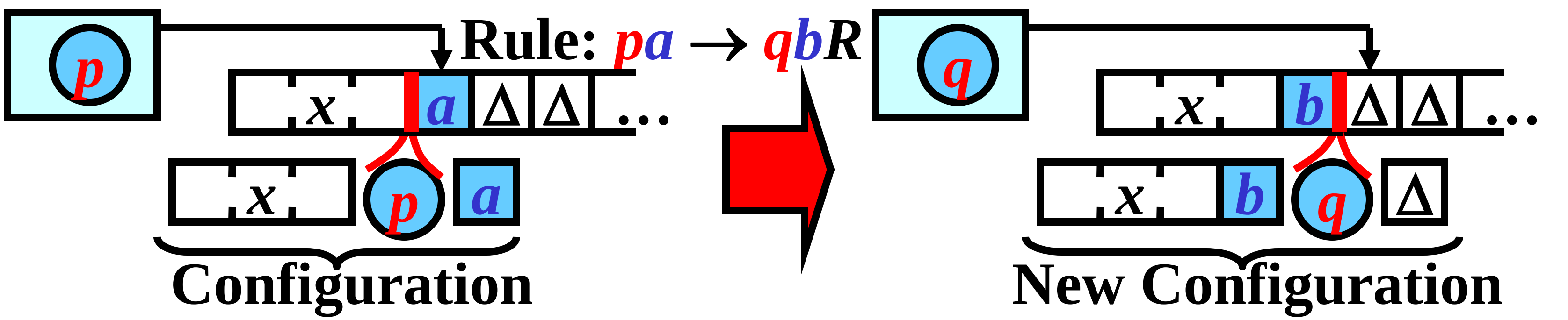
Definition: Let χ, χ' be two configurations of *M*. Then, *M* makes a *right move* from χ to χ' according to *r*, written as $\chi \vdash_R \chi'$ [*r*] or, simply, $\chi \vdash_R \chi'$ if $\chi = xpay$, *r*: $pa \rightarrow qbR \in R$ and

(1) $\chi' = xbqy$, $y \neq \varepsilon$ **or**

(2) $\chi' = xbq\Delta$, $y = \varepsilon$



or

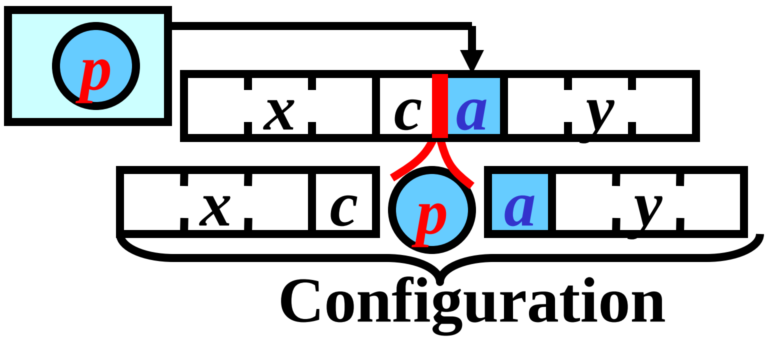


Left Move

Definition: Let χ, χ' be two configurations of *M*. Then, *M* makes a *left move* from χ to χ' according to *r*, written as $\chi \vdash_L \chi'$ [*r*] or, simply, $\chi \vdash_L \chi'$ if

(1) $\chi = xcpay$, $\chi' = xqcby$, $y \neq \varepsilon$ or $b \neq \Delta$, $r: pa \rightarrow qbL \in R$ **or**

(2) $\chi = xcpa$, $\chi' = xqc$, $r: pa \rightarrow q\Delta L \in R$

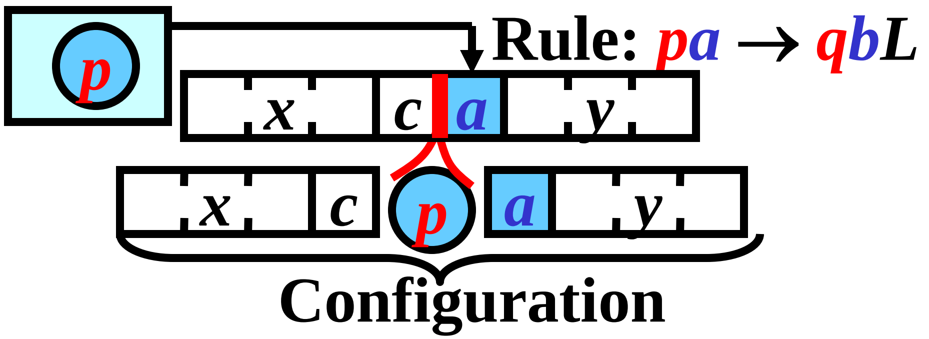


Left Move

Definition: Let χ, χ' be two configurations of *M*. Then, *M* makes a *left move* from χ to χ'according to *r*, written as $\chi \vdash_L \chi'$ [*r*] or, simply, $\chi \vdash_L \chi'$ if

(1) $\chi = xcpay$, $\chi' = xqcby$, $y \neq \varepsilon$ or $b \neq \Delta$, r: $pa \rightarrow qbL \in R$ **or**

(2) $\chi = xcpa$, $\chi' = xqc$, r: $pa \rightarrow q\Delta L \in R$

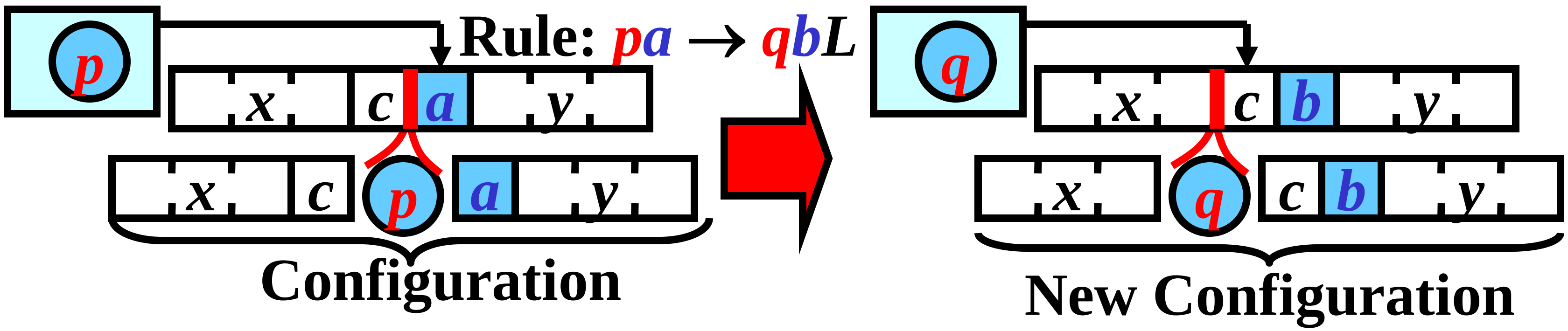


Left Move

Definition: Let χ, χ' be two configurations of M. Then, M makes a *left move* from χ to χ' according to r, written as $\chi \vdash_L \chi'$ [r] or, simply, $\chi \vdash_L \chi'$ if

(1) $\chi = xcpay$, $\chi' = xqcby$, $y \neq \varepsilon$ or $b \neq \Delta$, r: $pa \rightarrow qbL \in R$ **or**

(2) $\chi = xcpa$, $\chi' = xqc$, r: $pa \rightarrow q\Delta L \in R$

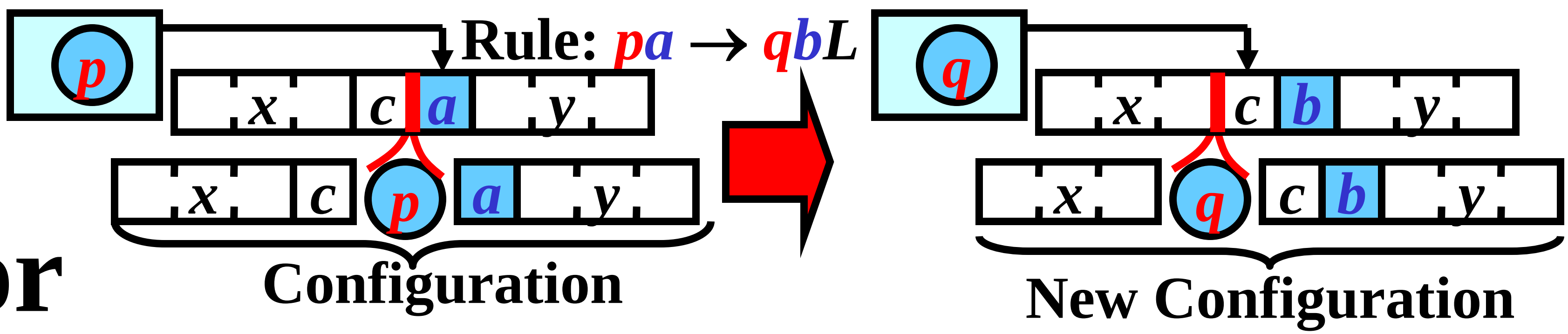


Left Move

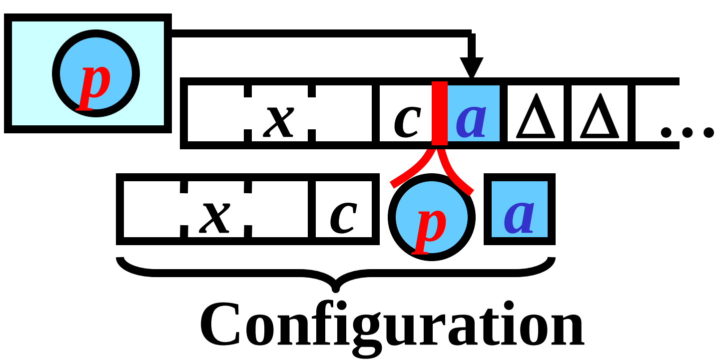
Definition: Let χ, χ' be two configurations of M. Then, M makes a *left move* from χ to χ' according to r, written as $\chi \vdash_L \chi'$ [r] or, simply, $\chi \vdash_L \chi'$ if

(1) $\chi = xcpay$, $\chi' = xqcby$, $y \neq \varepsilon$ or $b \neq \Delta$, r: $pa \rightarrow qbL \in R$ **or**

(2) $\chi = xcpa$, $\chi' = xqc$, r: $pa \rightarrow q\Delta L \in R$



or

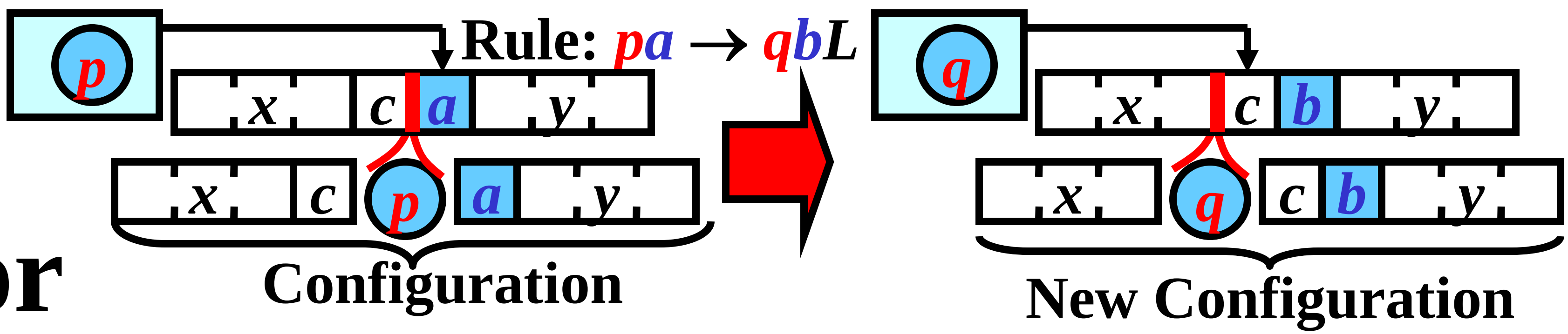


Left Move

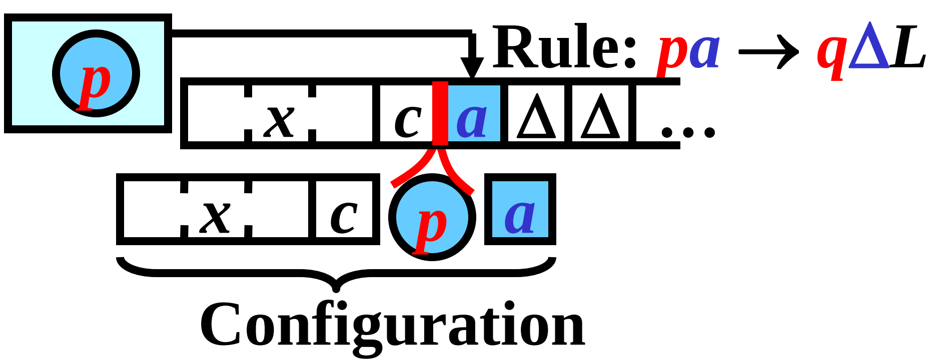
Definition: Let χ, χ' be two configurations of *M*. Then, *M* makes a *left move* from χ to χ'according to *r*, written as $\chi \vdash_L \chi' \; [r]$ or, simply, $\chi \vdash_L \chi'$ if

(1) $\chi = xcpay$, $\chi' = xqcby$, $y \neq \varepsilon$ or $b \neq \Delta$, $r: pa \to qbL \in R$ **or**

(2) $\chi = xcpa$, $\chi' = xqc$, $r: pa \to q\Delta L \in R$



or

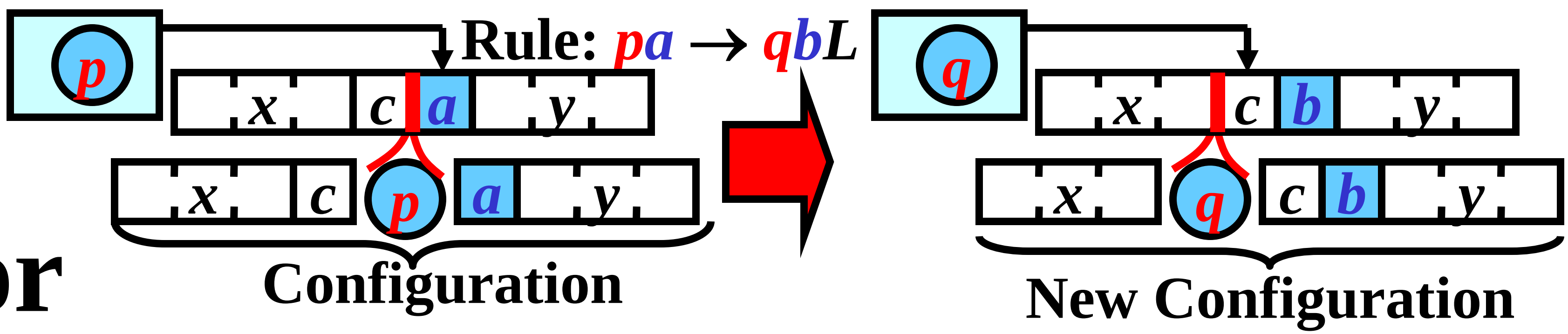


Left Move

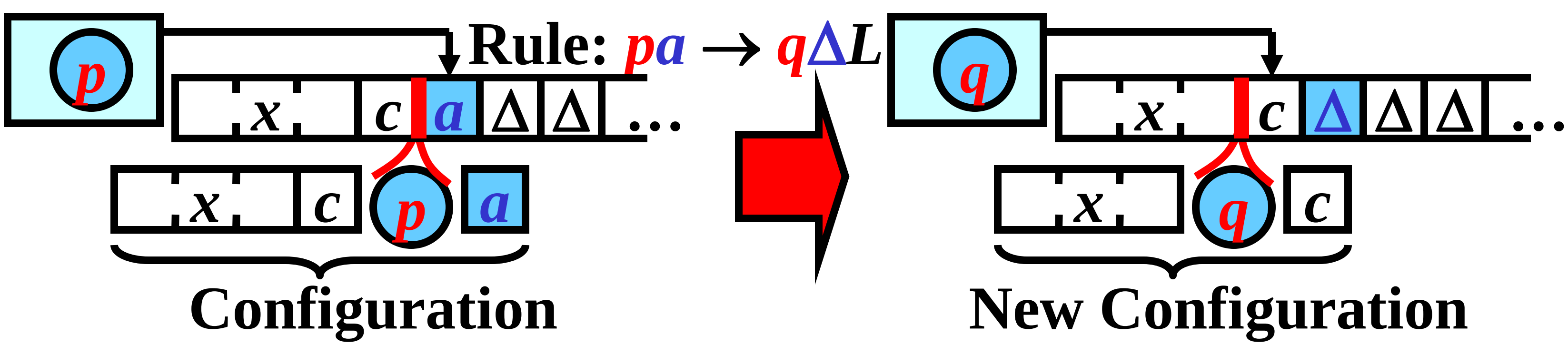
Definition: Let χ, χ' be two configurations of *M*. Then, *M* makes a *left move* from χ to χ' according to *r*, written as $\chi \vdash_L \chi'$ [*r*] or, simply, $\chi \vdash_L \chi'$ if

(1) $\chi = xcpay$, $\chi' = xqcby$, $y \neq \varepsilon$ or $b \neq \Delta$, $r: pa \rightarrow qbL \in R$ **or**

(2) $\chi = xcpa$, $\chi' = xqc$, $r: pa \rightarrow q\Delta L \in R$



or



Move

Definition: Let χ, χ' be two configurations of *M*. Then, *M* makes a *move* from χ to χ' according to a rule ***r***, written as $\chi \vdash \chi'$ [***r***] or, simply, $\chi \vdash \chi'$ if $\chi \vdash_X \chi'$ [***r***] for some $X \in \{S, R, L\}$.

Sequence of Moves 1/2

Gist: Several consecutive computational steps

Definition: Let χ be a configuration. *M* makes *zero moves* from χ to χ; in symbols,

$$\chi \vdash^0 \chi\ [\varepsilon] \text{ or, simply, } \chi \vdash^0 \chi$$

Definition: Let $\chi_0, \chi_1, ..., \chi_n$ be a sequence of configurations, $n \geq 1$, and $\chi_{i-1} \vdash \chi_i\ [r_i]$, $r_i \in R$, for all $i = 1, ..., n$; that is,

$$\chi_0 \vdash \chi_1\ [r_1] \vdash \chi_2\ [r_2] \dots \vdash \chi_n\ [r_n]$$

Then, *M* makes *n moves* from χ_0 to χ_n,

$$\chi_0 \vdash^n \chi_n\ [r_1 ... r_n] \text{ or, simply, } \chi_0 \vdash^n \chi_n$$

Sequence of Moves 2/2

If $\chi_0 \vdash^n \chi_n$ [ρ] for some $n \geq 1$, then

$\chi_0 \vdash^+ \chi_n$ [ρ] or, simply, $\chi_0 \vdash^+ \chi_n$

If $\chi_0 \vdash^n \chi_n$ [ρ] for some $n \geq 0$, then

$\chi_0 \vdash^* \chi_n$ [ρ] or, simply, $\chi_0 \vdash^* \chi_n$

Example: Consider

apbc |– ***aqac*** [**1**: ***pb*** → ***qaS***], and

aqac |– ***acrc*** [**2**: ***qa*** → ***rcR***].

Then, ***apbc*** $\vdash^2$ ***acrc*** [**1 2**],

apbc $\vdash^+$ ***acrc*** [**1 2**],

apbc $\vdash^*$ ***acrc*** [**1 2**]

TM as a Language Acceptor

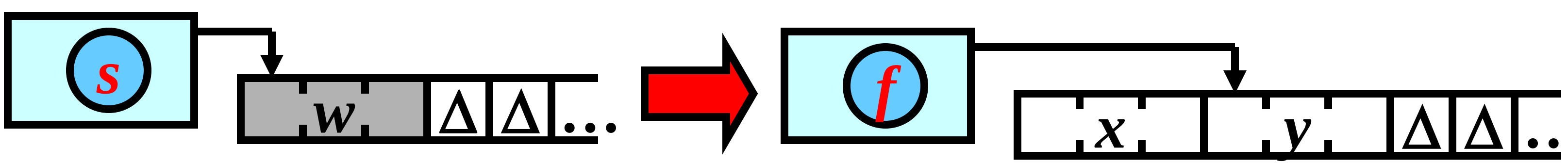
Gist: *M* accepts *w* by a sequence of moves from *s* to a final state.

Definition: Let $M = (Q, \Sigma, \Gamma, R, s, F)$ be a TM.
The *language accepted by M*, $L(M)$, is defined as:

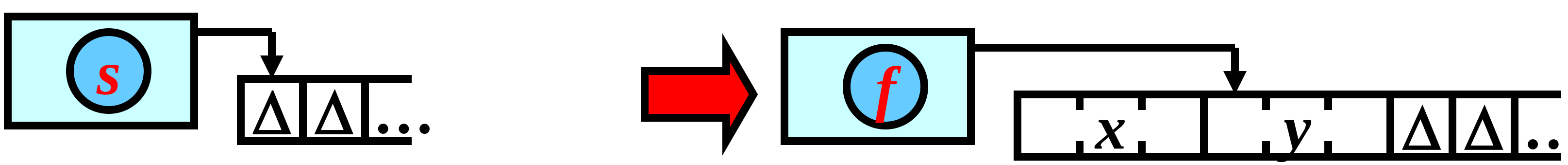
$$L(M) = \{w: w \in \Sigma^*, \mathbf{s}w \vdash^* x\mathbf{f}y;\ x, y \in \Gamma^*, f \in F\} \cup \{\varepsilon: \mathbf{s}\Delta \vdash^* x\mathbf{f}y;\ x, y \in \Gamma^*, f \in F\}$$

Illustration:

- **For $w \neq \varepsilon$:**

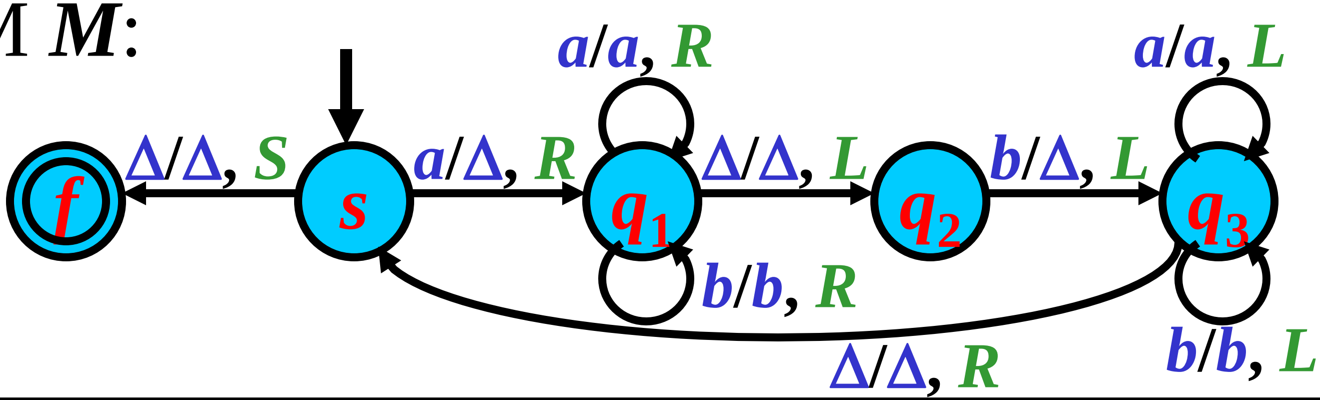


- **For $w = \varepsilon$:**



TM as an Acceptor: Example

TM ***M***:



$sabba \vdash \Delta q_1 abb \vdash \Delta a q_1 bb \vdash \Delta ab q_1 b \vdash \Delta abb q_1 \Delta \vdash \Delta ab q_2 b$
$\vdash \Delta a q_3 b \vdash \Delta q_3 ab \vdash q_3 \Delta ab \vdash \Delta sab \vdash \Delta\Delta q_1 b \vdash \Delta\Delta q_1 b$
$\vdash \Delta\Delta b q_1 \Delta \vdash \Delta\Delta q_2 b \vdash \Delta q_3 \Delta \vdash s\Delta \vdash f\Delta$

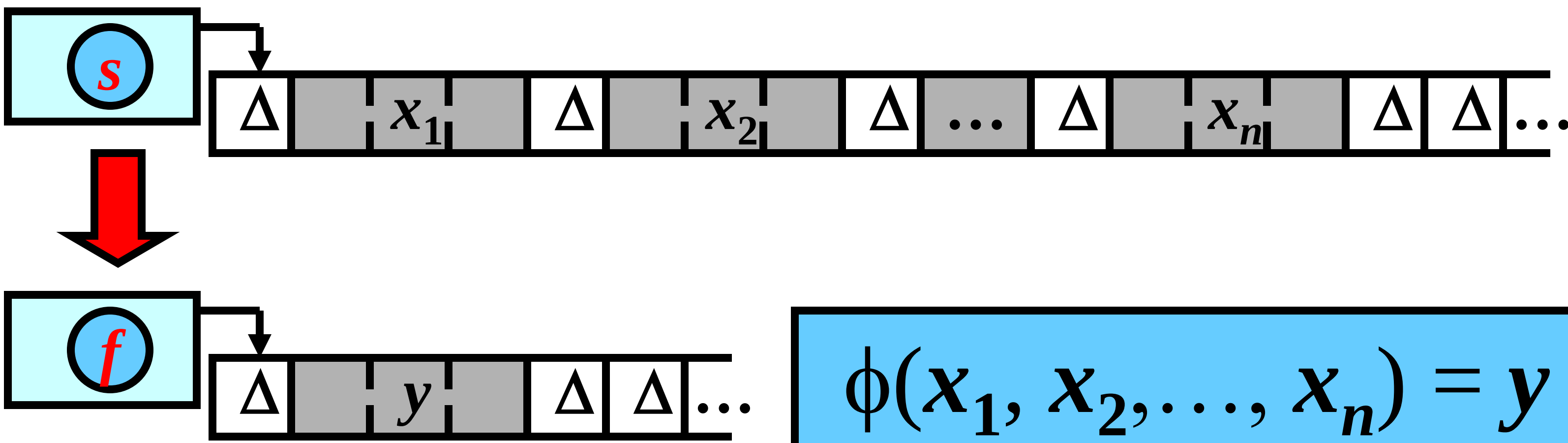
Summary: $abba \in L(M)$

Note: $L(M) = \{a^n b^n : n \geq 0\}$

TM as a Computational Model

Definition: Let $M = (Q, \Sigma, \Gamma, R, s, F)$ be a TM; *n-place function* ϕ *is computed by M* provided that $s\Delta x_1\Delta x_2\ldots\Delta x_n \vdash^* f\Delta y$ with $f \in F$ **if and only if** $\phi(x_1, x_2,\ldots, x_n) = y$.

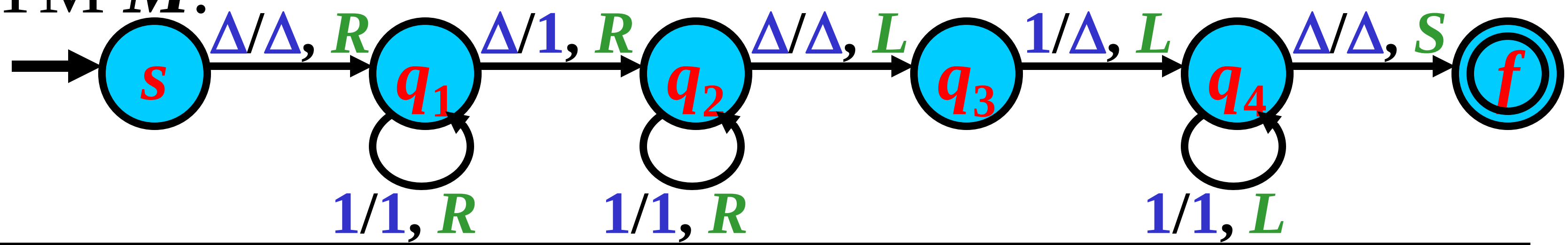
Illustration:



$\phi(x_1, x_2,\ldots, x_n) = y$

TM as a Computational Model: Example

TM ***M***:



$s\Delta 11\Delta 11 \vdash \Delta q_1 11\Delta 11 \vdash \Delta 1 q_1 1\Delta 11 \vdash \Delta 11 q_1 \Delta 11 \vdash \Delta 111 q_2 11$
$\vdash \Delta 1111 q_2 1 \vdash \Delta 11111 q_2 \Delta \vdash \Delta 1111 q_3 1 \vdash \Delta 111 q_4 1$
$\vdash \Delta 11 q_4 11 \vdash \Delta 1 q_4 111 \vdash \Delta q_4 1111 \vdash q_4 \Delta 1111$
$\vdash f\Delta 1111$

Summary: $\phi(11, 11) = 1111$

Note: $\phi(x_1, x_2) = x_1 + x_2$, where

- $x_1 = 1^a$ represents a natural number a
- $x_2 = 1^b$ represents a natural number b

Deterministic Turing Machine (DTM)

Gist: Deterministic TM makes no more than one move from any configuration.

Definition: Let $M = (Q, \Sigma, \Gamma, R, s, F)$ be a TM. M is a *deterministic TM* if for each rule $pa \rightarrow qbt \in R$ it holds that $R - \{pa \rightarrow qbt\}$ contains no rule with the left-hand side equal to pa.

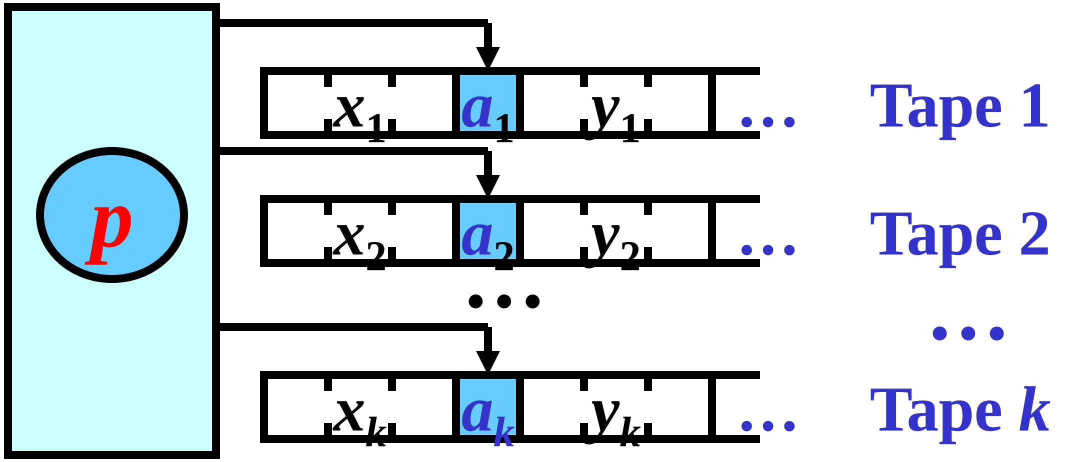
Theorem: For every TM M, there is an equivalent DTM M_d.

Proof: See page 634 in [Meduna: Automata and Languages]

k-Tape Turing Machine

Gist: Turing machine with *k* tapes

Illustration:



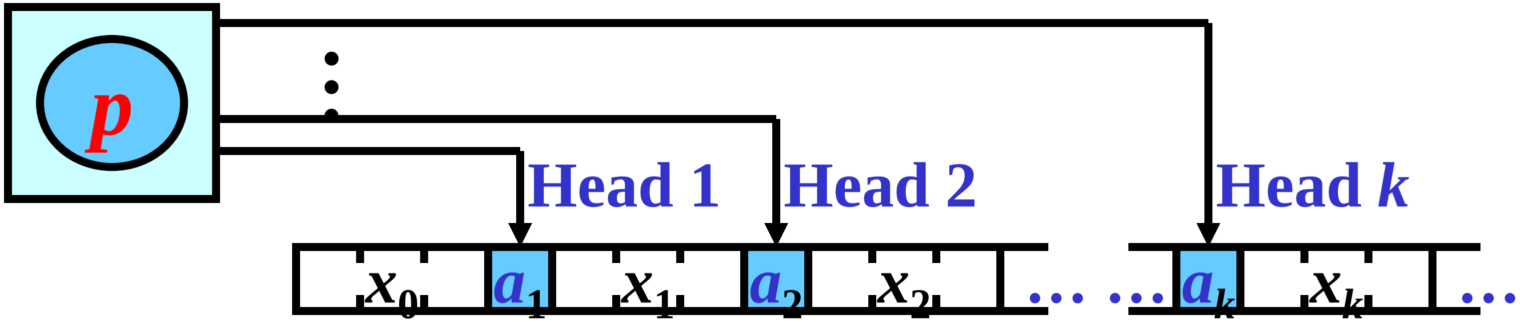
Theorem: For every k-tape TM M_t, there is an equivalent TM M.

Proof: See page 662 in [Meduna: Automata and Languages]

k-Head Turing Machine

Gist: Turing machine with k heads

Illustration:



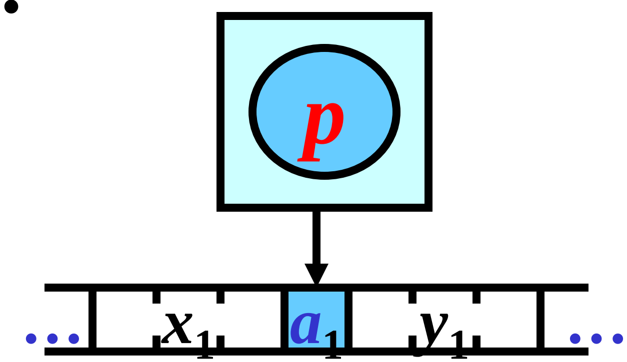
Theorem: For every k-head TM M_h, there is an equivalent TM M.

Proof: See page 667 in [Meduna: Automata and Languages]

TM with Two-way Infinite Tapes

Gist: Turing machine with tape infinite both to the right and to the left

Illustration:



Theorem: For every TM with two-way infinite tapes M_b, there is an equivalent TM M.

Proof: See page 673 in [Meduna: Automata and Languages]

Description of a Turing Machine

Gist: Turing machine representation using a string over {0, 1}

- Assume that TM M has the form $M = (Q, \Sigma, \Gamma, R, q_0, \{q_1\})$, where $Q = \{q_0, q_1, \dots, q_m\}$, $\Gamma = \{a_0, a_1, \dots, a_n\}$ so that $a_0 = \Delta$
- Let δ is the mapping from $(Q \cup \Gamma \cup \{S, L, R\})$ to $\{0, 1\}^*$ defined as:

$$\delta(S) = 01,\ \delta(L) = 001,\ \delta(R) = 0001,$$
$$\delta(q_i) = 0^{i+1}1 \text{ for all } i = 0 \dots m,$$
$$\delta(a_i) = 0^{i+1}1 \text{ for all } i = 0 \dots n$$

- For every $r: pa \rightarrow qbt \in R$ we define

$$\delta(r) = \delta(p)\delta(a)\delta(q)\delta(b)\delta(t)1$$

- Let $R = \{r_0, r_1, \dots, r_k\}$. Then

$$\delta(M) = 111\delta(r_0)\delta(r_1)\dots\delta(r_k)1 \text{ is the description of TM } M$$

Description of TM: Example

$M = (Q, \Sigma, \Gamma, R, q_0, \{q_1\})$, where

$Q = \{q_0, q_1\}$; $\Sigma = \{a_1, a_2\}$; $\Gamma = \{\Delta, a_1, a_2\}$;

$R = \{1\text{: } q_0a_1 \rightarrow q_0a_2R,\ 2\text{: } q_0a_2 \rightarrow q_0a_1R,\ 3\text{: } q_0\Delta \rightarrow q_1\Delta S\}$

Task: Decription of M, $\delta(M)$.

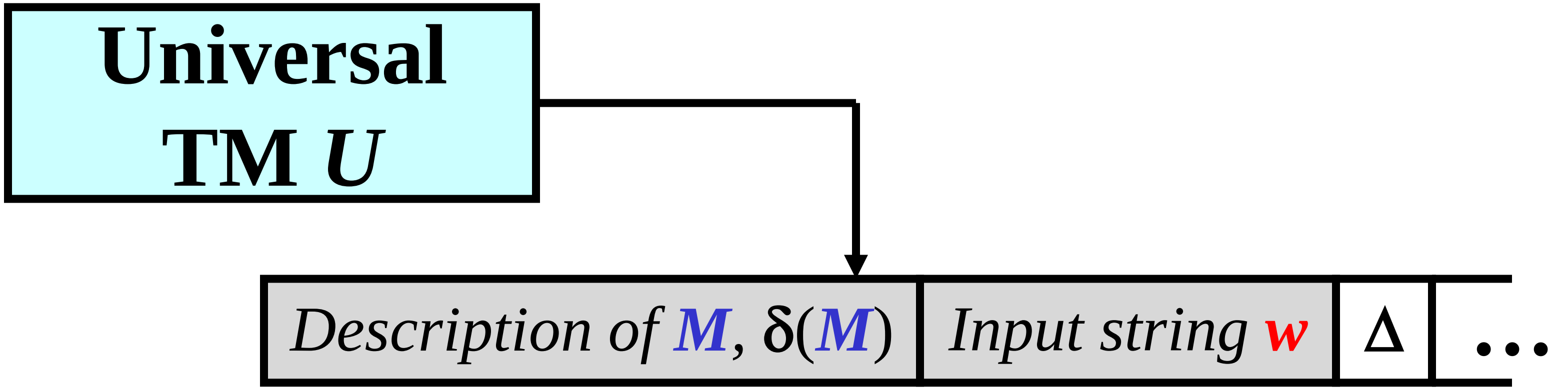
$\delta(S) = 01$, $\delta(L) = 001$, $\delta(R) = 0001$,
$\delta(q_0) = 01$, $\delta(q_1) = 001$,
$\delta(\Delta) = 01$, $\delta(a_1) = 001$, $\delta(a_2) = 0001$.

$$\begin{aligned}\delta(M) &= 111\delta(1)\delta(2)\delta(3)1 \\ &= 111\delta(q_0)\delta(a_1)\delta(q_0)\delta(a_2)\delta(R)1 \\ &\quad\ \delta(q_0)\delta(a_2)\delta(q_0)\delta(a_1)\delta(R)1 \\ &\quad\ \delta(q_0)\delta(\Delta)\delta(q_1)\delta(\Delta)\delta(S)11 \\ &= 11101001010001000011 \\ &\quad\ 0100010100100011 \\ &\quad\ 0101001010111\end{aligned}$$

Universal Turing Machine

Gist: Universal TM can simulate every DTM

Illustration:



Note: Universal TM ***U*** reads the description of TM ***M***, and the input string ***w***, and then simulates the moves that ***M*** makes with ***w***.

Unrestricted Grammar: Definition

Gist: Generalization of CFG

Definition: *An unrestricted grammar* (URG) is a quadruple $G = (N, T, P, S)$, where

- N is an alphabet of *nonterminals*
- T is an alphabet of *terminals*, $N \cap T = \emptyset$
- P is a finite set of *rules* of the form $x \rightarrow y$, where $x \in (N \cup T)^* N(N \cup T)^*, y \in (N \cup T)^*$
- $S \in N$ is the *start nonterminal*

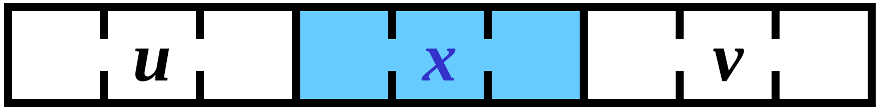
Mathematical Note on Rules:

- Strictly mathematically, P is a finite relation from $(N \cup T)^* N (N \cup T)^*$ to $(N \cup T)^*$
- Instead of $(x, y) \in P$, we write $x \rightarrow y \in P$

Derivation Step

Gist: A change of a string by a rule.

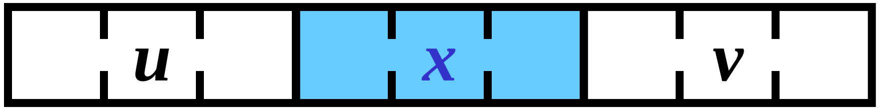
Definition: Let $G = (N, T, P, S)$ be a URG. Let $\boldsymbol{u}, \boldsymbol{v} \in (N \cup T)^*$ and $\boldsymbol{p}: \boldsymbol{x} \rightarrow \boldsymbol{y} \in P$. Then, $\boldsymbol{uxv}$ *directly derives* $\boldsymbol{uyv}$ *according to* $\boldsymbol{p}$ in G, written as $\boldsymbol{uxv} \Rightarrow \boldsymbol{uyv}$ [$\boldsymbol{p}$] or, simply, $\boldsymbol{uxv} \Rightarrow \boldsymbol{uyv}$.



Derivation Step

Gist: A change of a string by a rule.

Definition: Let $G = (N, T, P, S)$ be a URG. Let $\boldsymbol{u}, \boldsymbol{v} \in (N \cup T)^*$ and $\boldsymbol{p}$: $\boldsymbol{x} \rightarrow \boldsymbol{y} \in P$. Then, $\boldsymbol{uxv}$ *directly derives* $\boldsymbol{uyv}$ *according to* $\boldsymbol{p}$ in G, written as $\boldsymbol{uxv} \Rightarrow \boldsymbol{uyv}$ [$\boldsymbol{p}$] or, simply, $\boldsymbol{uxv} \Rightarrow \boldsymbol{uyv}$.



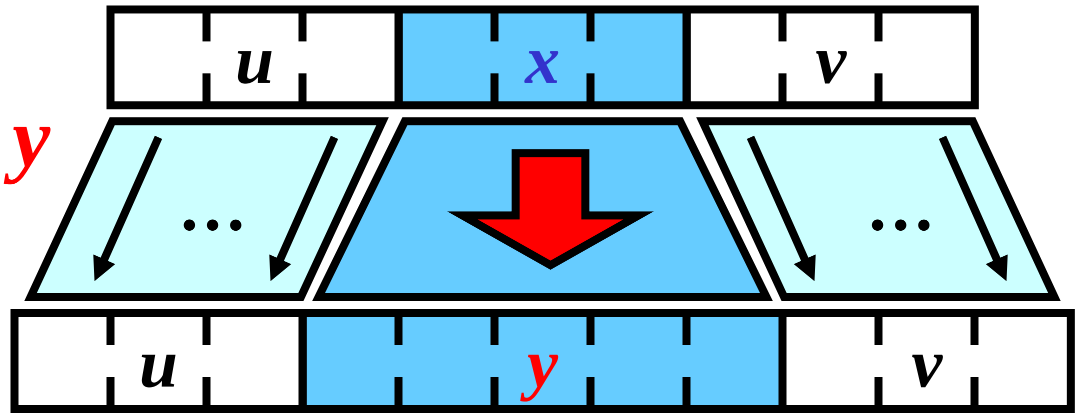
Rule: $\boldsymbol{x} \rightarrow \boldsymbol{y}$

Derivation Step

Gist: A change of a string by a rule.

Definition: Let $G = (N, T, P, S)$ be a URG. Let $\boldsymbol{u}, \boldsymbol{v} \in (N \cup T)^*$ and p: $x \rightarrow y \in P$. Then, $\boldsymbol{uxv}$ *directly derives* $\boldsymbol{uyv}$ *according to* p in G, written as $\boldsymbol{uxv} \Rightarrow \boldsymbol{uyv}$ [p] or, simply, $\boldsymbol{uxv} \Rightarrow \boldsymbol{uyv}$.

Rule: $x \rightarrow y$

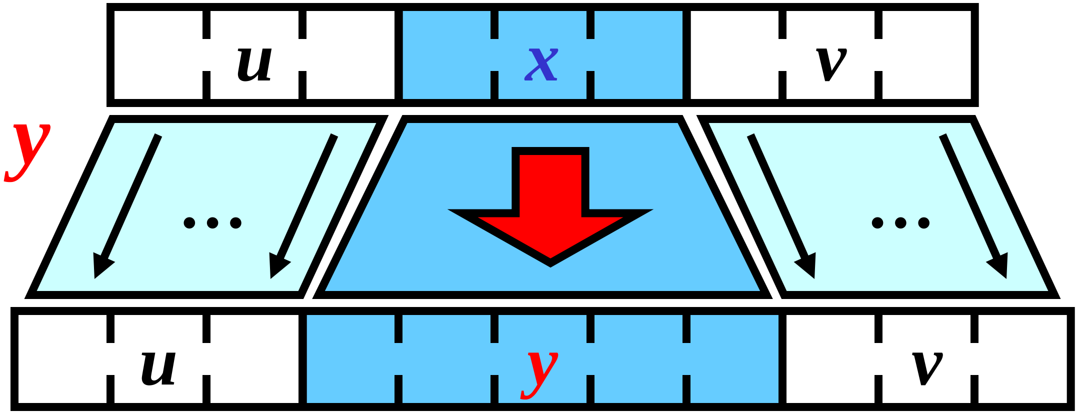


Derivation Step

Gist: A change of a string by a rule.

Definition: Let $G = (N, T, P, S)$ be a URG. Let $\boldsymbol{u}, \boldsymbol{v} \in (N \cup T)^*$ and $\boldsymbol{p}: \boldsymbol{x} \to \boldsymbol{y} \in P$. Then, $\boldsymbol{uxv}$ *directly derives* $\boldsymbol{uyv}$ *according to* $\boldsymbol{p}$ in G, written as $\boldsymbol{uxv} \Rightarrow \boldsymbol{uyv}$ $[\boldsymbol{p}]$ or, simply, $\boldsymbol{uxv} \Rightarrow \boldsymbol{uyv}$.

Rule: $\boldsymbol{x} \to \boldsymbol{y}$



Note: $\Rightarrow^n$, $\Rightarrow^+$, $\Rightarrow^*$ and $L(G)$ are defined by analogy with the corresponding definitions in terms of CFGs.

Unrestricted Grammar: Example

$G = (N, T, P, S)$, where $N = \{S, A, B\}$, $T = \{a\}$

$P = \{$ 1: $S \to ASB$, 2: $S \to a$,
3: $Aa \to aaA$, 4: $AB \to \varepsilon$ $\}$

$S \Rightarrow a$ [2]

$S \Rightarrow A\underline{S}B$ [1] $\Rightarrow \underline{Aa}B$ [2] $\Rightarrow aa\underline{AB}$ [3] $\Rightarrow aa$ [4]

$S \Rightarrow A\underline{S}B$ [1] $\Rightarrow AA\underline{S}BB$ [1] $\Rightarrow A\underline{Aa}BB$ [2] $\Rightarrow$
$\underline{Aa}aABB$ [3] $\Rightarrow aa\underline{Aa}ABB$ [3] $\Rightarrow$
$aaaaA\underline{AB}B$ [3] $\Rightarrow aaaa\underline{AB}$ [4] $\Rightarrow aaaa$ [4]

⋮

Note: $L(G) = \{a^{2^n} : n \geq 0\}$

Recursively Enumerable Languages

Definition: Let L be a language. L is a *resurcively enumerable language* if there exists a Turing machine M that $L = L(M)$.

Theorem: For every URG G, there is a TM M such that $L(G) = L(M)$.

Proof: See page 714 in [Meduna: Automata and Languages]

Theorem: For every TM M, there is a URG G such that $L(M) = L(G)$.

Proof: See page 715 in [Meduna: Automata and Languages]

Conclusion: The fundamental models for recursively enumerable languages are

1) Unrestricted grammars **2) Turing Machines**

Context-Sensitive Grammar

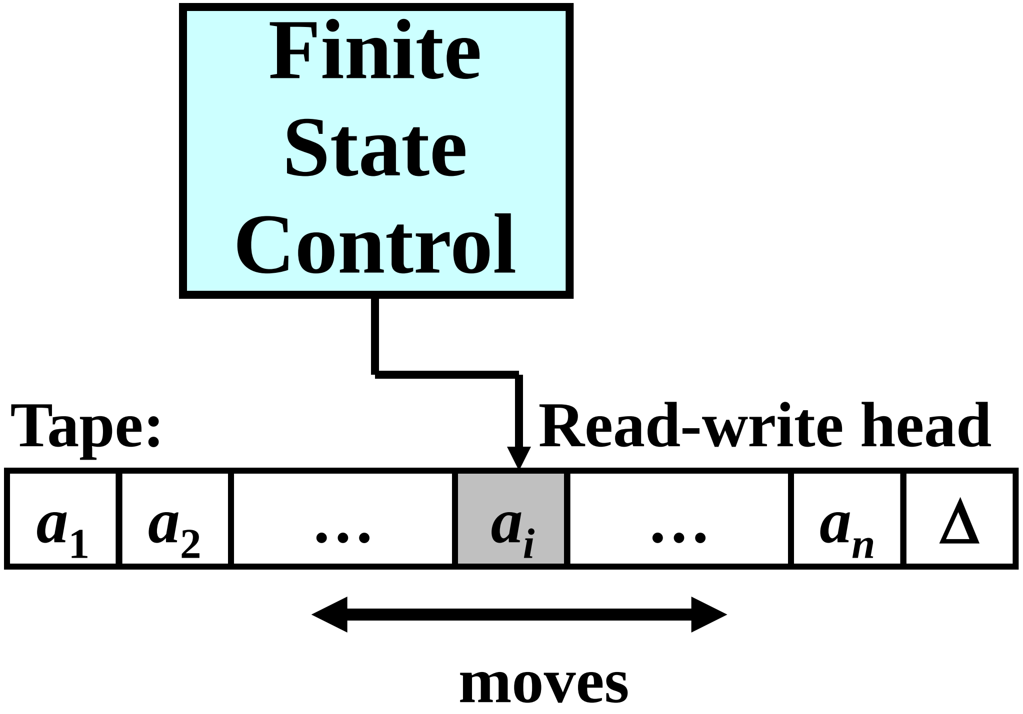
Gist: Restriction of URG

Definition: Let $G = (N, T, P, S)$ be an unrestricted grammar. G is *a context-sensitive* (or *length-increasing*) *grammar* (CSG) if every rule $\boldsymbol{x} \rightarrow \boldsymbol{y} \in P$ satisfies $|x| \leq |y|$.

Note: $\Rightarrow$, $\Rightarrow^n$, $\Rightarrow^+$, $\Rightarrow^*$ and $L(G)$ are defined by analogy with the definitions of the corresponding notions on URGs.

Linear Bounded Automaton

Gist: A Turing machine with a Tape Bounded by the Length of the Input String.

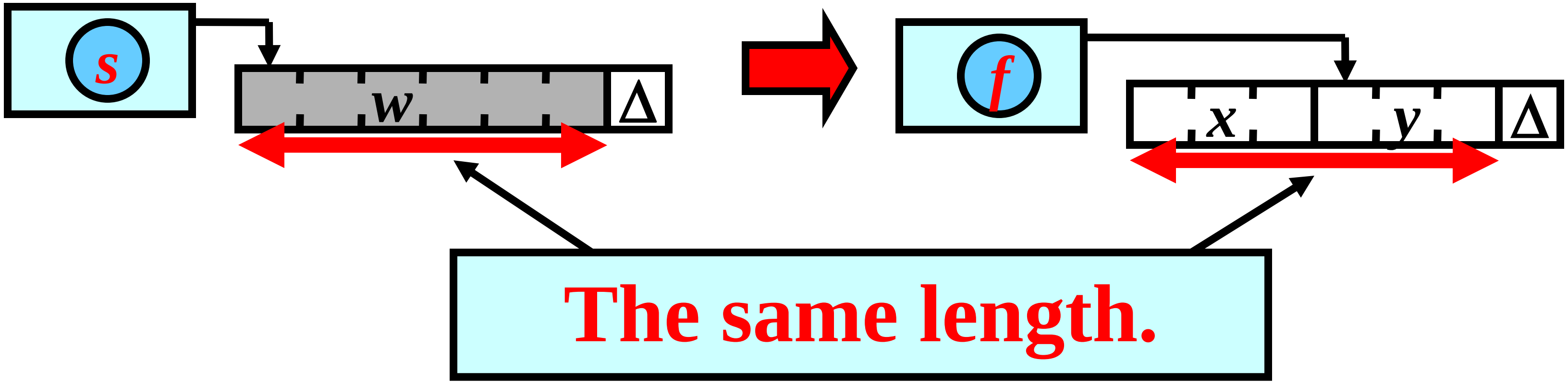


Linear Bounded Automaton: Definition

Gist: With *w* on its tape, *M's* tape is restricted to |*w*| squares.

Definition: *A linear bounded automaton* (LBA) is a TM that cannot extend its tape by any rule.

Accepted language: Illustration



Context-sensitive Languages

Definition: Let L be a language. L is a *context-sensitive* if there exists a context-sensitive grammar G that $L = L(G)$.

Theorem: For every CSG G, there is an LBA M such that $L(G) = L(M)$.

Proof: See page 732 in [Meduna: Automata and Languages]

Theorem: For every LBA M, there is a CSG G such that $L(M) = L(G)$.

Proof: See page 734 in [Meduna: Automata and Languages]

Conclusion: The fundamental models for context-sensitive languages are

1) **Context-sensitive grammars**
2) **Linear bounded automata**

Right-Linear Grammar: Definition

Gist: A CFG in which every rule has a string of terminals followed by no more that one nonterminal on the right-hand side.

Definition: Let $G = (N, T, P, S)$ be a CFG. G is *a right-linear grammar* (RLG) if every rule $\boldsymbol{A} \rightarrow \boldsymbol{x} \in P$ satisfies $\boldsymbol{x} \in \boldsymbol{T^*} \cup \boldsymbol{T^*N}$.

Example:

$G = (N, T, P, S)$, where $N = \{S, A\}$, $T = \{a, b\}$

$P = \{1: S \rightarrow aS,\ 2: S \rightarrow aA,\ 3: A \rightarrow bA,\ 4: A \rightarrow b\ \}$

- $S \Rightarrow a\underline{A}$ [2] $\Rightarrow ab$ [4]
- $S \Rightarrow a\underline{S}$ [1] $\Rightarrow aa\underline{A}$ [2] $\Rightarrow aab$ [4]
- $S \Rightarrow a\underline{A}$ [2] $\Rightarrow ab\underline{A}$ [3] $\Rightarrow abb$ [4]

⋮

Note: $L(G) = \{a^m b^n : m, n \geq 1\}$

Grammars for Regular Languages

Theorem: For every RLG G, there is an FA M such that $L(G) = L(M)$.

Proof: See page 575 in [Meduna: Automata and Languages]

Theorem: For every FA M, there is an RLG G such that $L(M) = L(G)$.

Proof: See page 583 in [Meduna: Automata and Languages]

Conclusion: Grammars for regular languages are

Right-linear grammar

Grammars: Summary

Languages	Grammar	Form of rules $x \to y$
Recursively enumerable	Unrestricted	$x \in (N \cup T)^* N (N \cup T)^*$ $y \in (N \cup T)^*$
Context-sensitive	Context-sensitive	$x \in (N \cup T)^* N (N \cup T)^*$ $y \in (N \cup T)^*$, $\lvert x \rvert \leq \lvert y \rvert$
Context-free	Context-free	$x \in N$ $y \in (N \cup T)^*$
Regular	Right-Linear	$x \in N$ $y \in T^* \cup T^* N$

Generalization

Restriction

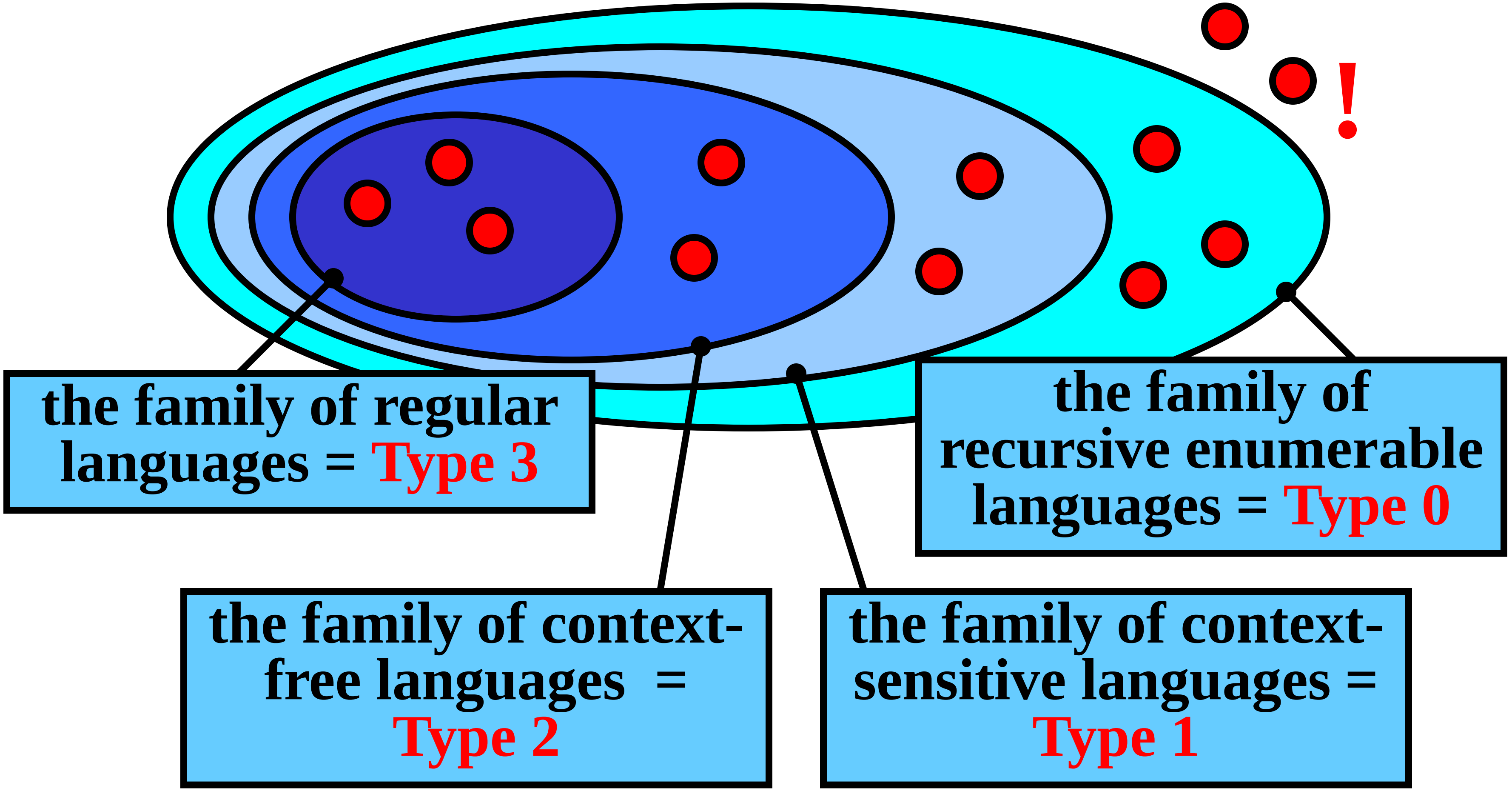
Automata: Summary

Languages	Accepting Device
Recursively enumerable	Turing machine
Context-sensitive	Linear bounded automaton
Context-free	Pushdown automaton
Regular	Finite automaton

Generalization ↑

Restriction ↓

Chomsky Hierarchy



Type 3 ⊂ Type 2 ⊂ Type 1 ⊂ Type 0

Language $L_{SelfAcceptance}$ 1/2

Gist: $L_{SelfAcceptance}$ is the language over $\{0, 1\}^*$, which contain a string $\delta(M)$, if and only if DTM M accepts $\delta(M)$.

Definition:

$L_{SelfAcceptance} = \{\delta(M): M \text{ is a DTM}, \delta(M) \in L(M)\}$

Illustration:

TM *M*

Language $L_{\text{SelfAcceptance}}$ 1/2

Gist: $L_{\text{SelfAcceptance}}$ is the language over $\{0, 1\}^*$, which contain a string $\delta(M)$, if and only if DTM M accepts $\delta(M)$.

Definition:

$L_{\text{SelfAcceptance}} = \{\delta(M): M \text{ is a DTM}, \delta(M) \in L(M)\}$

Illustration: **TM *M*** → **Description of *M*:** $\delta(M) = \mathbf{1110\ldots1}$

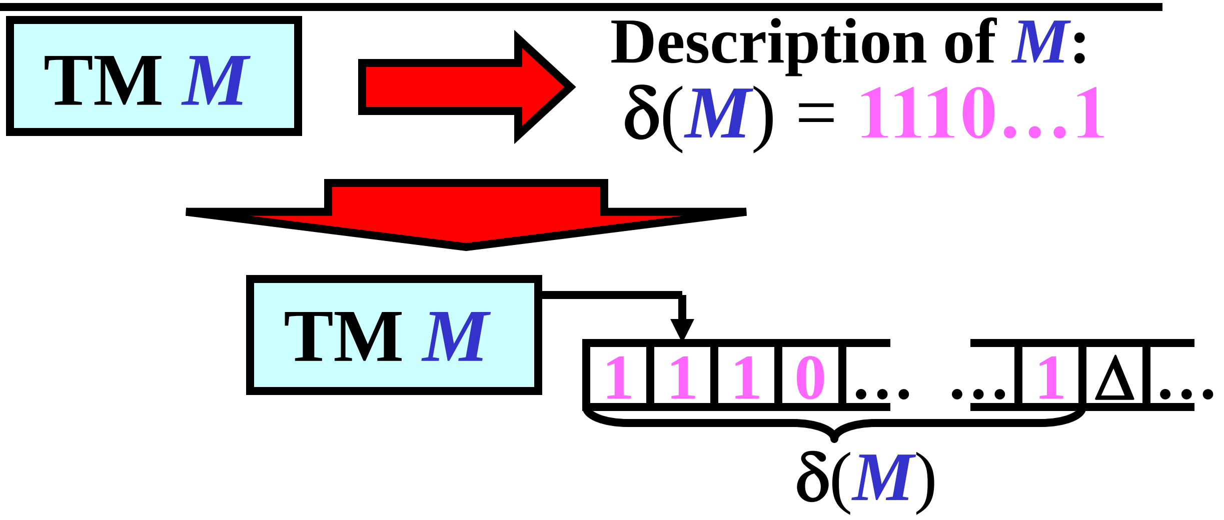
Language $L_{SelfAcceptance}$ 1/2

Gist: $L_{SelfAcceptance}$ is the language over $\{0, 1\}^*$, which contain a string $\delta(M)$, if and only if DTM M accepts $\delta(M)$.

Definition:

$L_{SelfAcceptance} = \{\delta(M): M \text{ is a DTM}, \delta(M) \in L(M)\}$

Illustration:



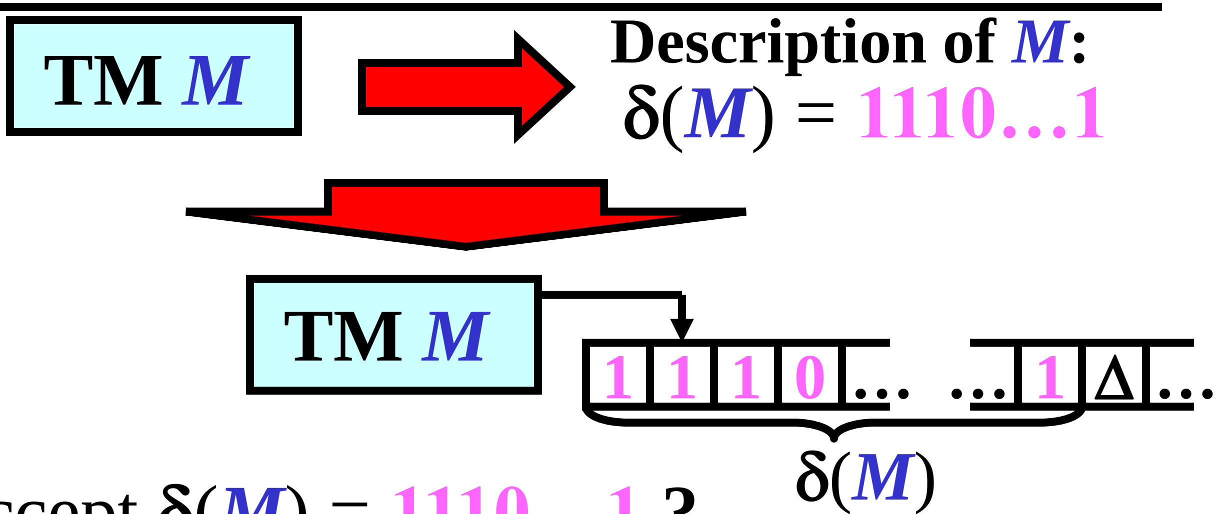
Language $L_{SelfAcceptance}$ 1/2

Gist: $L_{SelfAcceptance}$ is the language over $\{0, 1\}^*$, which contain a string $\delta(M)$, if and only if DTM M accepts $\delta(M)$.

Definition:

$L_{SelfAcceptance} = \{\delta(M): M \text{ is a DTM}, \delta(M) \in L(M)\}$

Illustration:



- Does TM M accept $\delta(M) = 1110\ldots1$?

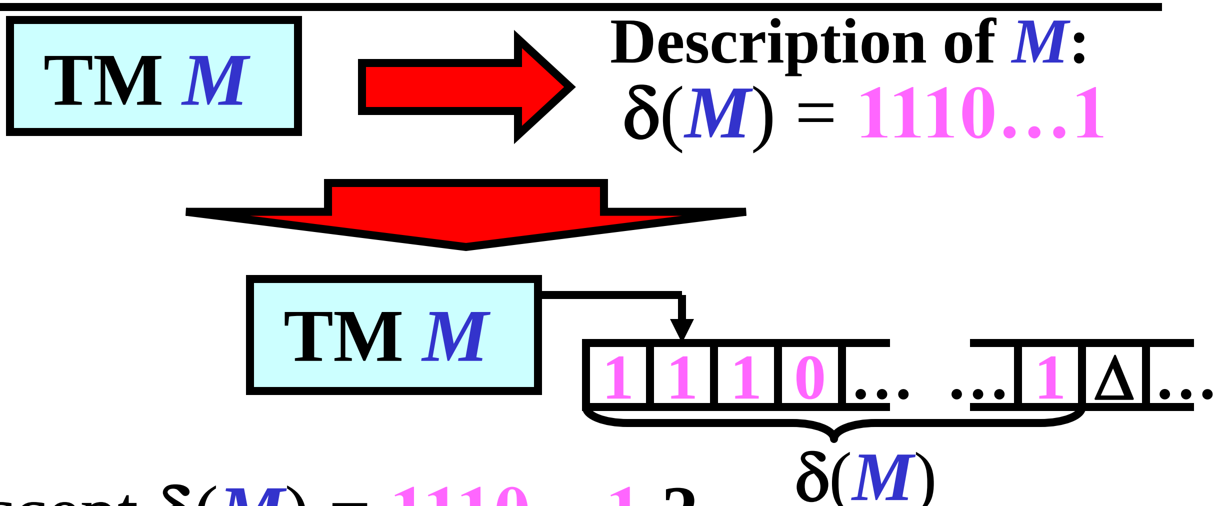
Language $L_{SelfAcceptance}$ 1/2

Gist: $L_{SelfAcceptance}$ is the language over $\{0, 1\}^*$, which contain a string $\delta(M)$, if and only DTM M accepts $\delta(M)$.

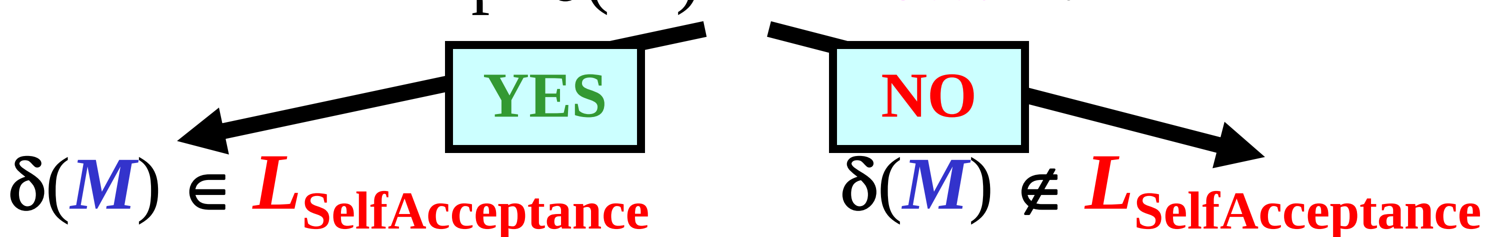
Definition:

$L_{SelfAcceptance} = \{\delta(M): M \text{ is a DTM}, \delta(M) \in L(M)\}$

Illustration:



- Does TM M accept $\delta(M)$ = 1110…1 ?



Language $L_{SelfAcceptance}$ 2/2

Theorem: $L_{SelfAcceptance}$ is accept by some TM.

Proof (idea):

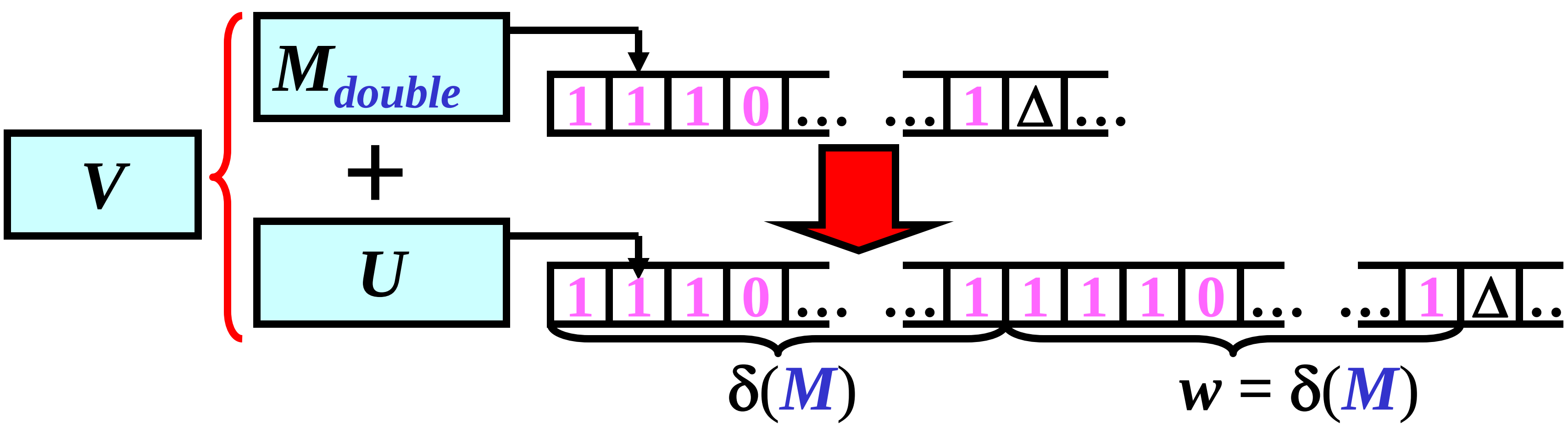
- We construct a DTM V, which:

1) Replace an input string $w = \delta(M)$ with $\delta(M)\delta(M)$

2) Simulate an activity of a universal TM U

- Then, $L(V) = L_{SelfAcceptance}$, thus theorem holds.

Illustration:



Language $L_{\text{NonSelfAcceptance}}$ 1/3

Gist: $L_{\text{NonSelfAcceptance}} = \overline{L_{\text{SelfAcceptance}}}$

Definition:

$$L_{\text{NonSelfAcceptance}} = \{0, 1\}^* - L_{\text{SelfAcceptance}}$$

TM ***M***

Language $L_{\text{NonSelfAcceptance}}$ 1/3

Gist: $L_{\text{NonSelfAcceptance}} = \overline{L_{\text{SelfAcceptance}}}$

Definition:

$$L_{\text{NonSelfAcceptance}} = \{0, 1\}^* - L_{\text{SelfAcceptance}}$$

TM M → **Description of** M**:**

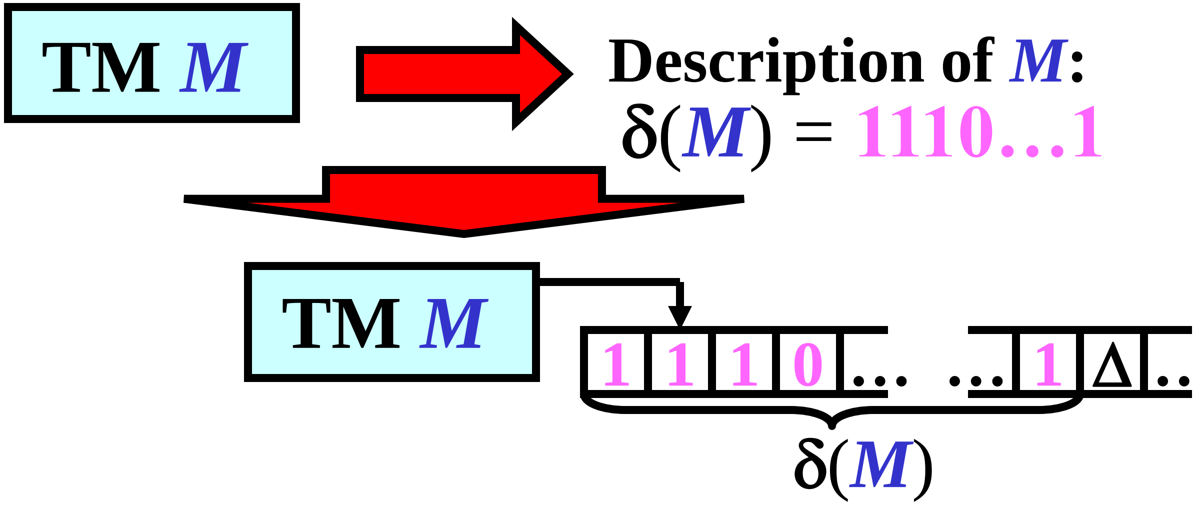
$\delta(M) = 1110\ldots1$

Language $L_{\text{NonSelfAcceptance}}$ 1/3

Gist: $L_{\text{NonSelfAcceptance}} = \overline{L_{\text{SelfAcceptance}}}$

Definition:

$$L_{\text{NonSelfAcceptance}} = \{0, 1\}^* - L_{\text{SelfAcceptance}}$$

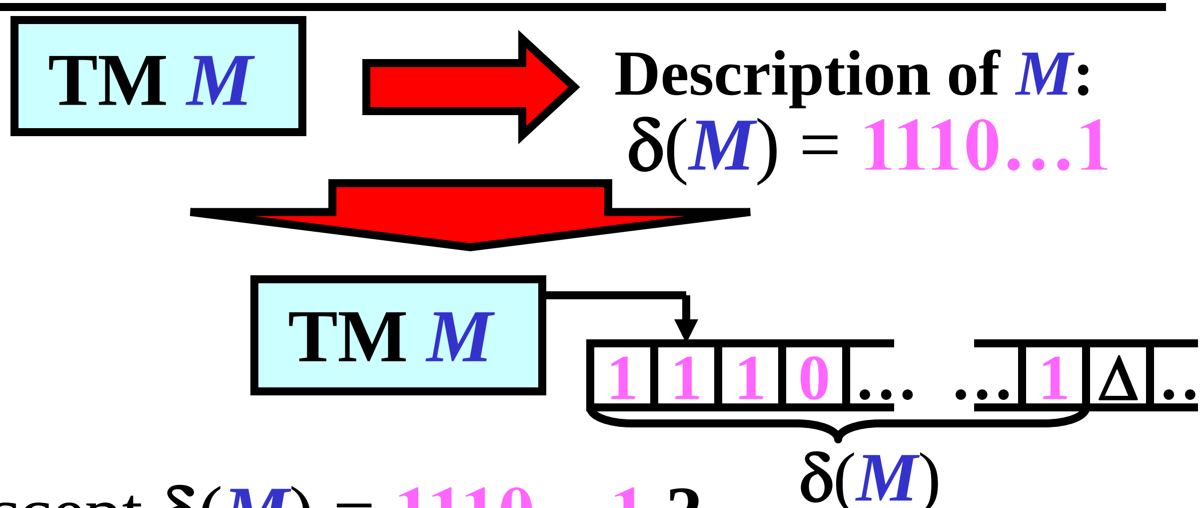


Language $L_{\text{NonSelfAcceptance}}$ 1/3

Gist: $L_{\text{NonSelfAcceptance}} = \overline{L_{\text{SelfAcceptance}}}$

Definition:

$$L_{\text{NonSelfAcceptance}} = \{0, 1\}^* - L_{\text{SelfAcceptance}}$$



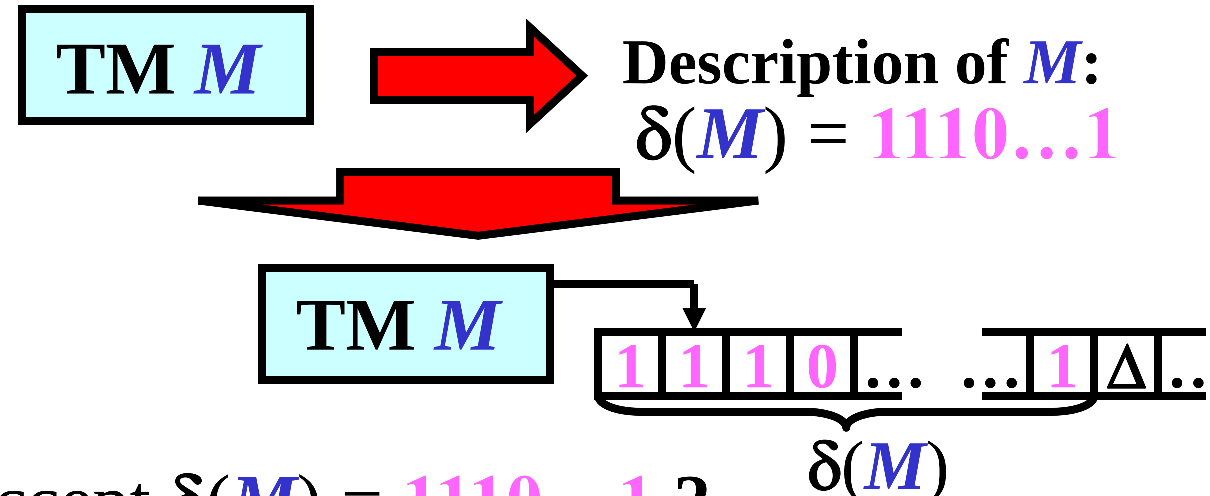
- Does TM M accept $\delta(M) = 1110\ldots1$?

Language $L_{\text{NonSelfAcceptance}}$ 1/3

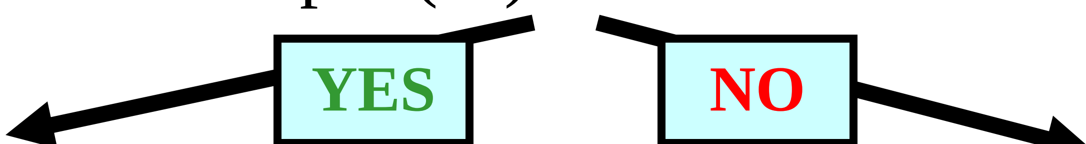
Gist: $L_{\text{NonSelfAcceptance}} = \overline{L_{\text{SelfAcceptance}}}$

Definition:

$$L_{\text{NonSelfAcceptance}} = \{0, 1\}^* - L_{\text{SelfAcceptance}}$$



- Does TM M accept $\delta(M)$ = 1110…1 ?



$\delta(M) \notin L_{\text{NonSelfAcceptance}}$ $\delta(M) \in L_{\text{NonSelfAcceptance}}$

Language $L_{\text{NonSelfAcceptance}}$ 2/3

Theorem: $L_{\text{NonSelfAcceptance}}$ is accept by **no** TM.

Proof (by contradiction):

- Assume that $L_{\text{NonSelfAcceptance}}$ is accepted by a TM. Consider this infinite table:

M_i	$m_i = \delta(M_i)$	*SelfAcceptance*(M_i)
M_1	111001001001101	Yes
M_2	1110101011100101	No
M_3	111001000101000100101	Yes
⋮	⋮	⋮

All TMs ↓

Note:

- ***SelfAcceptance***(M_i) = **Yes** **if** $m_i \in L(M_i)$
  No **if** $m_i \notin L(M_i)$

Language $L_{\text{NonSelfAcceptance}}$ 3/3

- **Notice:** $L_{\text{NonSelfAcceptance}} = \{\boldsymbol{m_i} : \boldsymbol{m_i} \notin L(\boldsymbol{M_i}), i = 1, \ldots\}$
- Let $L(\boldsymbol{M_k}) = L_{\text{NonSelfAcceptance}}$

Language $L_{\text{NonSelfAcceptance}}$ 3/3

- **Notice:** $L_{\text{NonSelfAcceptance}} = \{\boldsymbol{m_i} : \boldsymbol{m_i} \notin L(\boldsymbol{M_i}),\ i = 1, \ldots\}$
- Let $L(\boldsymbol{M_k}) = L_{\text{NonSelfAcceptance}}$
- ***SelfAcceptance***$(\boldsymbol{M_k})$ = **No** implies

$\boldsymbol{m_k} \notin L(\boldsymbol{M_k})$ implies

$\boldsymbol{m_k} \in L_{\text{NonSelfAcceptance}}$ implies

$\boldsymbol{m_k} \in L(\boldsymbol{M_k})$

Language $L_{\text{NonSelfAcceptance}}$ 3/3

- **Notice:** $L_{\text{NonSelfAcceptance}} = \{\boldsymbol{m_i} : \boldsymbol{m_i} \notin L(\boldsymbol{M_i}), i = 1, \ldots\}$
- Let $L(\boldsymbol{M_k}) = L_{\text{NonSelfAcceptance}}$
- ***SelfAcceptance***$(\boldsymbol{M_k})$ = **No** implies

$\boldsymbol{m_k} \notin L(\boldsymbol{M_k})$ implies

$\boldsymbol{m_k} \in L_{\text{NonSelfAcceptance}}$ implies

$\boldsymbol{m_k} \in L(\boldsymbol{M_k})$

contradiction

Language $L_{\text{NonSelfAcceptance}}$ 3/3

- **Notice:** $L_{\text{NonSelfAcceptance}} = \{ m_i : m_i \notin L(M_i), i = 1, \ldots \}$
- Let $L(M_k) = L_{\text{NonSelfAcceptance}}$
- ***SelfAcceptance***(M_k) = **No** implies

  $m_k \notin L(M_k)$ implies

  $m_k \in L_{\text{NonSelfAcceptance}}$ implies

  $m_k \in L(M_k)$

  contradiction

- ***SelfAcceptance***(M_k) = **Yes** implies

  $m_k \in L(M_k)$ implies

  $m_k \notin L_{\text{NonSelfAcceptance}}$ implies

  $m_k \notin L(M_k)$

Language $L_{\text{NonSelfAcceptance}}$ 3/3

- **Notice:** $L_{\text{NonSelfAcceptance}} = \{m_i : m_i \notin L(M_i), i = 1, \ldots\}$
- Let $L(M_k) = L_{\text{NonSelfAcceptance}}$
- ***SelfAcceptance***(M_k) = **No** implies

  $m_k \notin L(M_k)$ implies

  $m_k \in L_{\text{NonSelfAcceptance}}$ implies

  $m_k \in L(M_k)$

  contradiction

- ***SelfAcceptance***(M_k) = **Yes** implies

  $m_k \in L(M_k)$ implies

  $m_k \notin L_{\text{NonSelfAcceptance}}$ implies

  $m_k \notin L(M_k)$

  contradiction

Language $L_{\text{NonSelfAcceptance}}$ 3/3

- **Notice:** $L_{\text{NonSelfAcceptance}} = \{\boldsymbol{m_i} : \boldsymbol{m_i} \notin L(\boldsymbol{M_i}), i = 1, \ldots\}$
- Let $L(\boldsymbol{M_k}) = L_{\text{NonSelfAcceptance}}$
- ***SelfAcceptance***$(\boldsymbol{M_k})$ = **No** implies

  $\boldsymbol{m_k} \notin L(\boldsymbol{M_k})$ implies

  $\boldsymbol{m_k} \in L_{\text{NonSelfAcceptance}}$ implies

  $\boldsymbol{m_k} \in L(\boldsymbol{M_k})$

  contradiction
- ***SelfAcceptance***$(\boldsymbol{M_k})$ = **Yes** implies

  $\boldsymbol{m_k} \in L(\boldsymbol{M_k})$ implies

  $\boldsymbol{m_k} \notin L_{\text{NonSelfAcceptance}}$ implies

  $\boldsymbol{m_k} \notin L(\boldsymbol{M_k})$

  contradiction
- $L_{\text{NonSelfAcceptance}}$ is accepted by **no** TM $\boldsymbol{M_k}$

Recursive Language

Gist: Recursive Language accepts TM that always halt

Definition: Let L be a language. If $L = L(M)$, where M is DTM that always halts, then L is a *recursive language*.

Theorem: The family of recursive languages is closed under complement.

Proof: See page 693 in [Meduna: Automata and Languages]

Theorem: The family of recursively enumerable languages is **not** closed under complement.

Proof: See the $L_{\mathrm{SelfAcceptance}}$

Other Hierarchy of Languages

The family of recursive languages (accepted by TMs that always halt) ⊂ **The family of rec. enumerable languages (accepted by TMs)** ⊂ **All Lang.**