Bezkontextové gramatiky

Bezkontextové gramatiky

Příklad:

$$
\begin{aligned}
\langle \text{STMT} \rangle &\rightarrow \langle \text{IF-STMT} \rangle \mid \langle \text{WHILE-STMT} \rangle \mid \langle \text{BLOCK-STMT} \rangle \mid \\
&\quad \langle \text{ASSG-STMT} \rangle \\
\langle \text{IF-STMT} \rangle &\rightarrow \textbf{if } \langle \text{BOOL-EXPR} \rangle \textbf{ then } \langle \text{STMT} \rangle \textbf{ else } \langle \text{STMT} \rangle \\
\langle \text{WHILE-STMT} \rangle &\rightarrow \textbf{while } \langle \text{BOOL-EXPR} \rangle \textbf{ do } \langle \text{STMT} \rangle \\
\langle \text{BLOCK-STMT} \rangle &\rightarrow \textbf{begin } \langle \text{STMT-LIST} \rangle \textbf{ end} \\
\langle \text{STMT-LIST} \rangle &\rightarrow \langle \text{STMT} \rangle \mid \langle \text{STMT} \rangle \, ; \, \langle \text{STMT-LIST} \rangle \\
\langle \text{ASSG-STMT} \rangle &\rightarrow \langle \text{VAR} \rangle := \langle \text{ARITH-EXPR} \rangle \\
\langle \text{BOOL-EXPR} \rangle &\rightarrow \langle \text{ARITH-EXPR} \rangle \langle \text{COMPARE-OP} \rangle \langle \text{ARITH-EXPR} \rangle \\
\langle \text{COMPARE-OP} \rangle &\rightarrow < \mid > \mid \leq \mid \geq \mid = \mid \neq \\
\langle \text{ARITH-EXPR} \rangle &\rightarrow \langle \text{VAR} \rangle \mid \langle \text{CONST} \rangle \mid \\
&\quad (\langle \text{ARITH-EXPR} \rangle \langle \text{ARITH-OP} \rangle \langle \text{ARITH-EXPR} \rangle) \\
\langle \text{ARITH-OP} \rangle &\rightarrow + \mid - \mid * \mid / \\
\langle \text{CONST} \rangle &\rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
\langle \text{VAR} \rangle &\rightarrow a \mid b \mid c \mid \cdots \mid x \mid y \mid z
\end{aligned}
$$

Bezkontextové gramatiky

Symbolům, které mají v předchozím příkladě tvar $\langle \text{XXX} \rangle$, se říká **neterminální symboly** (**neterminály**).

Pravidla popisují, jaké řetězce může daný neterminál reprezentovat.

Z neterminálu $\langle \text{STMT} \rangle$ můžeme například dostat text

$$\textbf{while } x \leq y \textbf{ do begin } x := (x + 1)\,;\; y := (y - 1) \textbf{ end}$$

Bezkontextové gramatiky

while $x \leq y$ **do begin** $x := (x + 1)\,;\; y := (y - 1)$ **end**

Bezkontextové gramatiky

while $x \leq y$ **do begin** $x := (x + 1)\,;\; y := (y - 1)$ **end**

$\langle \text{STMT} \rangle$

Bezkontextové gramatiky

while $x \leq y$ **do begin** $x := (x + 1)\,;\; y := (y - 1)$ **end**

$\langle \text{STMT} \rangle$
$\langle \text{WHILE-STMT} \rangle$

Bezkontextové gramatiky

while $x \leq y$ **do begin** $x := (x + 1)\,;\; y := (y - 1)$ **end**

$\langle \text{STMT} \rangle$
$\langle \text{WHILE-STMT} \rangle$
while $\langle \text{BOOL-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$

Bezkontextové gramatiky

while $x \leq y$ **do begin** $x := (x + 1)\,;\; y := (y - 1)$ **end**

$\langle \text{STMT} \rangle$
$\langle \text{WHILE-STMT} \rangle$
while $\langle \text{BOOL-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{ARITH-EXPR} \rangle \langle \text{COMPARE-OP} \rangle \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$

Bezkontextové gramatiky

while $x \leq y$ **do begin** $x := (x + 1)\,;\; y := (y - 1)$ **end**

$\langle \text{STMT} \rangle$
$\langle \text{WHILE-STMT} \rangle$
while $\langle \text{BOOL-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{ARITH-EXPR} \rangle \langle \text{COMPARE-OP} \rangle \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{VAR} \rangle \langle \text{COMPARE-OP} \rangle \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$

Bezkontextové gramatiky

while $x \leq y$ **do begin** $x := (x + 1)\,;\; y := (y - 1)$ **end**

$\langle \text{STMT} \rangle$
$\langle \text{WHILE-STMT} \rangle$
while $\langle \text{BOOL-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{ARITH-EXPR} \rangle \langle \text{COMPARE-OP} \rangle \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{VAR} \rangle \langle \text{COMPARE-OP} \rangle \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{VAR} \rangle \leq \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$

Bezkontextové gramatiky

while $x \leq y$ **do begin** $x := (x + 1)\,;\; y := (y - 1)$ **end**

$\langle \text{STMT} \rangle$
$\langle \text{WHILE-STMT} \rangle$
while $\langle \text{BOOL-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{ARITH-EXPR} \rangle \langle \text{COMPARE-OP} \rangle \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{VAR} \rangle \langle \text{COMPARE-OP} \rangle \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{VAR} \rangle \leq \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{VAR} \rangle \leq \langle \text{VAR} \rangle$ **do** $\langle \text{STMT} \rangle$

Bezkontextové gramatiky

while $x \leq y$ **do begin** $x := (x + 1)\,;\; y := (y - 1)$ **end**

$\langle \text{STMT} \rangle$
$\langle \text{WHILE-STMT} \rangle$
while $\langle \text{BOOL-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{ARITH-EXPR} \rangle \langle \text{COMPARE-OP} \rangle \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{VAR} \rangle \langle \text{COMPARE-OP} \rangle \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{VAR} \rangle \leq \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{VAR} \rangle \leq \langle \text{VAR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $x \leq \langle \text{VAR} \rangle$ **do** $\langle \text{STMT} \rangle$

Bezkontextové gramatiky

while $x \leq y$ **do begin** $x := (x + 1)\,;\; y := (y - 1)$ **end**

$\langle \text{STMT} \rangle$
$\langle \text{WHILE-STMT} \rangle$
while $\langle \text{BOOL-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{ARITH-EXPR} \rangle \langle \text{COMPARE-OP} \rangle \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{VAR} \rangle \langle \text{COMPARE-OP} \rangle \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{VAR} \rangle \leq \langle \text{ARITH-EXPR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $\langle \text{VAR} \rangle \leq \langle \text{VAR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $x \leq \langle \text{VAR} \rangle$ **do** $\langle \text{STMT} \rangle$
while $x \leq y$ **do** $\langle \text{STMT} \rangle$

Bezkontextové gramatiky

while $x \leq y$ **do begin** $x := (x + 1)\,;\; y := (y - 1)$ **end**

$\langle$STMT$\rangle$
$\langle$WHILE-STMT$\rangle$
while $\langle$BOOL-EXPR$\rangle$ **do** $\langle$STMT$\rangle$
while $\langle$ARITH-EXPR$\rangle\langle$COMPARE-OP$\rangle\langle$ARITH-EXPR$\rangle$ **do** $\langle$STMT$\rangle$
while $\langle$VAR$\rangle\langle$COMPARE-OP$\rangle\langle$ARITH-EXPR$\rangle$ **do** $\langle$STMT$\rangle$
while $\langle$VAR$\rangle \leq \langle$ARITH-EXPR$\rangle$ **do** $\langle$STMT$\rangle$
while $\langle$VAR$\rangle \leq \langle$VAR$\rangle$ **do** $\langle$STMT$\rangle$
while $x \leq \langle$VAR$\rangle$ **do** $\langle$STMT$\rangle$
while $x \leq y$ **do** $\langle$STMT$\rangle$
while $x \leq y$ **do** $\langle$BLOCK-STMT$\rangle$

Bezkontextové gramatiky

while $x \leq y$ **do begin** $x := (x + 1)\,;\; y := (y - 1)$ **end**

$\langle$STMT$\rangle$
$\langle$WHILE-STMT$\rangle$
while $\langle$BOOL-EXPR$\rangle$ **do** $\langle$STMT$\rangle$
while $\langle$ARITH-EXPR$\rangle\langle$COMPARE-OP$\rangle\langle$ARITH-EXPR$\rangle$ **do** $\langle$STMT$\rangle$
while $\langle$VAR$\rangle\langle$COMPARE-OP$\rangle\langle$ARITH-EXPR$\rangle$ **do** $\langle$STMT$\rangle$
while $\langle$VAR$\rangle \leq \langle$ARITH-EXPR$\rangle$ **do** $\langle$STMT$\rangle$
while $\langle$VAR$\rangle \leq \langle$VAR$\rangle$ **do** $\langle$STMT$\rangle$
while $x \leq \langle$VAR$\rangle$ **do** $\langle$STMT$\rangle$
while $x \leq y$ **do** $\langle$STMT$\rangle$
while $x \leq y$ **do** $\langle$BLOCK-STMT$\rangle$
while $x \leq y$ **do begin** $\langle$STMT-LIST$\rangle$ **end**

Bezkontextové gramatiky

while $x \leq y$ **do begin** $x := (x + 1)\,;\; y := (y - 1)$ **end**

$\langle$STMT$\rangle$
$\langle$WHILE-STMT$\rangle$
while $\langle$BOOL-EXPR$\rangle$ **do** $\langle$STMT$\rangle$
while $\langle$ARITH-EXPR$\rangle\langle$COMPARE-OP$\rangle\langle$ARITH-EXPR$\rangle$ **do** $\langle$STMT$\rangle$
while $\langle$VAR$\rangle\langle$COMPARE-OP$\rangle\langle$ARITH-EXPR$\rangle$ **do** $\langle$STMT$\rangle$
while $\langle$VAR$\rangle \leq \langle$ARITH-EXPR$\rangle$ **do** $\langle$STMT$\rangle$
while $\langle$VAR$\rangle \leq \langle$VAR$\rangle$ **do** $\langle$STMT$\rangle$
while $x \leq \langle$VAR$\rangle$ **do** $\langle$STMT$\rangle$
while $x \leq y$ **do** $\langle$STMT$\rangle$
while $x \leq y$ **do** $\langle$BLOCK-STMT$\rangle$
while $x \leq y$ **do begin** $\langle$STMT-LIST$\rangle$ **end**
...

Bezkontextové gramatiky

Formálně je **bezkontextová gramatika** definována jako čtveřice

$$G = (\Pi, \Sigma, S, P)$$

kde:

- Π je konečná množina **neterminálních symbolů** (**neterminálů**)
- Σ je konečná množina **terminálních symbolů** (**terminálů**), přičemž $\Pi \cap \Sigma = \emptyset$
- $S \in \Pi$ je **počáteční neterminál**
- $P \subseteq \Pi \times (\Pi \cup \Sigma)^*$ je konečná množina **přepisovacích pravidel**

Bezkontextové gramatiky

Poznámky:

- Pro označení neterminálních symbolů budeme používat velká písmena A, B, C, ...
- Pro označení terminálních symbolů budeme používat malá písmena a, b, c, ... nebo číslice 0, 1, 2, ...
- Pro označení řetězců z $(\Pi \cup \Sigma)^*$ budeme používat malá písmena řecké abecedy α, β, γ, ...
- Místo zápisu (A, α) budeme pro pravidla používat zápis

$$A \to \alpha$$

A – levá strana pravidla
α – pravá strana pravidla

Bezkontextové gramatiky

Příklad: Gramatika $G = (\Pi, \Sigma, S, P)$, kde

- $\Pi = \{A, B, C\}$
- $\Sigma = \{a, b\}$
- $S = A$
- P obsahuje pravidla

$$A \to aBBb$$
$$A \to AaA$$
$$B \to \varepsilon$$
$$B \to bCA$$
$$C \to AB$$
$$C \to a$$
$$C \to b$$

Bezkontextové gramatiky

Poznámka: Pokud máme více pravidel se stejnou levou stranou, jako třeba

$$A \to \alpha_1 \qquad A \to \alpha_2 \qquad A \to \alpha_3$$

můžeme je stručněji zapsat jako

$$A \to \alpha_1 \mid \alpha_2 \mid \alpha_3$$

Například pravidla dříve uvedené gramatiky můžeme zapsat jako

$$\begin{aligned} A &\to aBBb \mid AaA \\ B &\to \varepsilon \mid bCA \\ C &\to AB \mid a \mid b \end{aligned}$$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo *abbabb* je možné v gramatice G vygenerovat následujícím způsobem:

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

A

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$\underline{A} \longrightarrow aBBb \mid AaA$$
$$B \longrightarrow \varepsilon \mid bCA$$
$$C \longrightarrow AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$\underline{A}$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$\underline{A} \longrightarrow \underline{aBBb} \mid AaA$$
$$B \longrightarrow \varepsilon \mid bCA$$
$$C \longrightarrow AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$\underline{A} \Rightarrow \underline{aBBb}$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \longrightarrow aBBb \mid AaA$$
$$B \longrightarrow \varepsilon \mid bCA$$
$$C \longrightarrow AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow a\underline{B}Bb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$\begin{aligned} A &\to aBBb \mid AaA \\ B &\to \varepsilon \mid bCA \\ C &\to AB \mid a \mid b \end{aligned}$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$\underline{A} \to \underline{aBBb} \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abC\underline{A}Bb \Rightarrow abC\underline{aBBb}Bb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaB\underline{B}bBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \longrightarrow aBBb \mid AaA$$
$$B \longrightarrow \varepsilon \mid bCA$$
$$\underline{C} \longrightarrow AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow ab\underline{C}aBbBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \longrightarrow aBBb \mid AaA$$
$$B \longrightarrow \varepsilon \mid bCA$$
$$\underline{C} \longrightarrow AB \mid a \mid \underline{b}$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow ab\underline{C}aBbBb \Rightarrow ab\underline{b}aBbBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abbaBbb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abbaBbb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abbaBbb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abbaBbb \Rightarrow abbabb$

Bezkontextové gramatiky

Gramatiky slouží ke generování slov.

Příklad: $G = (\Pi, \Sigma, A, P)$, kde $\Pi = \{A, B, C\}$, $\Sigma = \{a, b\}$ a P obsahuje pravidla

$$A \to aBBb \mid AaA$$
$$B \to \varepsilon \mid bCA$$
$$C \to AB \mid a \mid b$$

Například slovo $abbabb$ je možné v gramatice G vygenerovat následujícím způsobem:

$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abbaBbb \Rightarrow abbabb$

Bezkontextové gramatiky

Řetězce z $(\Pi \cup \Sigma)^*$ nazýváme **větné formy**.

Na větných formách definujeme relaci $\Rightarrow \subseteq (\Pi \cup \Sigma)^* \times (\Pi \cup \Sigma)^*$ takovou, že

$$\alpha \Rightarrow \alpha'$$

právě když $\alpha = \beta_1 A \beta_2$ a $\alpha' = \beta_1 \gamma \beta_2$ pro nějaká $\beta_1, \beta_2, \gamma \in (\Pi \cup \Sigma)^*$ a $A \in \Pi$, kde $(A \rightarrow \gamma) \in P$.

Příklad: Jestliže $(B \rightarrow bCA) \in P$, pak

$$aCBbA \Rightarrow aCbCAbA$$

Poznámka: Neformálně řečeno zápis $\alpha \Rightarrow \alpha'$ znamená, že z větné formy α je možné jedním krokem odvodit větnou formu α, a to tak, že výskyt nějakého neterminálu A v α nahradíme pravou stranou nějakého pravidla $A \rightarrow \alpha$, kde se A vyskytuje na levé straně.

Bezkontextové gramatiky

Řetězce z $(\Pi \cup \Sigma)^*$ nazýváme **větné formy**.

Na větných formách definujeme relaci $\Rightarrow \subseteq (\Pi \cup \Sigma)^* \times (\Pi \cup \Sigma)^*$ takovou, že

$$\alpha \Rightarrow \alpha'$$

právě když $\alpha = \beta_1 A \beta_2$ a $\alpha' = \beta_1 \gamma \beta_2$ pro nějaká $\beta_1, \beta_2, \gamma \in (\Pi \cup \Sigma)^*$ a $A \in \Pi$, kde $(A \to \gamma) \in P$.

Příklad: Jestliže $(B \to bCA) \in P$, pak

$$aC\underline{B}bA \Rightarrow aC\underline{bCA}bA$$

Poznámka: Neformálně řečeno zápis $\alpha \Rightarrow \alpha'$ znamená, že z větné formy α je možné jedním krokem odvodit větnou formu α, a to tak, že výskyt nějakého neterminálu A v α nahradíme pravou stranou nějakého pravidla $A \to \alpha$, kde se A vyskytuje na levé straně.

Bezkontextové gramatiky

Derivace délky n větné formy α' z větné formy α je posloupnost větných forem

$$\beta_0, \beta_1, \beta_2, \cdots, \beta_n$$

takových, že

- $\alpha = \beta_0$
- $\beta_{i-1} \Rightarrow \beta_i$ pro všechna $i \in \{1, 2, \ldots, n\}$
- $\alpha' = \beta_n$

což můžeme stručněji zapsat jako

$$\alpha = \beta_0 \Rightarrow \beta_1 \Rightarrow \beta_2 \Rightarrow \ldots \Rightarrow \beta_{n-1} \Rightarrow \beta_n = \alpha'$$

Bezkontextové gramatiky

Skutečnost, že pro dané n existuje nějaká derivace délky n větné formy α' z větné formy α, označujeme zápisem

$$\alpha \Rightarrow^n \alpha'$$

Skutečnost, že existuje nějaká derivace (nějaké délky n, kde $n \geq 0$) větné formy α' z větné formy α, označujeme zápisem

$$\alpha \Rightarrow^* \alpha'$$

Poznámka: Relace $\Rightarrow^*$ je reflexivním a tranzitivním uzávěrem relace $\Rightarrow$ (tj. nejmenší reflexivní a tranzitivní relací obsahující relaci $\Rightarrow$).

Bezkontextové gramatiky

Jazyk $L(G)$ generovaný gramatikou $G = (\Pi, \Sigma, S, P)$ je množina všech slov v abecedě Σ, která lze odvodit nějakou derivací z počátečního neterminálu S pomocí pravidel z P, tj.

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

Bezkontextové gramatiky

Příklad: Chceme vytvořit gramatiku generující jazyk

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

Bezkontextové gramatiky

Příklad: Chceme vytvořit gramatiku generující jazyk

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

Gramatika $G = (\Pi, \Sigma, S, P)$, kde $\Pi = \{S\}$, $\Sigma = \{a, b\}$ a P obsahuje

$$S \to aSb \mid \varepsilon$$

Bezkontextové gramatiky

Příklad: Chceme vytvořit gramatiku generující jazyk

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

Gramatika $G = (\Pi, \Sigma, S, P)$, kde $\Pi = \{S\}$, $\Sigma = \{a, b\}$ a P obsahuje

$$S \to aSb \mid \varepsilon$$

$S \Rightarrow \varepsilon$
$S \Rightarrow aSb \Rightarrow ab$
$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$
$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$
$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaaaSbbbb \Rightarrow aaaabbbb$
$\ldots$

Bezkontextové gramatiky

Příklad: Chceme vytvořit gramatiku generující jazyk tvořený všemi palindromy nad abecedou $\{a, b\}$, tj.

$$L = \{w \in \{a, b\}^* \mid w = w^R\}$$

Poznámka: w^R označuje tzv. **zrcadlový obraz** slova w, tj. slovo w zapsané pozpátku.

Bezkontextové gramatiky

Příklad: Chceme vytvořit gramatiku generující jazyk tvořený všemi palindromy nad abecedou $\{a, b\}$, tj.

$$L = \{w \in \{a, b\}^* \mid w = w^R\}$$

Poznámka: w^R označuje tzv. **zrcadlový obraz** slova w, tj. slovo w zapsané pozpátku.

Řešení:

$$S \to aSa \mid bSb \mid a \mid b \mid \varepsilon$$

Bezkontextové gramatiky

Příklad: Chceme vytvořit gramatiku generující jazyk tvořený všemi palindromy nad abecedou $\{a, b\}$, tj.

$$L = \{w \in \{a, b\}^* \mid w = w^R\}$$

Poznámka: w^R označuje tzv. **zrcadlový obraz** slova w, tj. slovo w zapsané pozpátku.

Řešení:

$$S \to aSa \mid bSb \mid a \mid b \mid \varepsilon$$

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abaSaba \Rightarrow abaaaba$$

Bezkontextové gramatiky

Příklad: Chceme vytvořit gramatiku generující jazyk L tvořený všemi dobře uzávorkovanými sekvencemi symbolů '(' a ')'.

Například $(()())(()) \in L$, ale $)()) \notin L$.

Bezkontextové gramatiky

Příklad: Chceme vytvořit gramatiku generující jazyk L tvořený všemi dobře uzávorkovanými sekvencemi symbolů '(' a ')'.

Například $(()())(()) \in L$, ale $)()) \notin L$.

Řešení:

$$S \to \varepsilon \mid (S) \mid SS$$

Bezkontextové gramatiky

Příklad: Chceme vytvořit gramatiku generující jazyk L tvořený všemi dobře uzávorkovanými sekvencemi symbolů '(' a ')'.

Například $(()())(()) \in L$, ale $)()) \notin L$.

Řešení:

$$S \to \varepsilon \mid (S) \mid SS$$

$S \Rightarrow SS \Rightarrow (S)S \Rightarrow (S)(S) \Rightarrow (SS)(S) \Rightarrow ((S)S)(S) \Rightarrow$
$(()S)(S) \Rightarrow (()(S))(S) \Rightarrow (()())(S) \Rightarrow (()())((S)) \Rightarrow$
$(()())(())$

Bezkontextové gramatiky

Příklad: Chceme vytvořit gramatiku generující jazyk L tvořený všemi dobře vytvořenými aritmetickými výrazy, kde operandy jsou vždy tvaru 'a', a kde jako operátory můžeme používat symboly $+$ a $*$.

Například $(a + a) * a + (a * a) \in L$.

Bezkontextové gramatiky

Příklad: Chceme vytvořit gramatiku generující jazyk L tvořený všemi dobře vytvořenými aritmetickými výrazy, kde operandy jsou vždy tvaru 'a', a kde jako operátory můžeme používat symboly $+$ a $*$.

Například $(a + a) * a + (a * a) \in L$.

Řešení:

$$E \rightarrow a \mid E + E \mid E * E \mid (E)$$

Bezkontextové gramatiky

Příklad: Chceme vytvořit gramatiku generující jazyk L tvořený všemi dobře vytvořenými aritmetickými výrazy, kde operandy jsou vždy tvaru 'a', a kde jako operátory můžeme používat symboly $+$ a $*$.

Například $(a + a) * a + (a * a) \in L$.

Řešení:

$$E \rightarrow a \mid E + E \mid E * E \mid (E)$$

$$E \Rightarrow E+E \Rightarrow E*E+E \Rightarrow (E)*E+E \Rightarrow (a+a)*E+E \Rightarrow (a+a)*a+E \Rightarrow$$
$$(a+a)*a+(E) \Rightarrow (a+a)*a+(E*E) \Rightarrow (a+a)*a+(a*E) \Rightarrow (a+a)*a+(a*a)$$

Derivační strom

$A \rightarrow aBBb \mid AaA$
$B \rightarrow \varepsilon \mid bCA$
$C \rightarrow AB \mid a \mid b$

Derivační strom

A

$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$

A

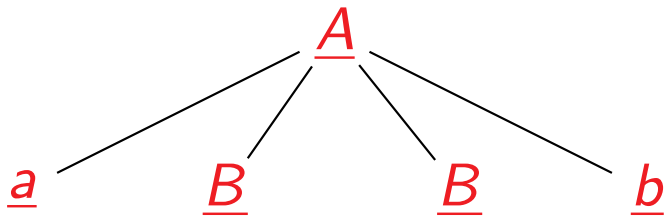
Derivační strom

$\underline{A}$

$\underline{A} \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$

$\underline{A}$

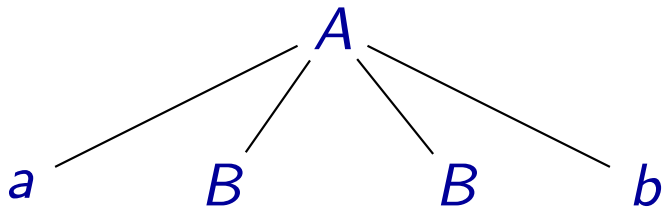
Derivační strom



$\underline{A} \rightarrow \underline{aBBb} \mid AaA$
$B \rightarrow \varepsilon \mid bCA$
$C \rightarrow AB \mid a \mid b$

$\underline{A} \Rightarrow \underline{aBBb}$

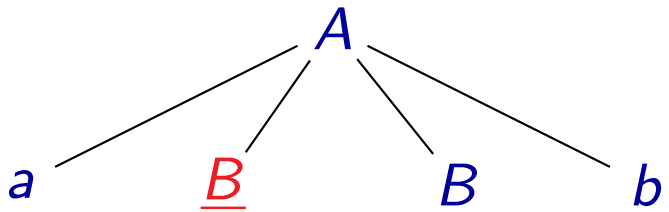
Derivační strom



$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$

$A \Rightarrow aBBb$

Derivační strom

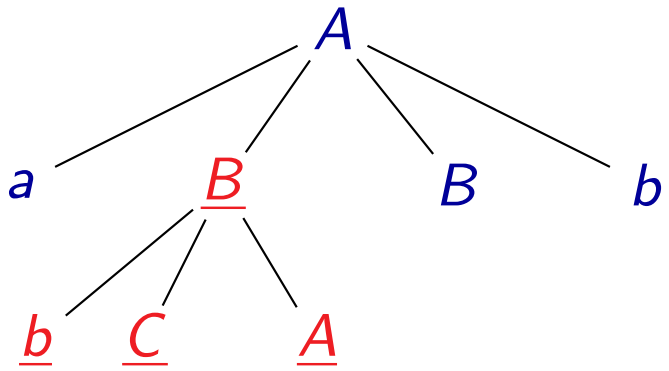


$A \to aBBb \mid AaA$
$\underline{B} \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$

$A \Rightarrow a\underline{B}Bb$

Derivační strom

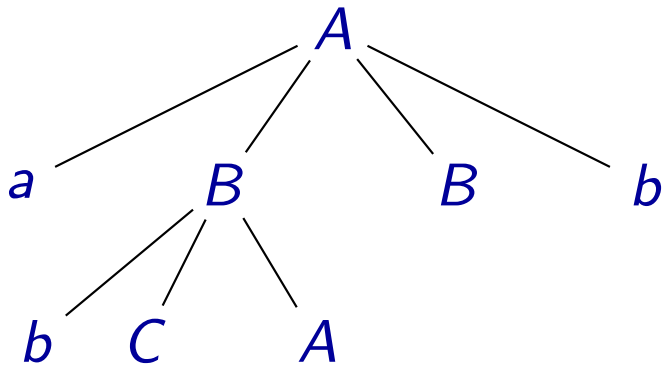
$A \to aBBb \mid AaA$
$\underline{B} \to \varepsilon \mid \underline{bCA}$
$C \to AB \mid a \mid b$



$A \Rightarrow a\underline{B}Bb \Rightarrow a\underline{bCA}Bb$

Derivační strom

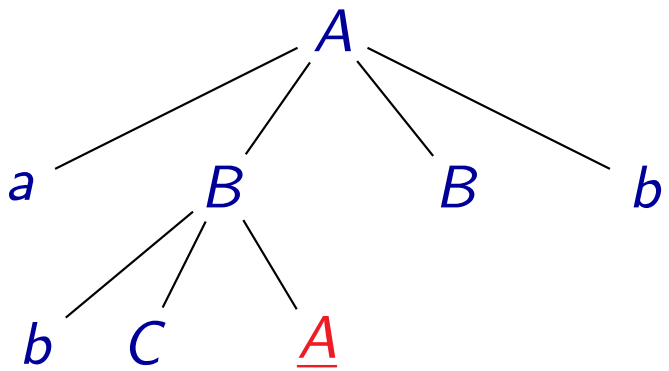
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb$

Derivační strom

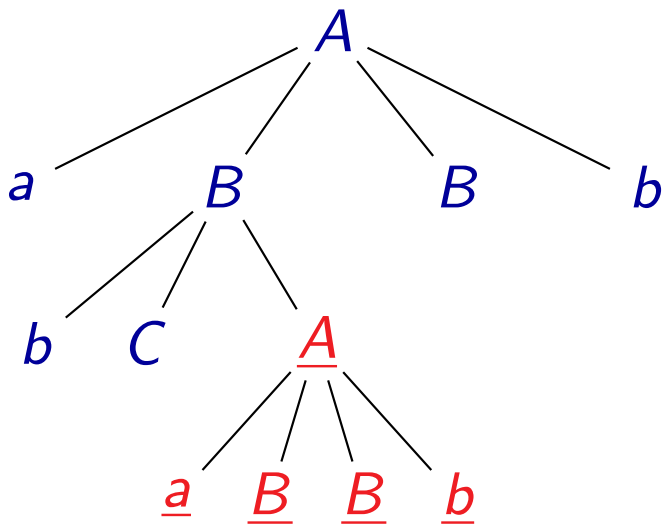
$\underline{A} \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abC\underline{A}Bb$

Derivační strom

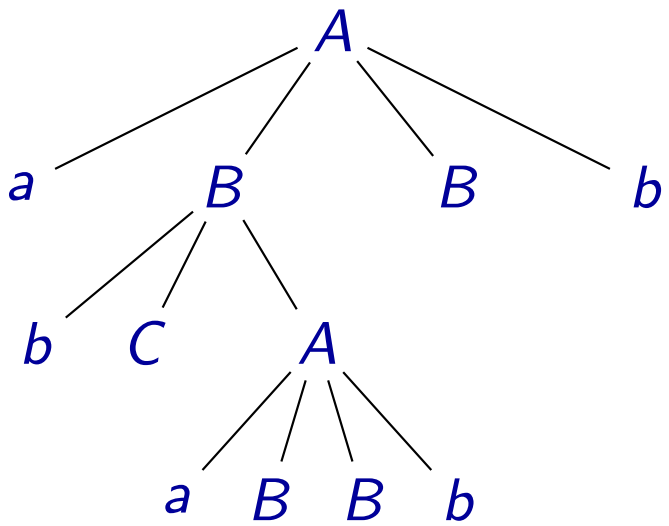
$\underline{A} \to \underline{aBBb} \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abC\underline{A}Bb \Rightarrow abC\underline{aBBb}Bb$

Derivační strom

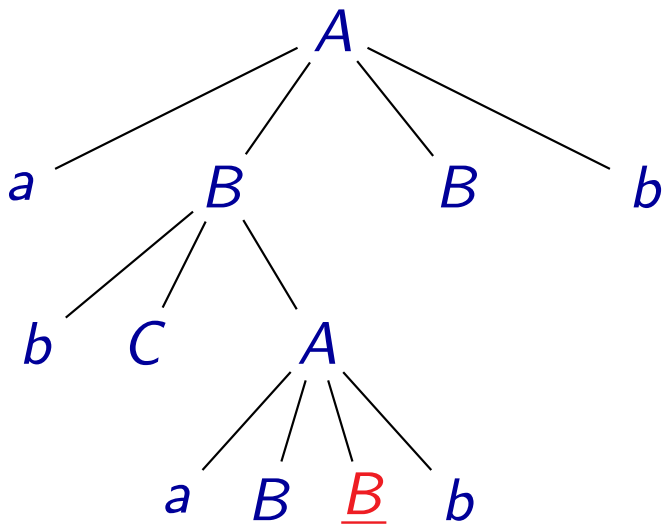
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb$

Derivační strom

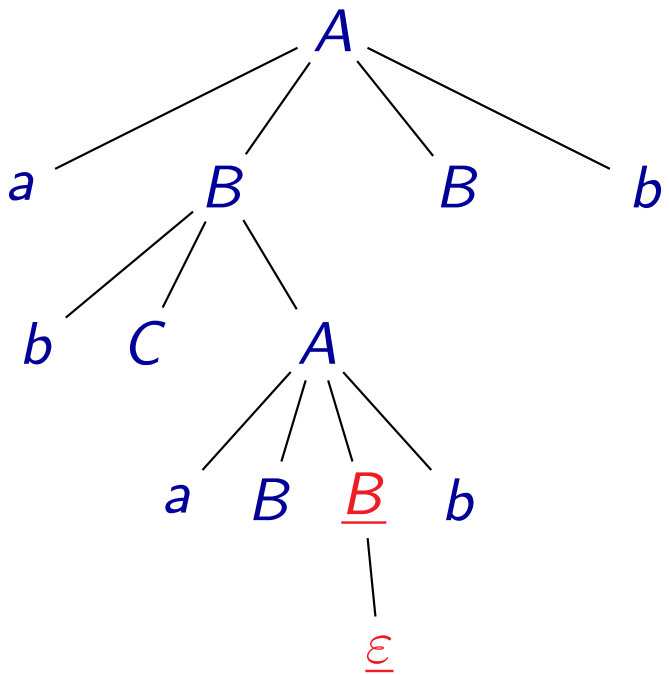
$A \to aBBb \mid AaA$
$\underline{B} \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaB\underline{B}bBb$

Derivační strom

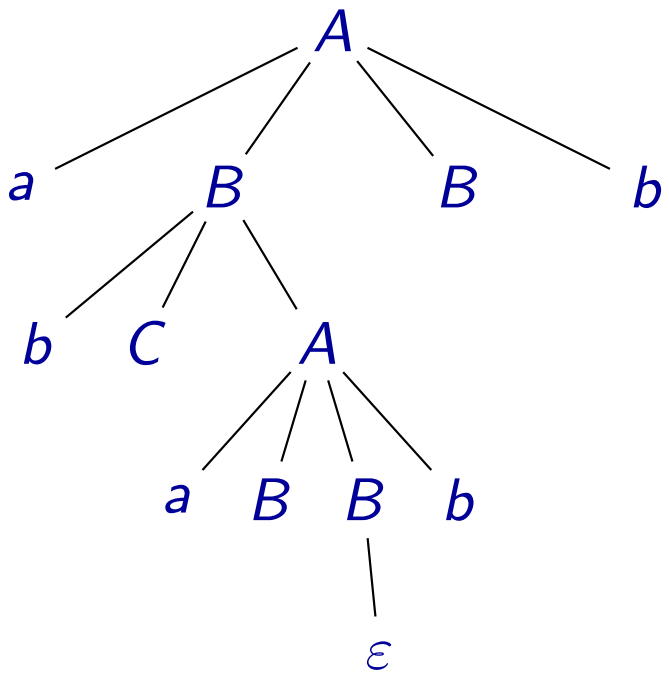
$A \to aBBb \mid AaA$
$\underline{B} \to \underline{\varepsilon} \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaB\underline{B}bBb \Rightarrow abCaBbBb$

Derivační strom

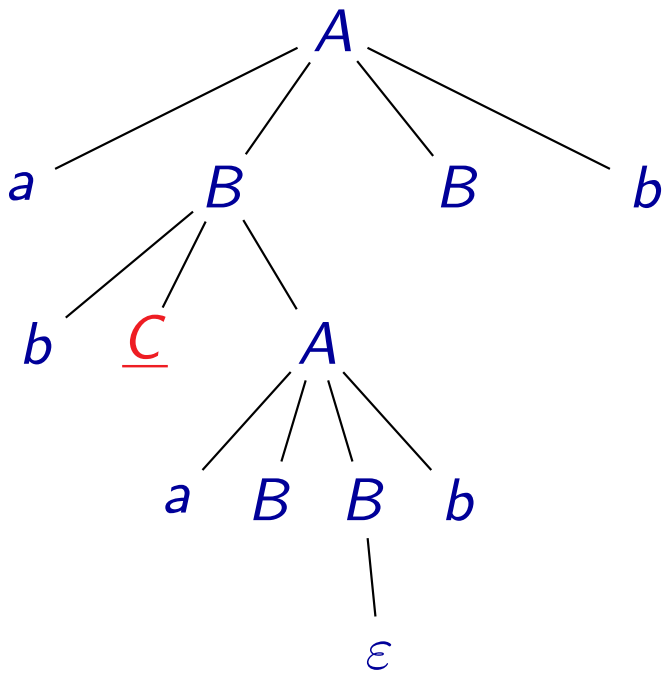
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb$

Derivační strom

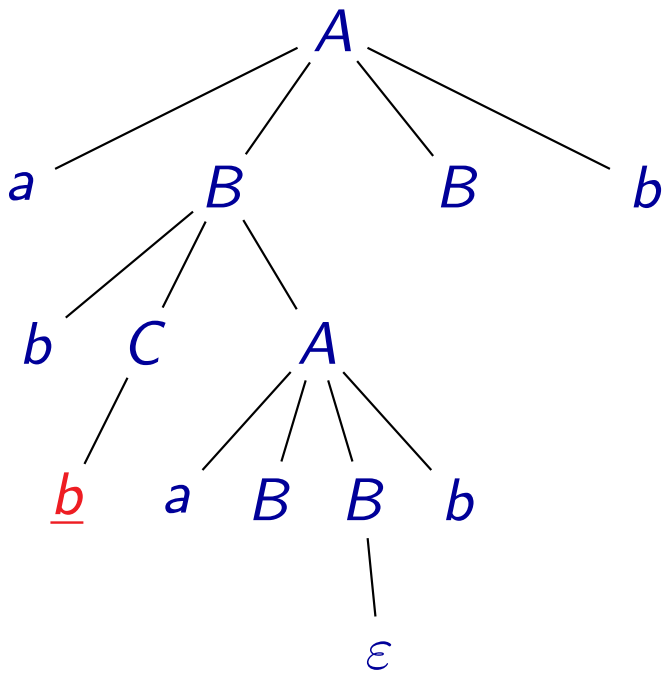
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$\underline{C} \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow ab\underline{C}aBbBb$

Derivační strom

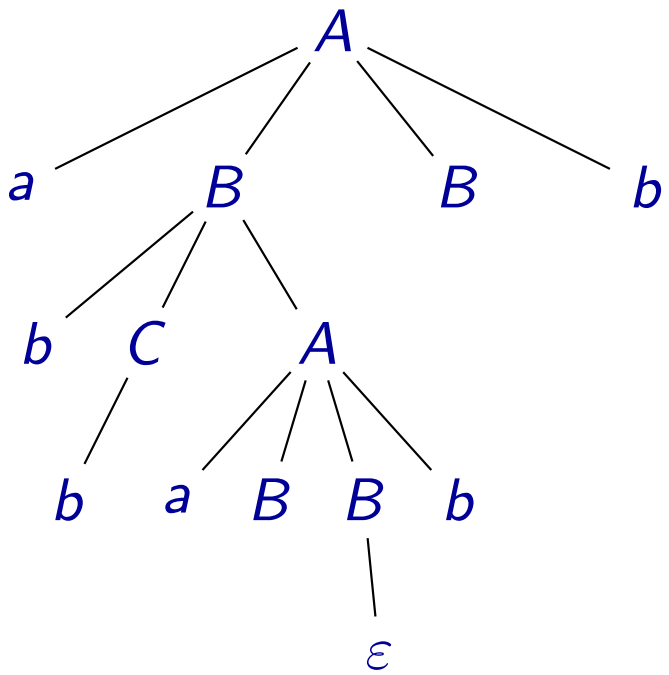
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$\underline{C} \to AB \mid a \mid \underline{b}$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow ab\underline{C}aBbBb \Rightarrow ab\underline{b}aBbBb$

Derivační strom

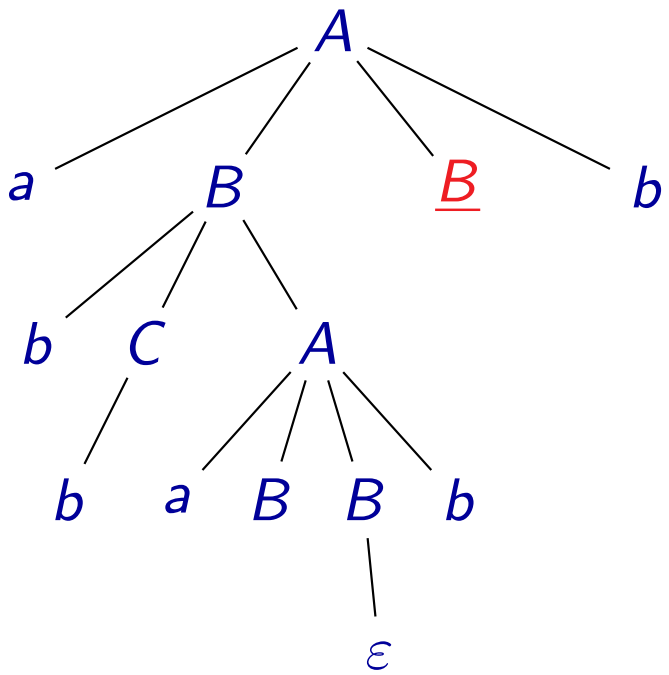
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb$

Derivační strom

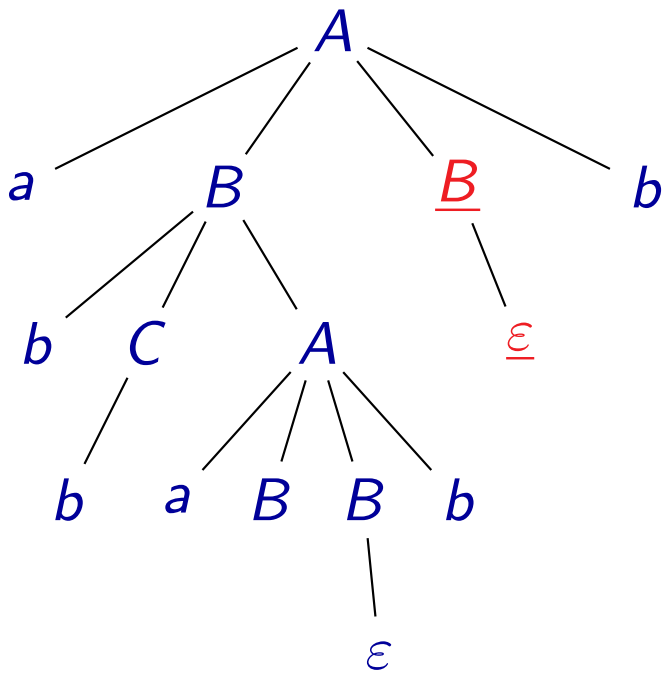
$A \to aBBb \mid AaA$
$\underline{B} \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBb\underline{B}b$$

Derivační strom

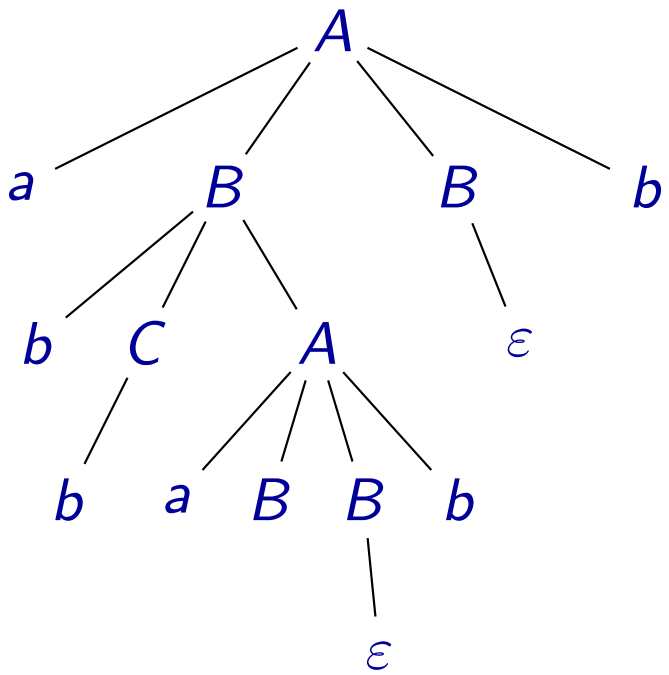
$A \to aBBb \mid AaA$
$\underline{B} \to \underline{\varepsilon} \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBb\underline{B}b \Rightarrow abbaBbb$

Derivační strom

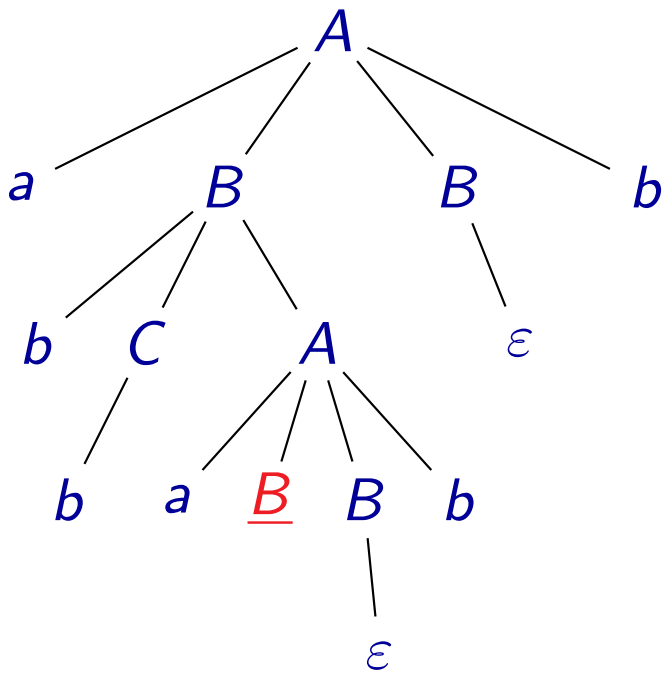
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abbaBbb$

Derivační strom

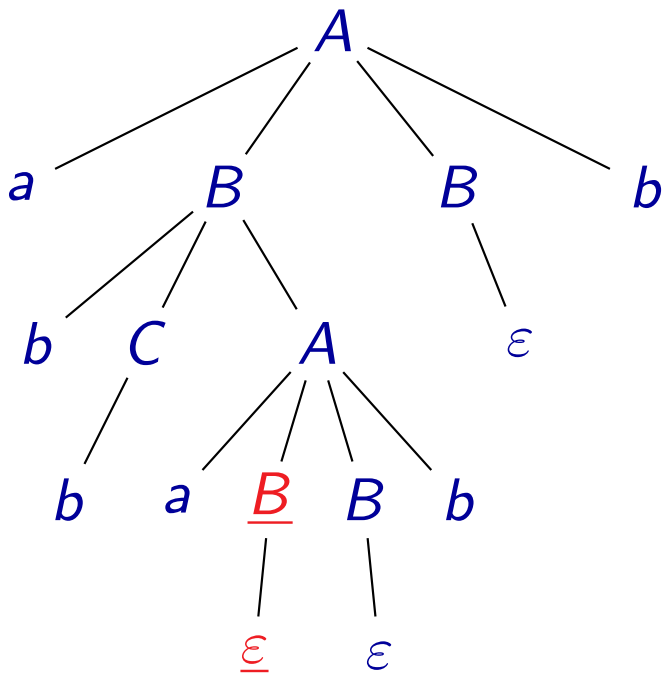
$A \to aBBb \mid AaA$
$\underline{B} \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abba\underline{B}bb$

Derivační strom

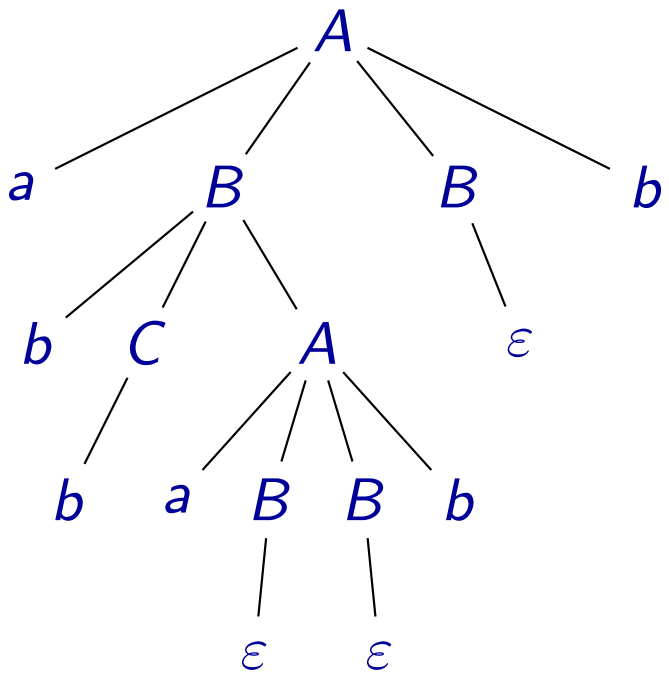
$A \to aBBb \mid AaA$
$\underline{B} \to \underline{\varepsilon} \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abba\underline{B}bb \Rightarrow abbabb$

Derivační strom

$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abbaBbb \Rightarrow abbabb$

Derivační strom

Každé derivaci odpovídá nějaký **derivační strom:**

- Vrcholy stromu jsou ohodnoceny terminály a neterminály.
- Kořen stromu je ohodnocen počátečním neterminálem.
- Listy stromu jsou ohodnoceny terminály nebo symboly ε.
- Ostatní vrcholy stromu jsou ohodnoceny neterminály.
- Pokud je vrchol ohodnocen neterminálem A, pak jeho potomci jsou ohodnoceni symboly pravé strany nějakého přepisovacího pravidla $A \to \alpha$.

Levá a pravá derivace

$$E \to E + E \mid E * E \mid (E) \mid a$$

Levá derivace je derivace, ve které v každém kroku nahrazujeme vždy nejlevější neterminál.

$$\underline{E} \Rightarrow \underline{E} + E \Rightarrow \underline{E} * E + E \Rightarrow a * \underline{E} + E \Rightarrow a * a + \underline{E} \Rightarrow a * a + a$$

Pravá derivace je derivace, ve které v každém kroku nahrazujeme vždy nejpravější neterminál.

$$\underline{E} \Rightarrow E + \underline{E} \Rightarrow \underline{E} + a \Rightarrow E * \underline{E} + a \Rightarrow \underline{E} * a + a \Rightarrow a * a + a$$

Derivace však nemusí být ani levá ani pravá:

$$\underline{E} \Rightarrow \underline{E} + E \Rightarrow E * \underline{E} + E \Rightarrow E * a + \underline{E} \Rightarrow \underline{E} * a + a \Rightarrow a * a + a$$

Levá a pravá derivace

- Jednomu derivačnímu stromu může odpovídat více různých derivací.
- Každému derivačnímu stromu odpovídá právě jedna levá a právě jedna pravá derivace.

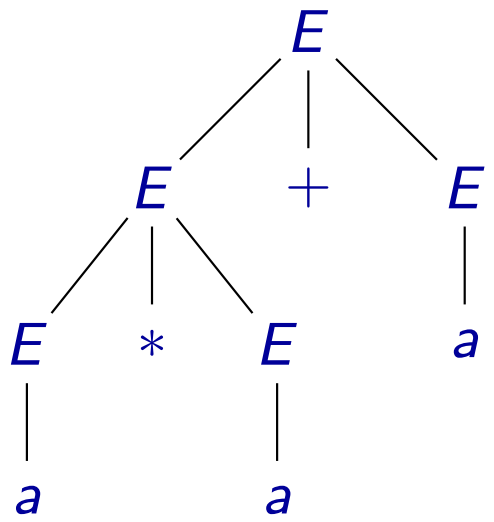
Nejednoznačné gramatiky

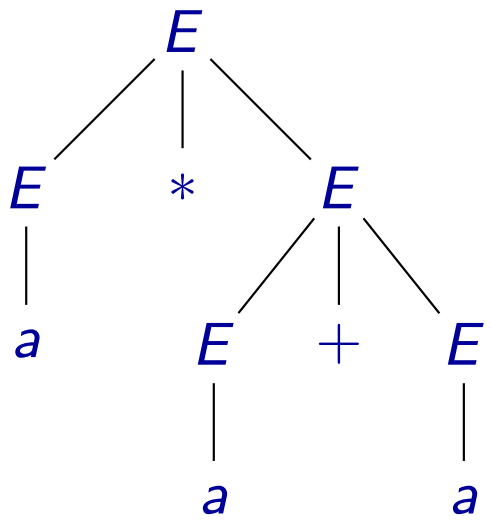
Gramatika G je **nejednoznačná**, jestliže existuje nějaké slovo $w \in L(G)$, kterému přísluší dva různé derivační stromy, resp. dvě různé levé či dvě různé pravé derivace.

Příklad:

$E \Rightarrow E + E \Rightarrow E * E + E \Rightarrow a * E + E \Rightarrow a * a + E \Rightarrow a * a + a$

$E \Rightarrow E * E \Rightarrow E * E + E \Rightarrow a * E + E \Rightarrow a * a + E \Rightarrow a * a + a$





Nejednoznačné gramatiky

Někdy je možné nejednoznačnou gramatiku nahradit gramatikou, která generuje tentýž jazyk, ale není nejednoznačná.

Příklad: Gramatiku

$$E \to E + E \mid E * E \mid (E) \mid a$$

můžeme nahradit ekvivalentní gramatikou

$$\begin{aligned} E &\to T \mid T + E \\ T &\to F \mid F * T \\ F &\to a \mid (E) \end{aligned}$$

Poznámka: Pokud se nejednoznačná gramatika žádnou ekvivalentní jednoznačnou gramatikou nahradit nedá, říkáme, že je **podstatně nejednoznačná**.

Ekvivalence gramatik

Gramatiky G_1 a G_2 jsou **ekvivalentní**, jestliže generují tentýž jazyk, tj. jestliže $L(G_1) = L(G_2)$.

Poznámka: Problém ekvivalence bezkontextových gramatik je algoritmicky nerozhodnutelný. Dá se dokázat, že není možné vytvořit algoritmus, který by pro libovolné dvě bezkontextové gramatiky rozhodl, zda jsou ekvivalentní či ne.

Dokonce je algoritmicky nerozhodnutelný i problém, zda gramatika generuje jazyk Σ^*.

Bezkontextové jazyky

Definice

Jazyk L je **bezkontextový**, jestliže existuje bezkontextová gramatika G taková, že $L = L(G)$.

Třída bezkontextových jazyků je uzavřená vůči:

- zřetězení
- sjednocení
- iteraci

Třída bezkontextových jazyků však není uzavřená vůči:

- doplňku
- průniku

Bezkontextové jazyky

Máme dány gramatiky $G_1 = (\Pi_1, \Sigma, S_1, P_1)$ a $G_2 = (\Pi_2, \Sigma, S_2, P_2)$, přičemž můžeme předpokládat, že $\Pi_1 \cap \Pi_2 = \emptyset$ a $S \notin \Pi_1 \cup \Pi_2$.

- Gramatika G taková, že $L(G) = L(G_1)L(G_2)$:

$$G = (\Pi_1 \cup \Pi_2 \cup \{S\}, \Sigma, S, P_1 \cup P_2 \cup \{S \to S_1 S_2\})$$

- Gramatika G taková, že $L(G) = L(G_1) \cup L(G_2)$:

$$G = (\Pi_1 \cup \Pi_2 \cup \{S\}, \Sigma, S, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\})$$

- Gramatika G taková, že $L(G) = L(G_1)^*$:

$$G = (\Pi_1 \cup \{S\}, \Sigma, S, P_1 \cup \{S \to \varepsilon, S \to S_1 S\})$$

Regulární gramatiky

Definice

Gramatika G je **regulární**, jestliže všechna její pravidla jsou tvaru:

- $X \to wY$, kde $X, Y \in \Pi$, $w \in \Sigma^*$, nebo
- $X \to w$, kde $X \in \Pi$, $w \in \Sigma^*$.

Příklad:

$$S \to abbA \mid bB \mid A \mid \varepsilon$$
$$A \to aA \mid babS \mid aaB$$
$$B \to bbB \mid abA \mid b$$

Regulární gramatiky generují právě třídu regulárních jazyků, tj.:

- Jazyk generovaný regulární gramatikou je vždy regulární.
- Každý regulární jazyk je generován nějakou regulární gramatikou.

Zásobníkové automaty

Zásobníkový automat

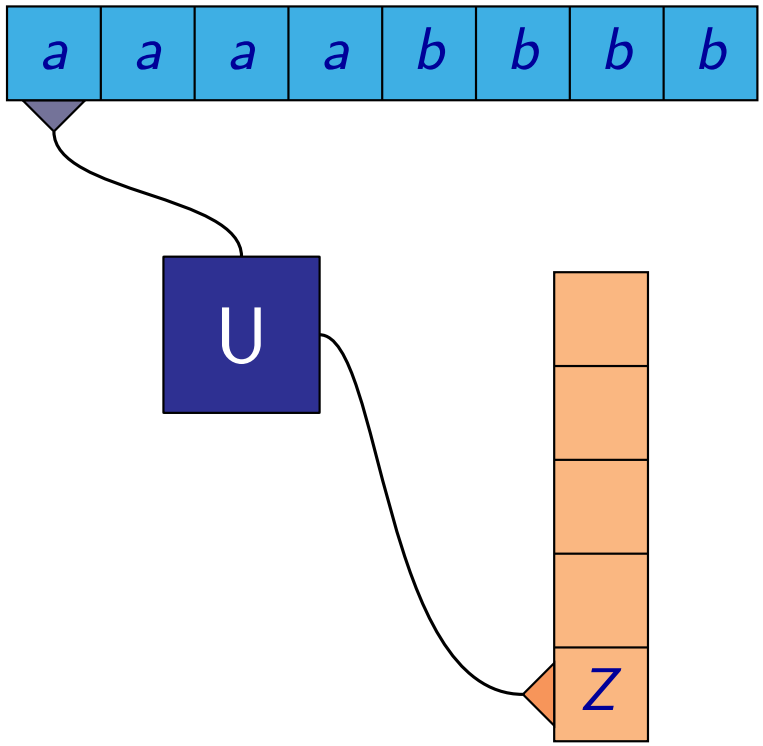
- Chtěli bychom rozpoznávat jazyk $L = \{a^i b^i \mid i \geq 1\}$
- Snažíme se navrhnout zařízení (podobné konečným automatům), které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.
- Při čtení a-ček si musíme pamatovat jejich počet, ať víme, kolik musí následovat b-ček

Zásobníkový automat

- Chtěli bychom rozpoznávat jazyk $L = \{a^i b^i \mid i \geq 1\}$
- Snažíme se navrhnout zařízení (podobné konečným automatům), které přečte slovo, a sdělí nám, zda toto slovo patří do jazyka L či ne.
- Při čtení a-ček si musíme pamatovat jejich počet, ať víme, kolik musí následovat b-ček
- Můžeme využít paměť typu zásobník
- Každé přečtené a si na zásobník zapíšeme, za každé přečtené b jeden symbol ze zásobníku odstraníme
- Pokud bude zásobník prázdný a podaří se přečíst celé slovo, tak patří do jazyka

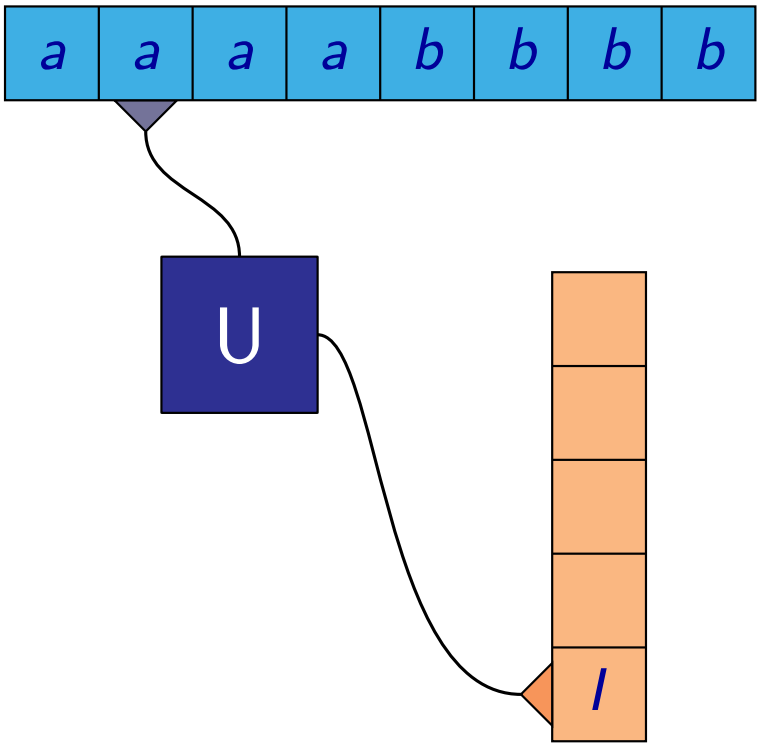
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



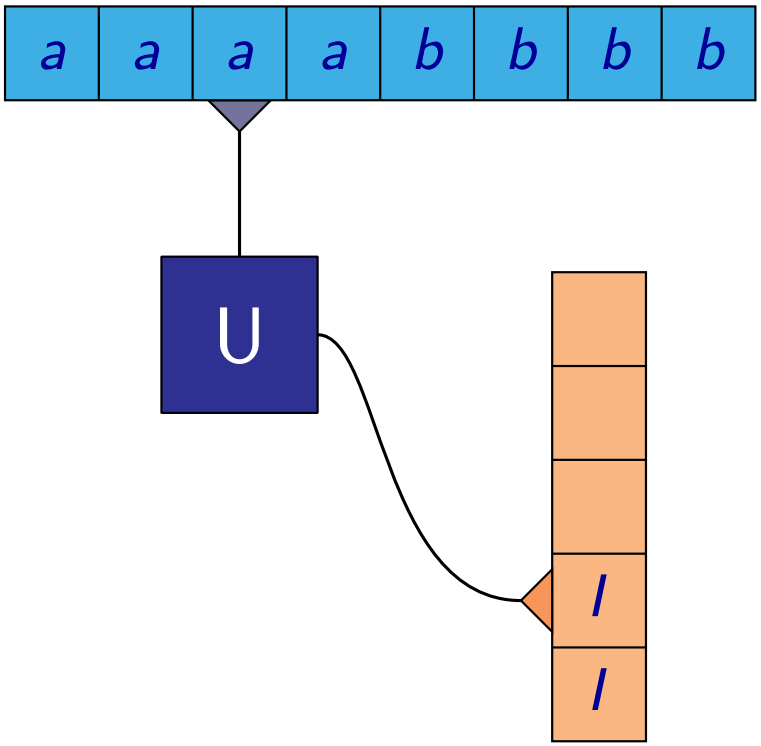
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



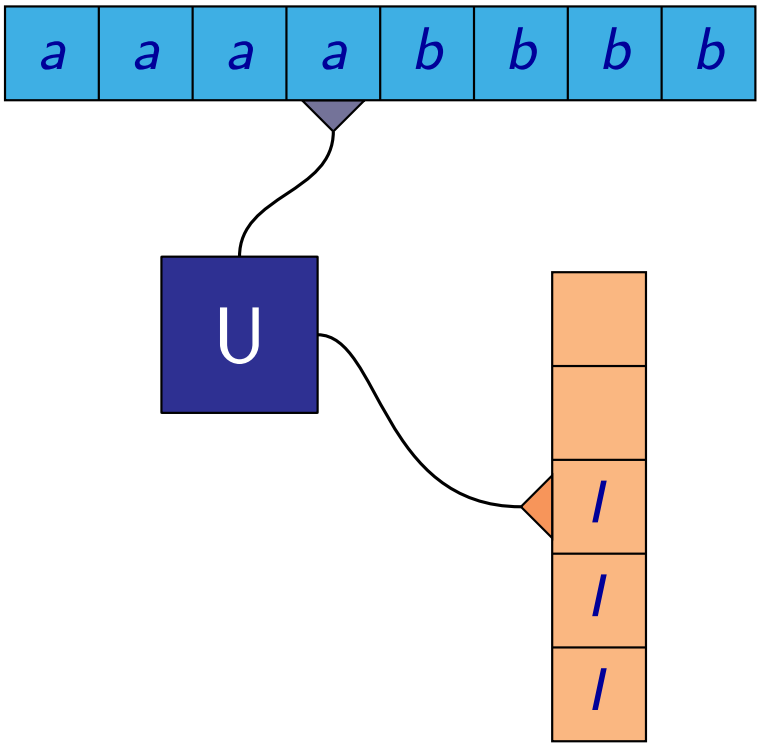
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



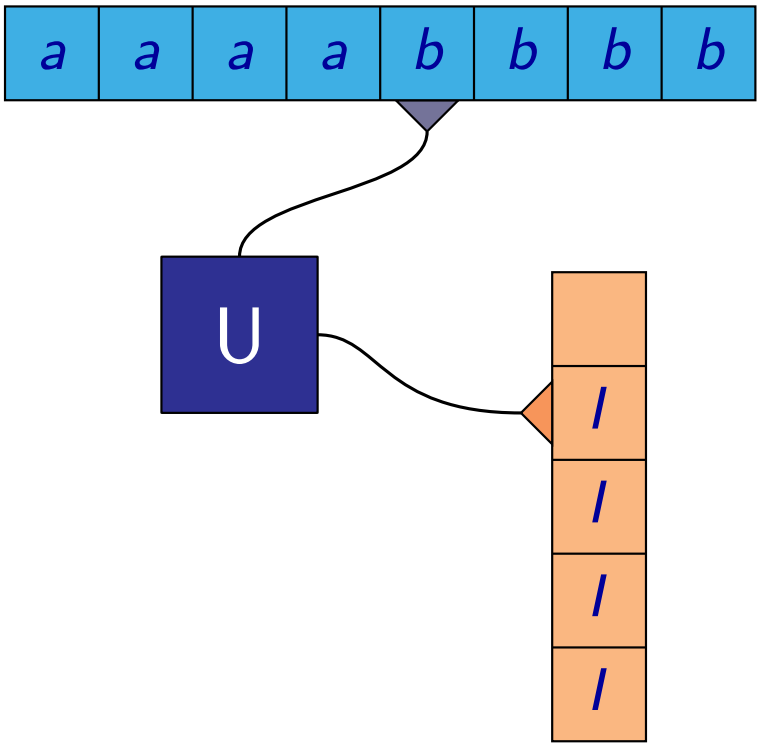
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



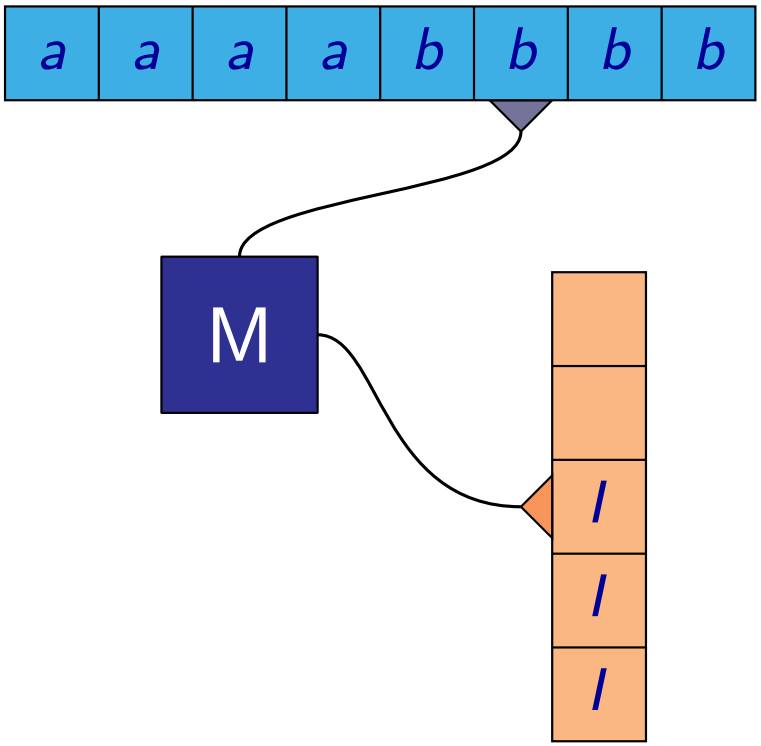
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



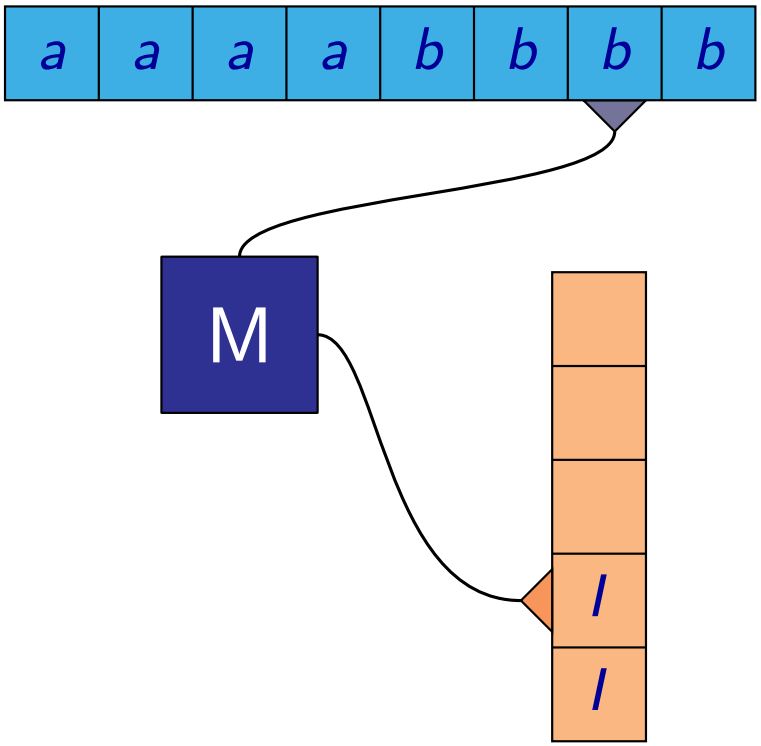
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



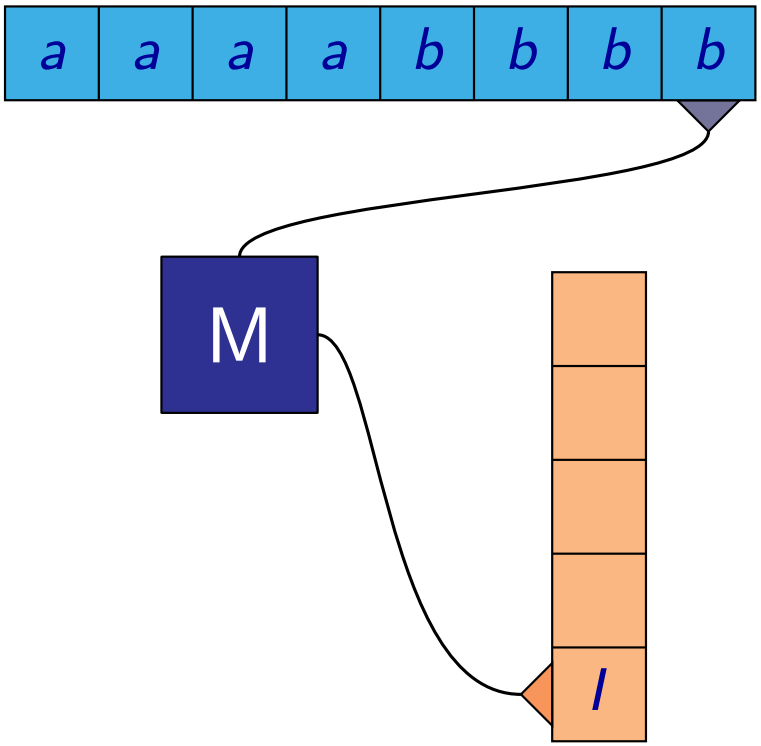
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



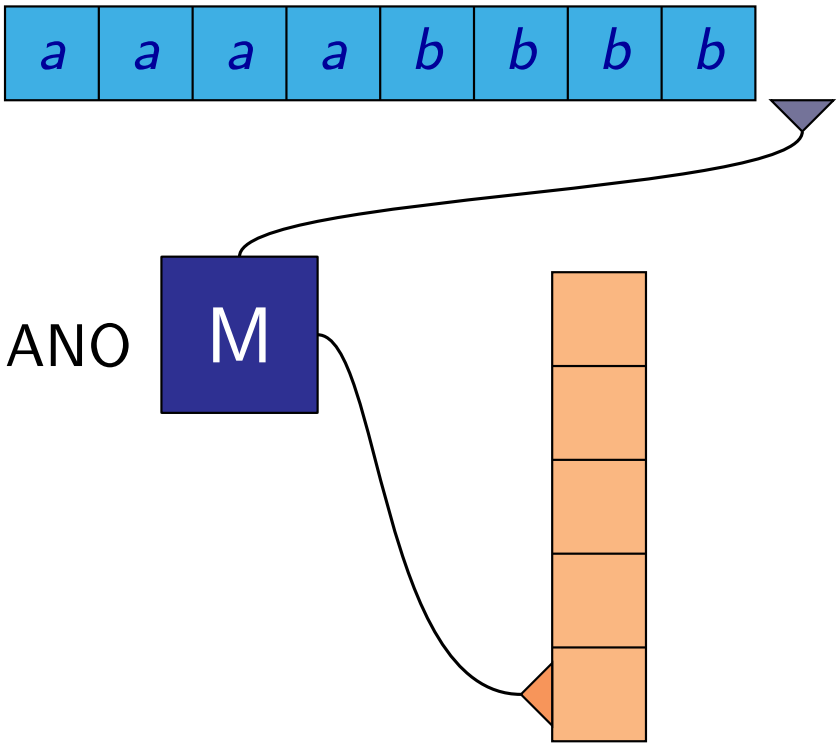
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



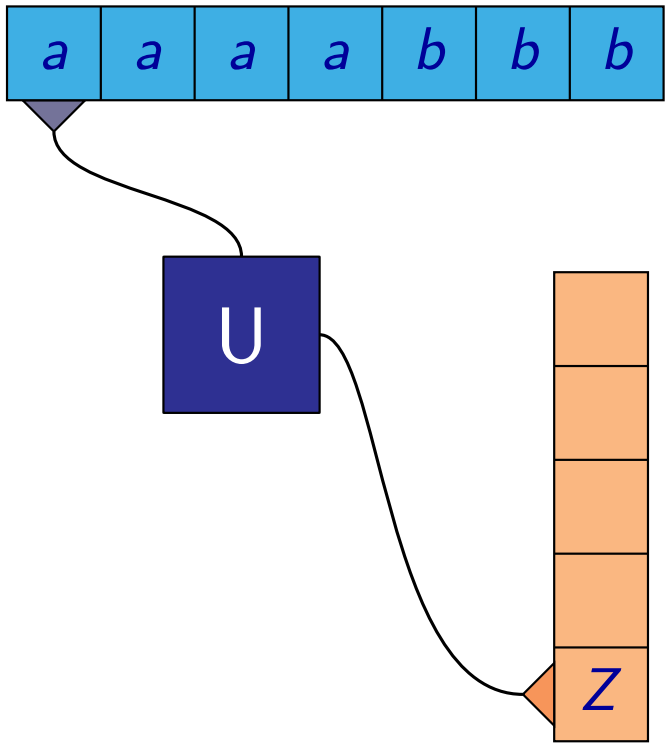
Zásobníkový automat

- Slovo *aaaabbbb* patří do jazyka $L = \{a^i b^i \mid i \geq 1\}$
- Automat přečetl celé slovo a skončil s prázdným zásobníkem, takže slovo přijal



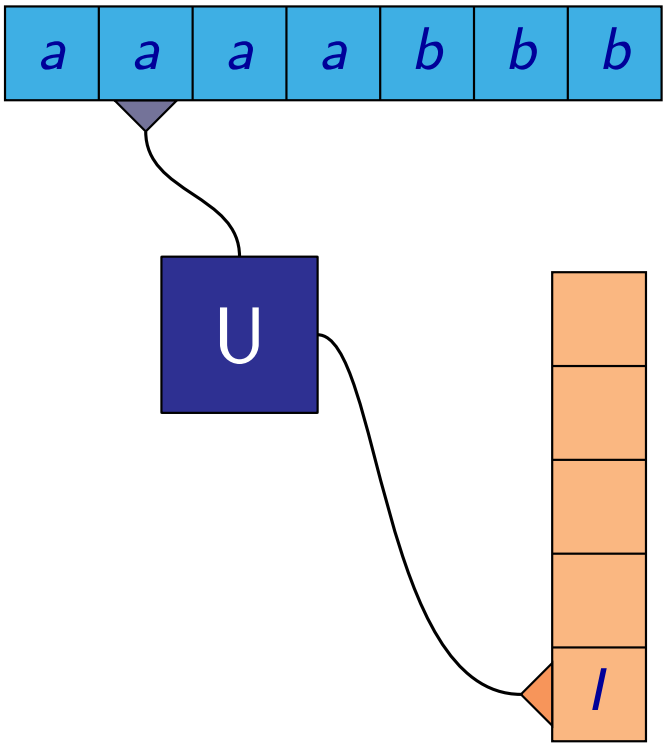
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



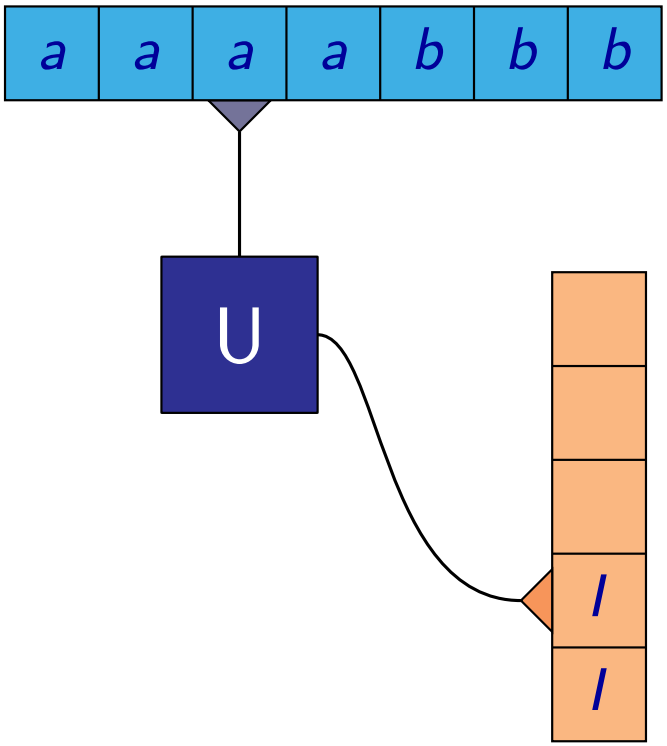
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



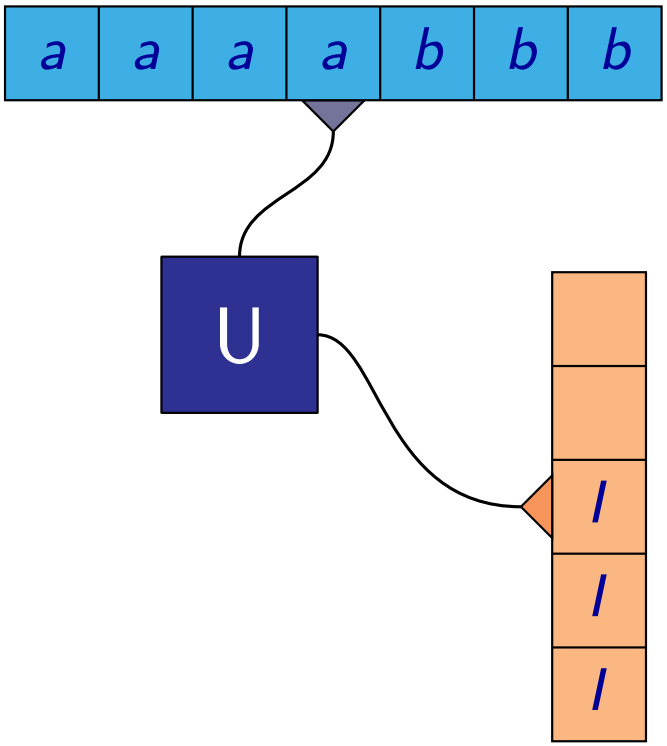
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



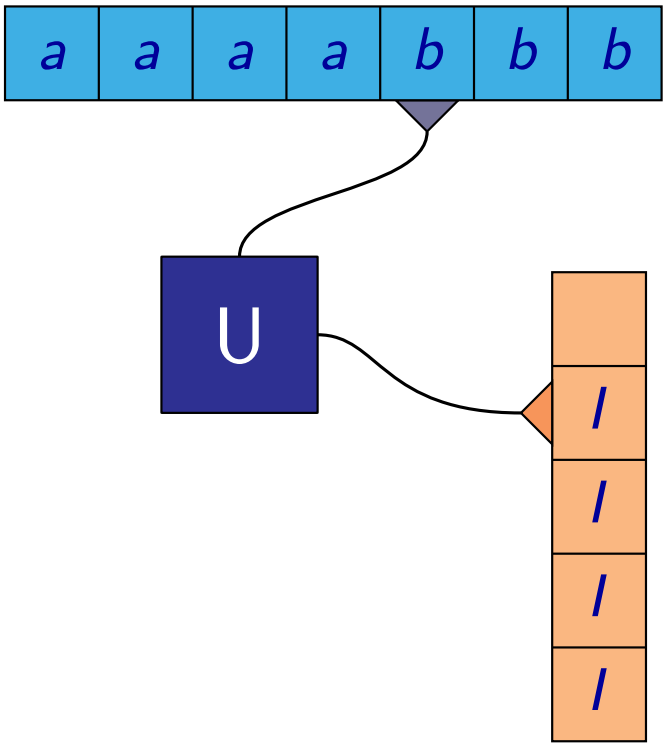
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



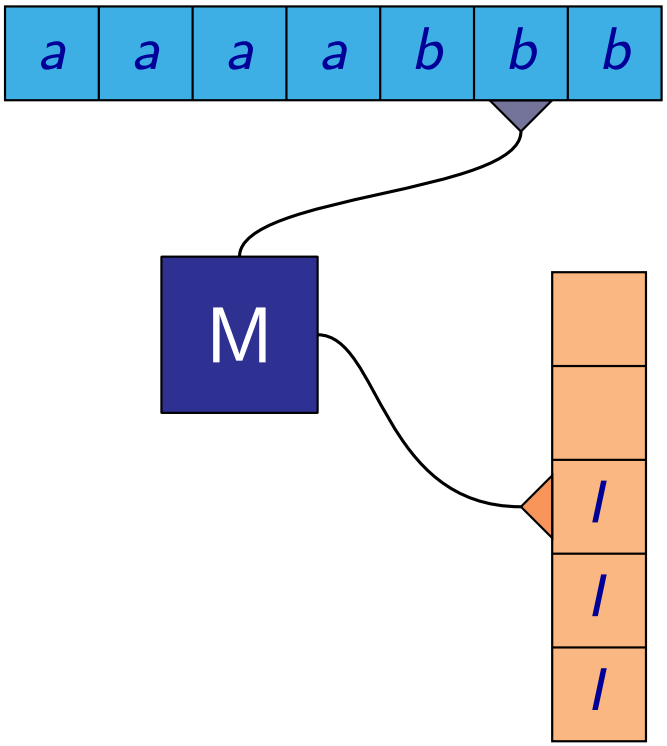
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



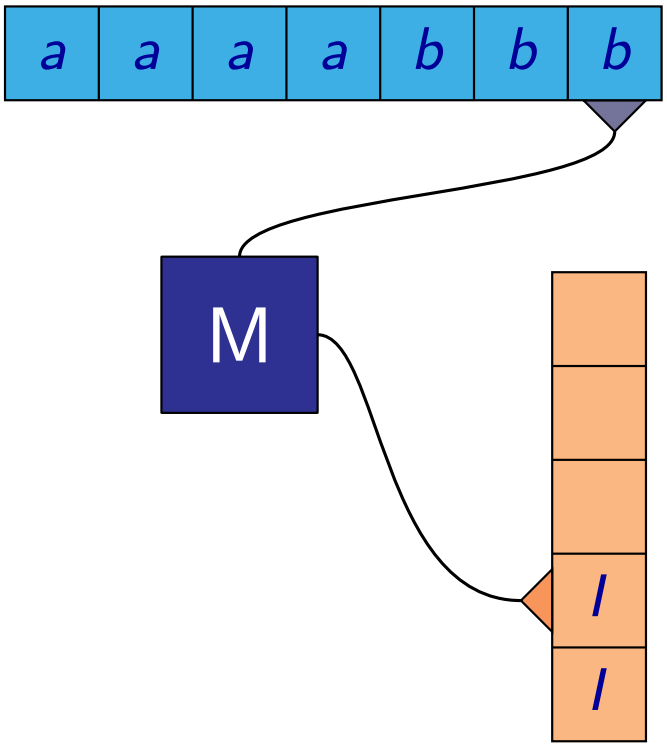
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



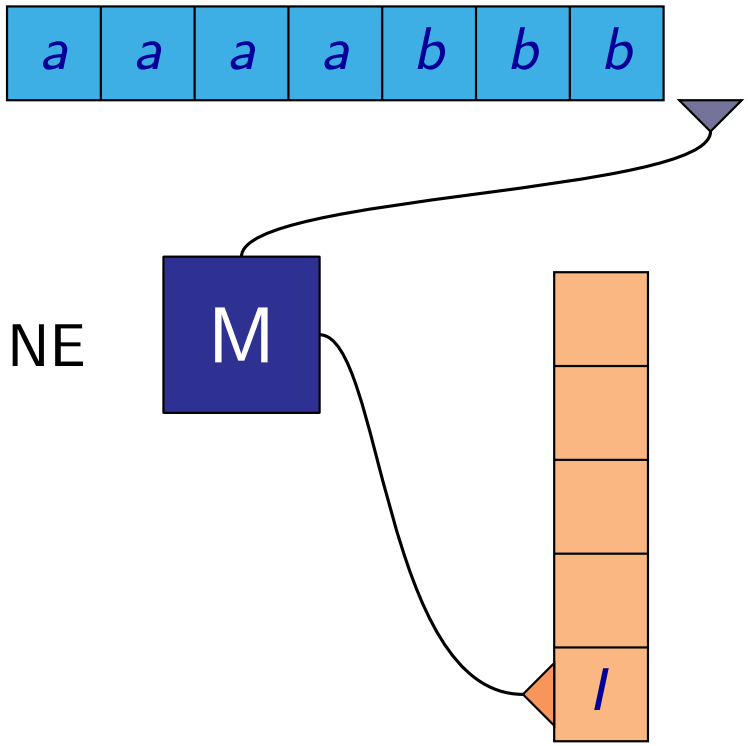
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



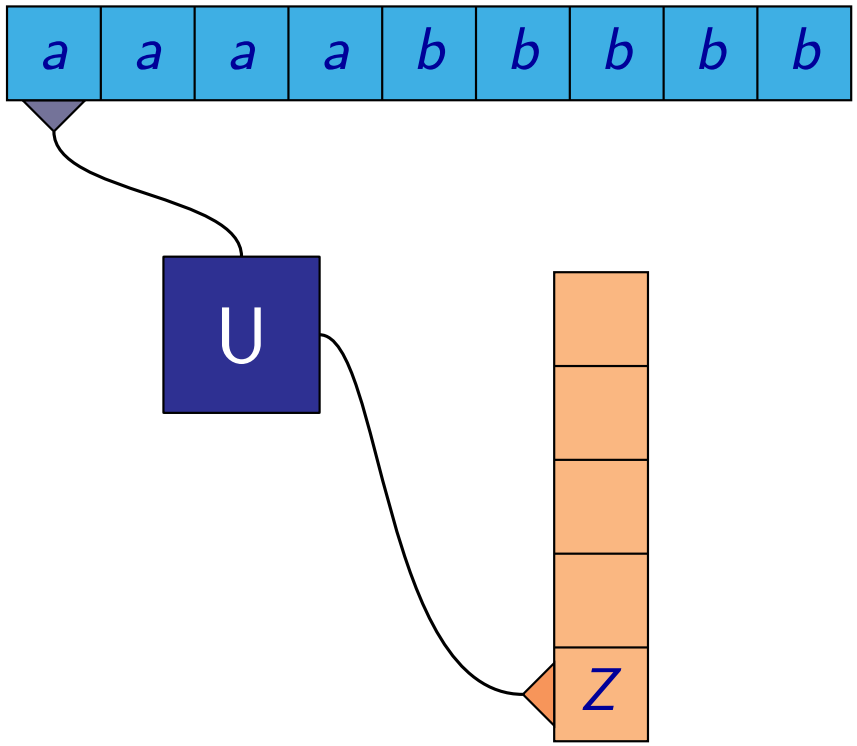
Zásobníkový automat

- Slovo *aaaabbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$
- Automat přečetl celé slovo, ale nevyprázdnil zásobník, takže slovo nepřijal



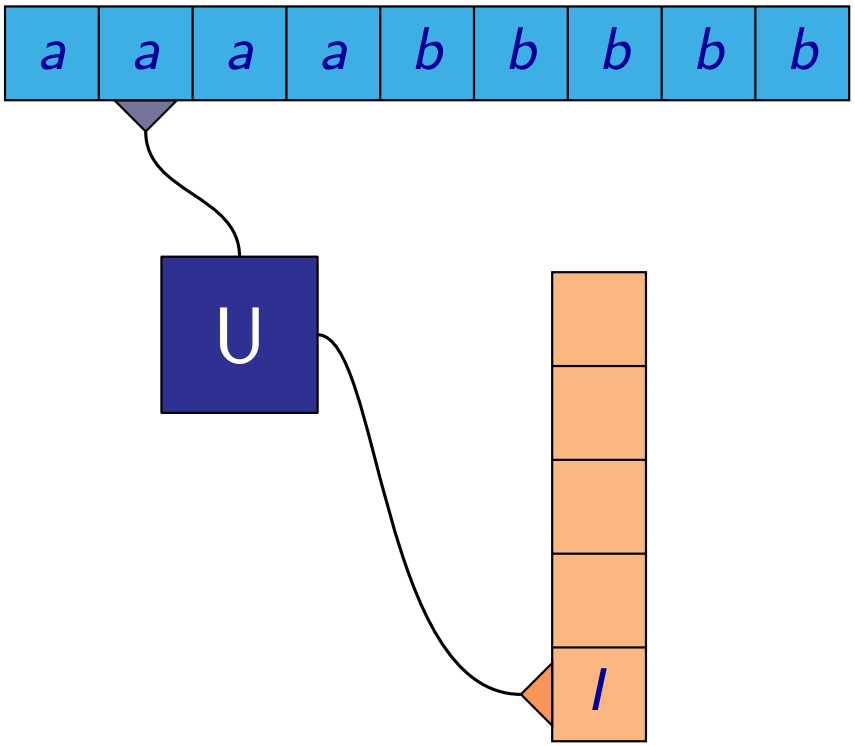
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



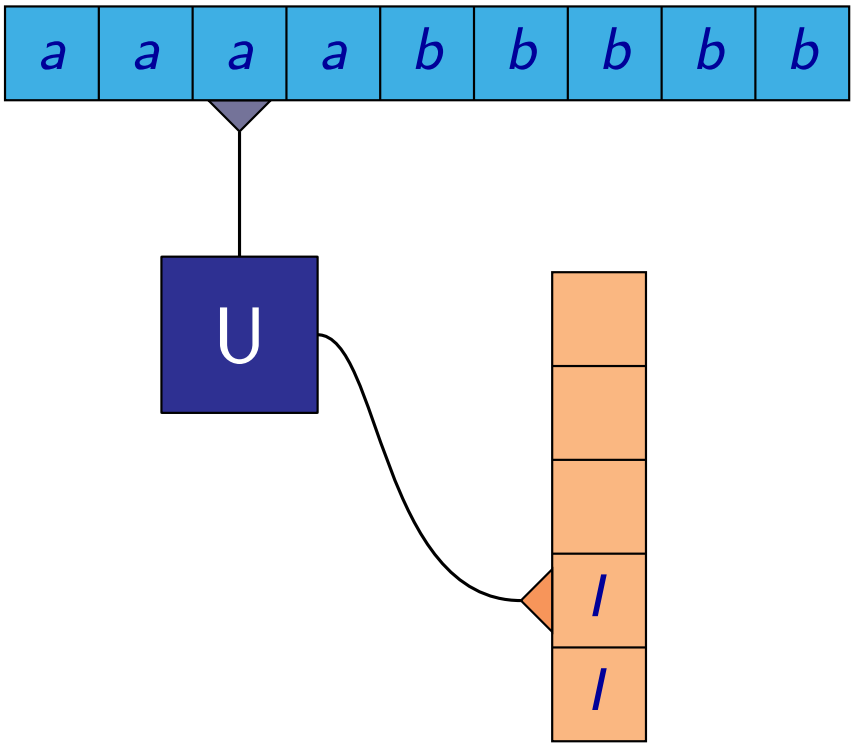
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



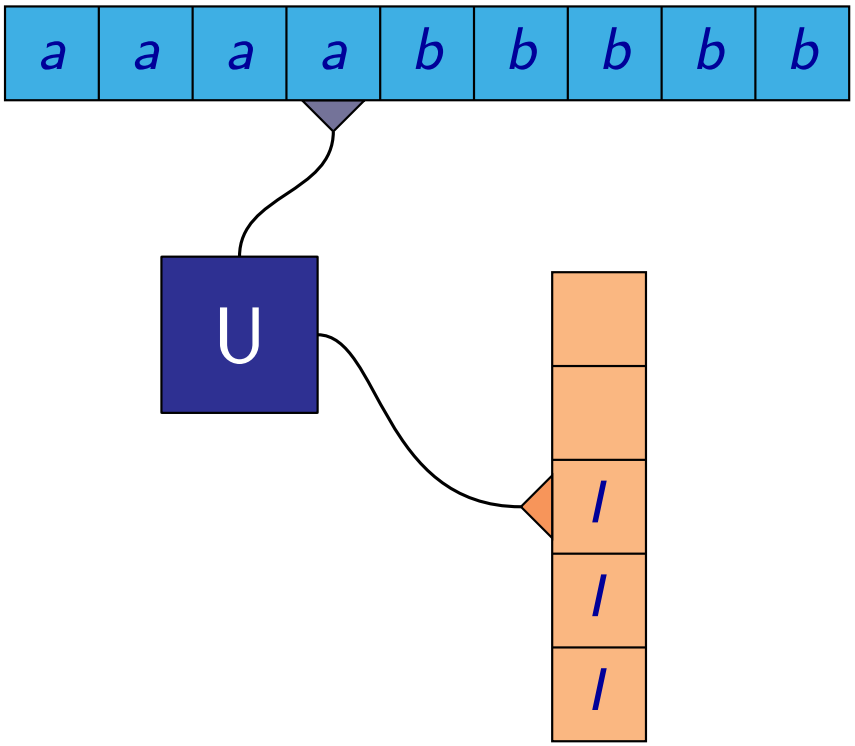
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



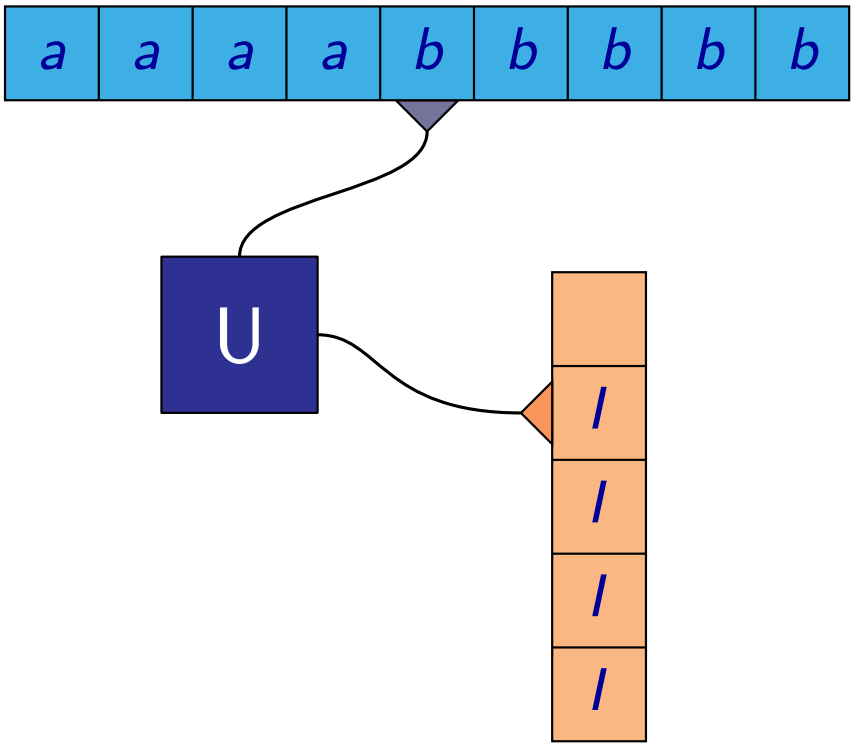
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



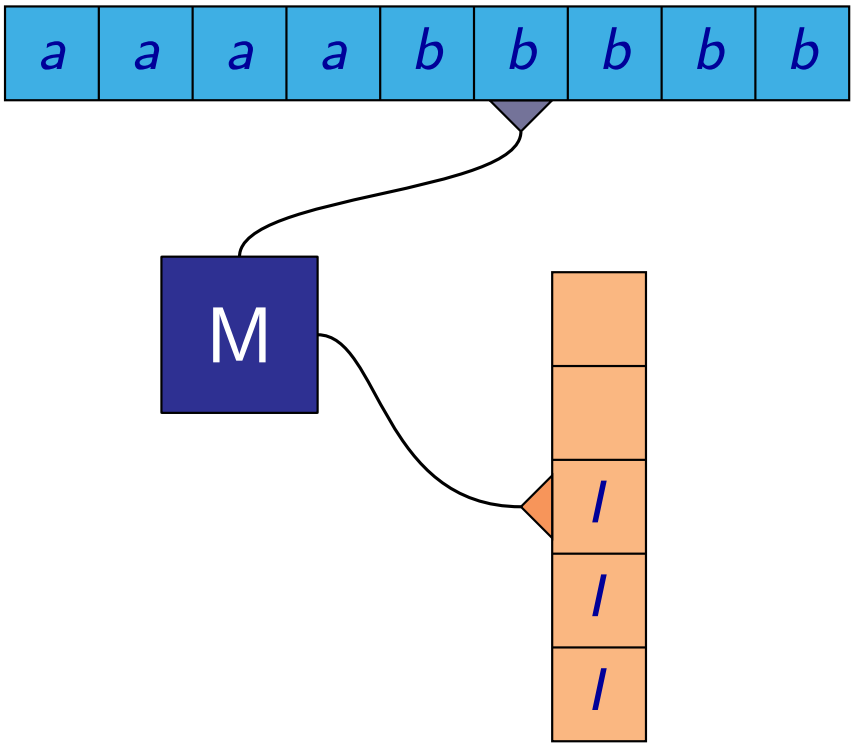
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



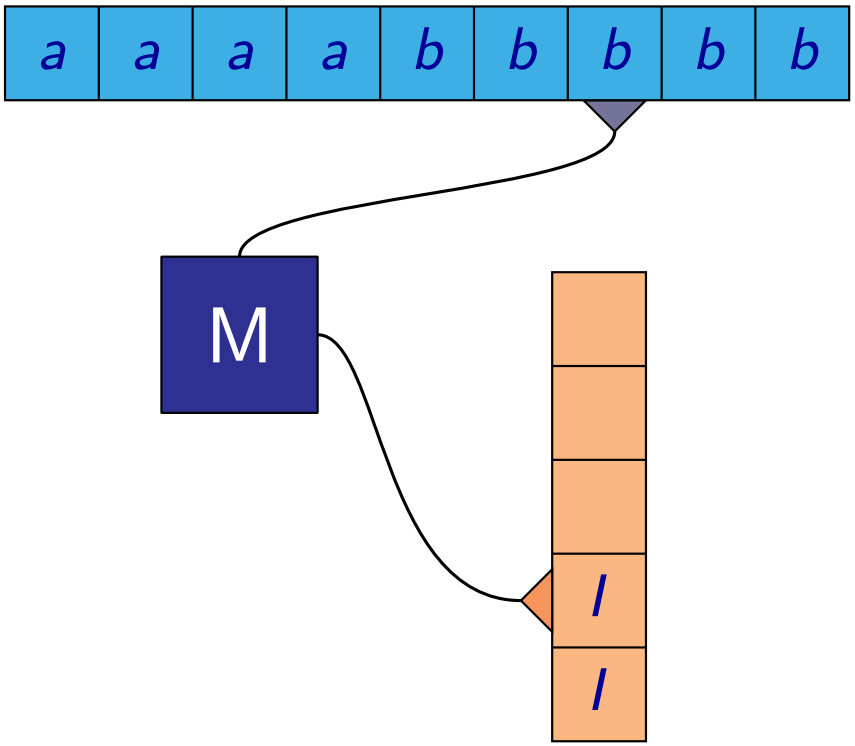
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



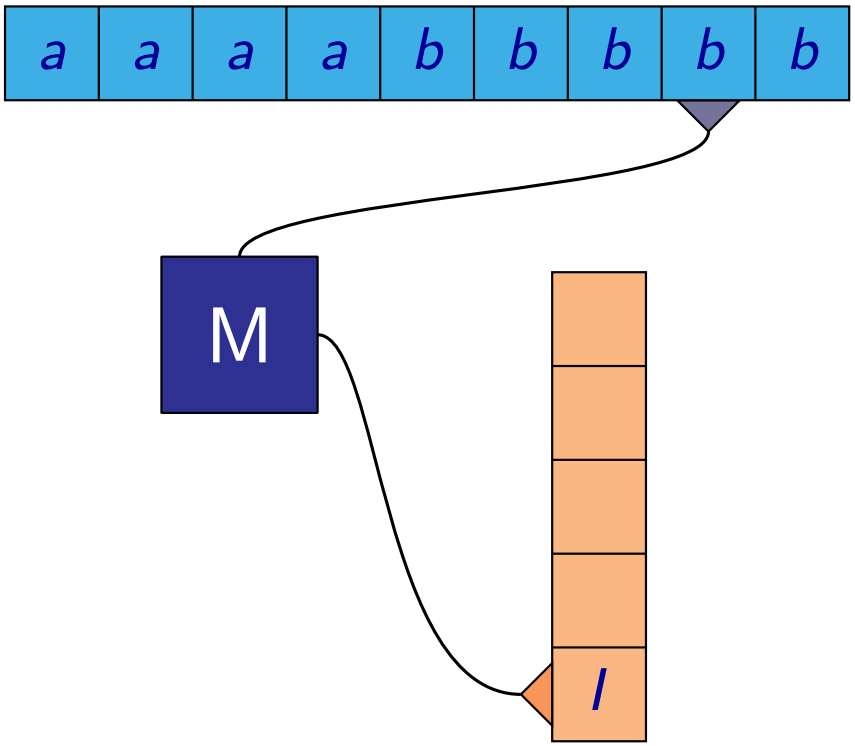
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



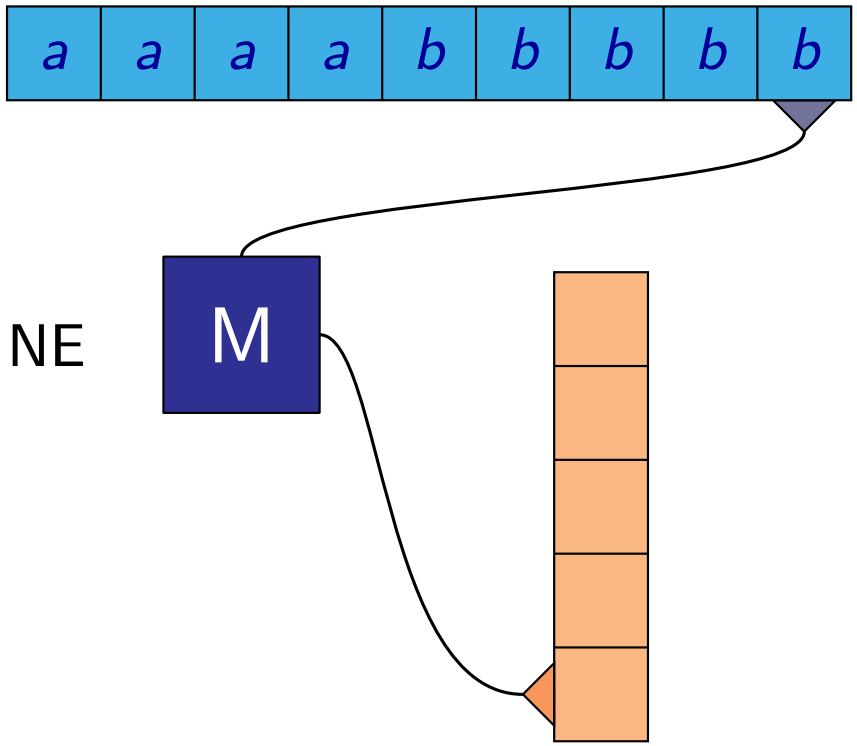
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



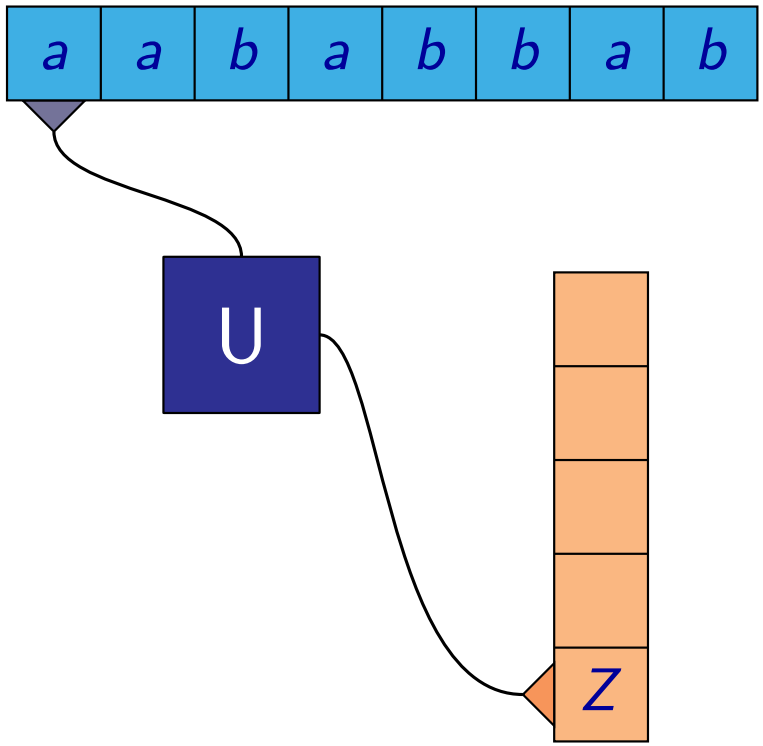
Zásobníkový automat

- Slovo *aaaabbbbb* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$
- Automat čte *b*, má smazat symbol na zásobníku a tam žádný není, takže slovo nepřijal



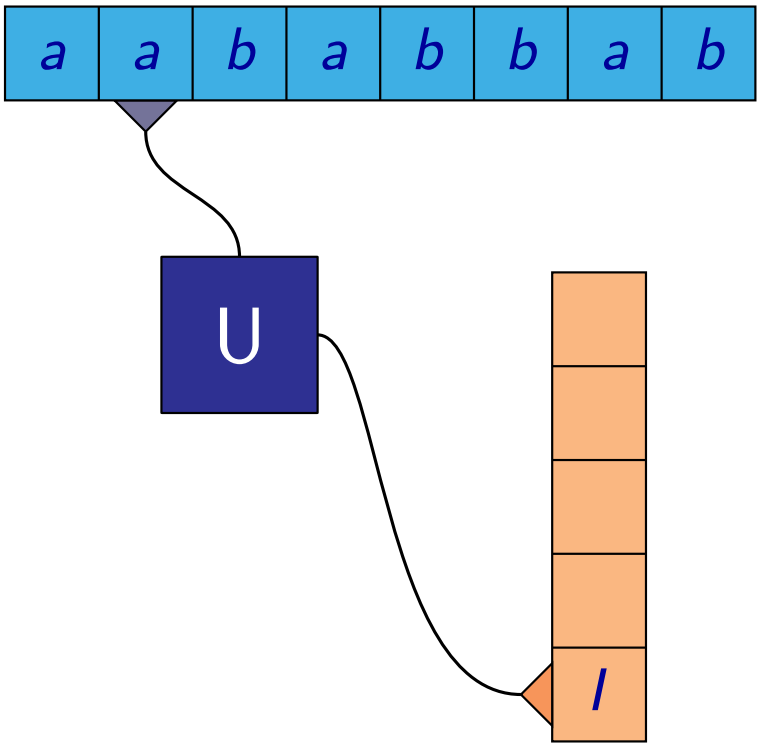
Zásobníkový automat

- Slovo *aababbab* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



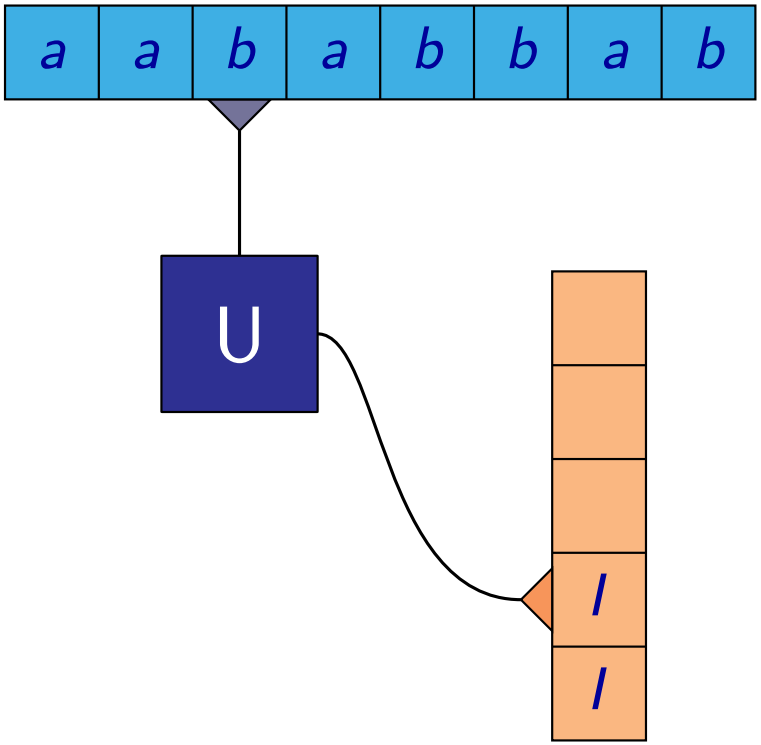
Zásobníkový automat

- Slovo *aababbab* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



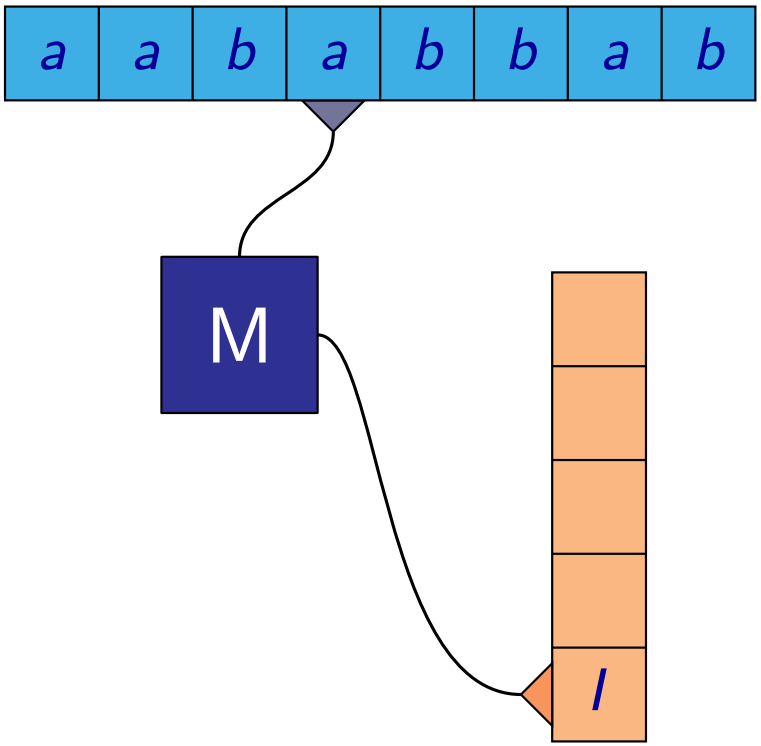
Zásobníkový automat

- Slovo *aababbab* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



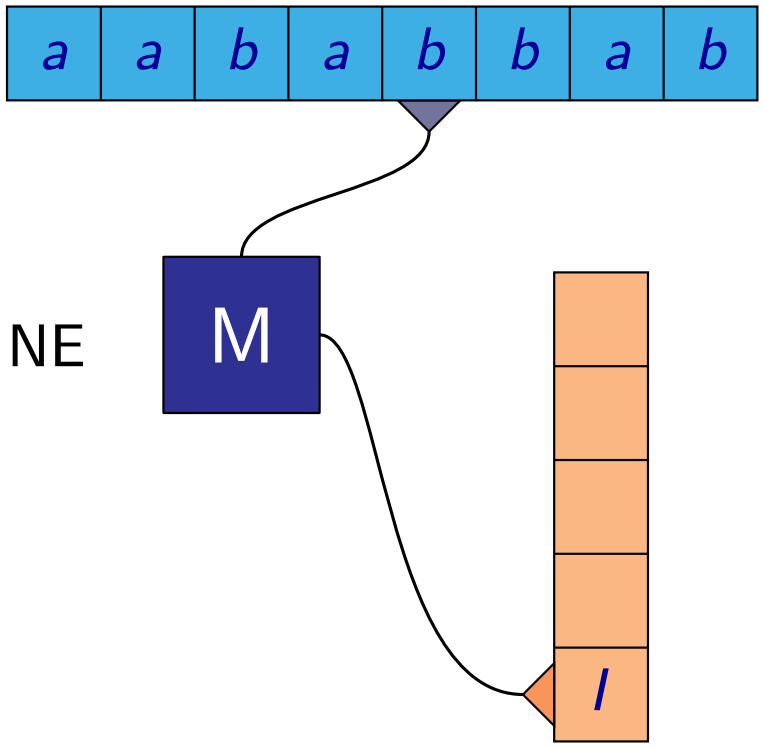
Zásobníkový automat

- Slovo *aababbab* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$



Zásobníkový automat

- Slovo *aababbab* nepatří do jazyka $L = \{a^i b^i \mid i \geq 1\}$
- Automat přečetl *a*, ale již byl ve stavu, kdy maže, takže slovo nepřijal



Zásobníkový automat

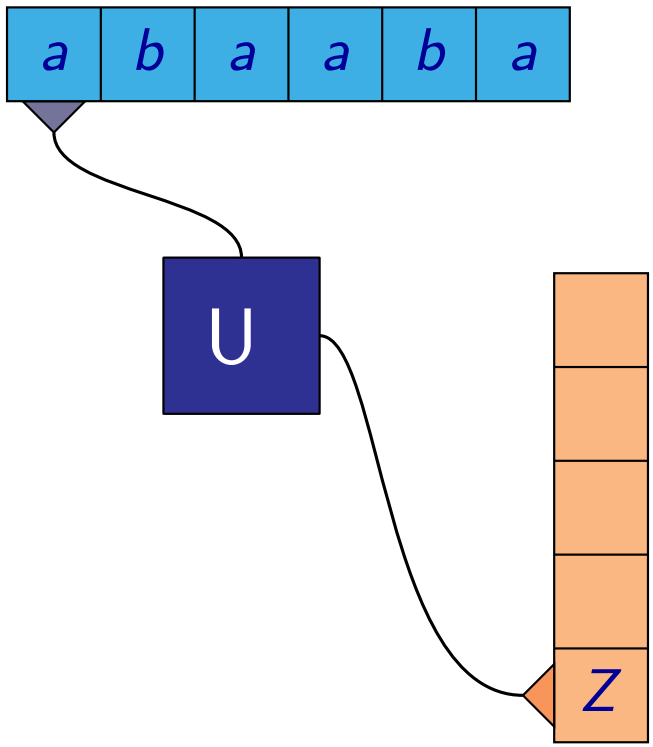
- Uvedený zásobníkový automat měl vždy jasně určen další krok – byl deterministický.
- Je možné každý bezkontextový jazyk poznat deterministickým zásobníkovým automatem?

Zásobníkový automat

- Uvedený zásobníkový automat měl vždy jasně určen další krok – byl deterministický.
- Je možné každý bezkontextový jazyk poznat deterministickým zásobníkovým automatem?
- Uvažujme jazyk $L = \{w(w)^R \mid w \in \{a, b\}^*\}$
- První půlku slova můžeme uložit na zásobník.
- Při čtení druhé půlky mažeme symboly ze zásobníku, pokud jsou stejné jako na vstupu.
- Pokud bude zásobník prázdný po přečtení celého slova, byla druhá půlka stejná jako první.

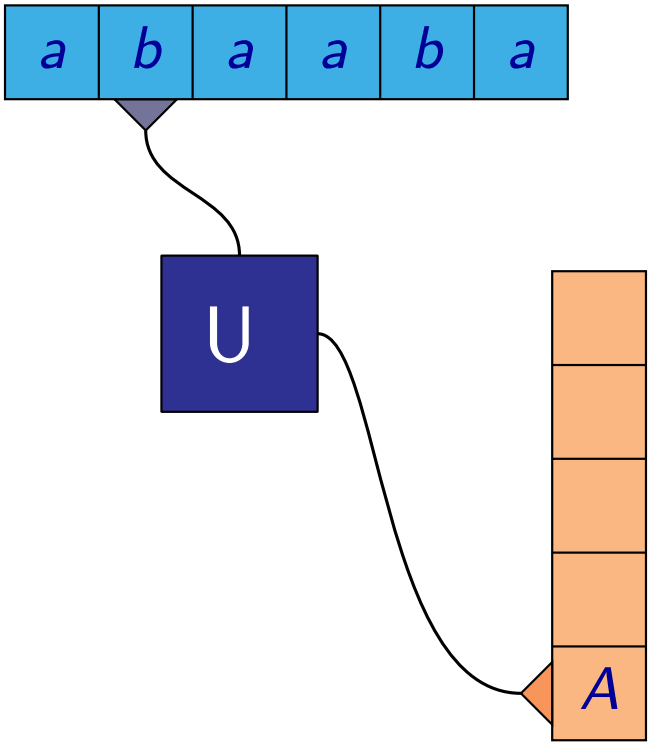
Zásobníkový automat

- Slovo *abaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$



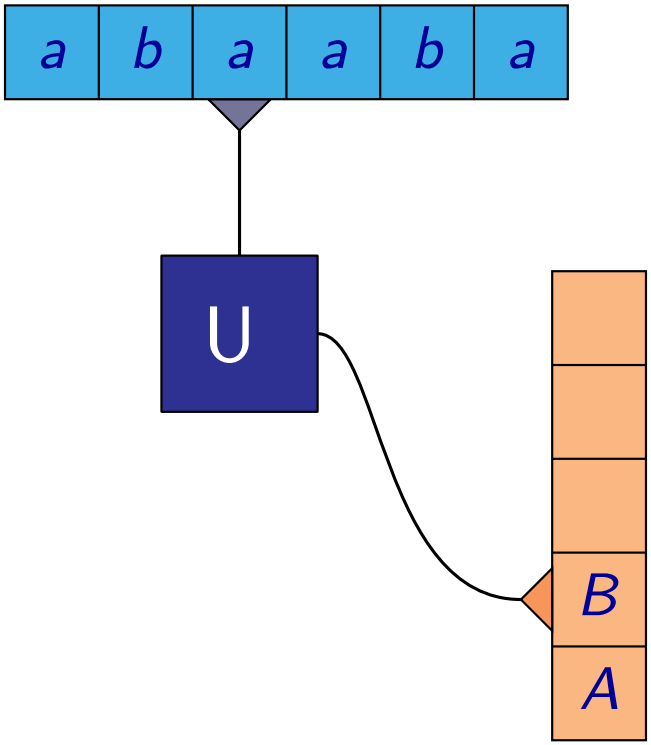
Zásobníkový automat

- Slovo *abaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$



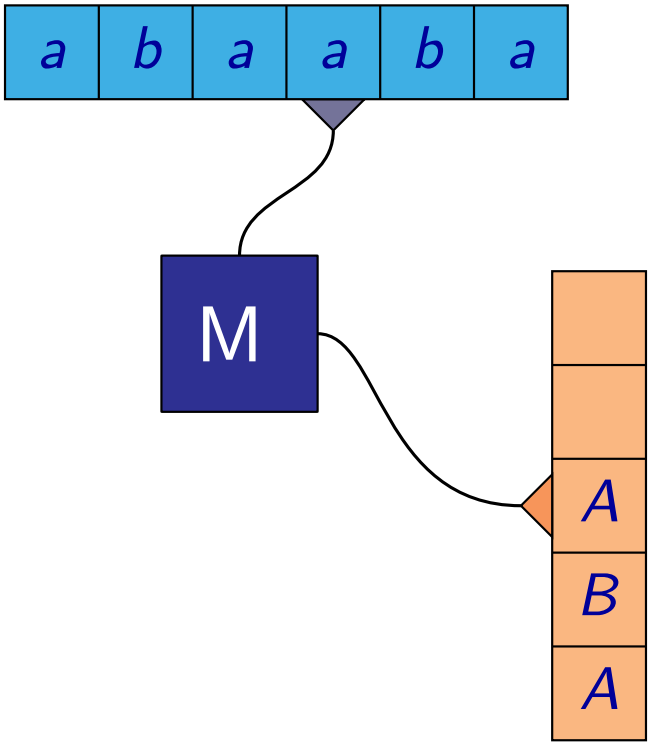
Zásobníkový automat

- Slovo *abaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$
- Při čtení a, stavu U a vrcholu zásobníku B, musí změnit stav na M a uložit A na zásobník



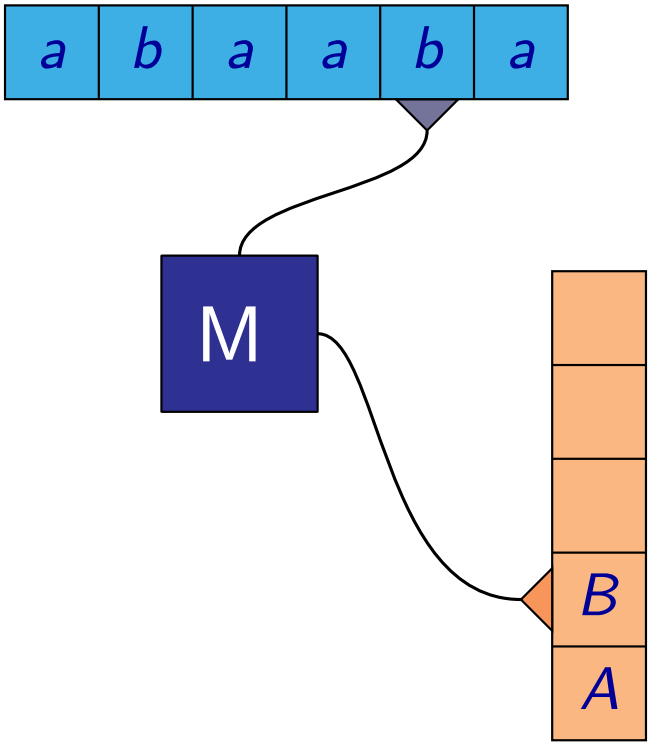
Zásobníkový automat

- Slovo *abaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$
- Při čtení a, stavu U a vrcholu zásobníku B, musí změnit stav na M a uložit A na zásobník



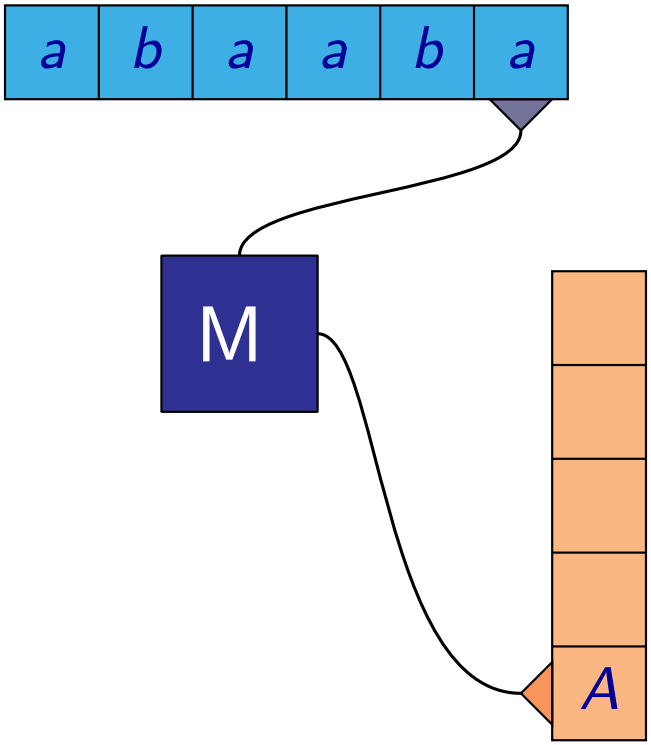
Zásobníkový automat

- Slovo *abaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$
- Při čtení a, stavu U a vrcholu zásobníku B, musí změnit stav na M a uložit A na zásobník



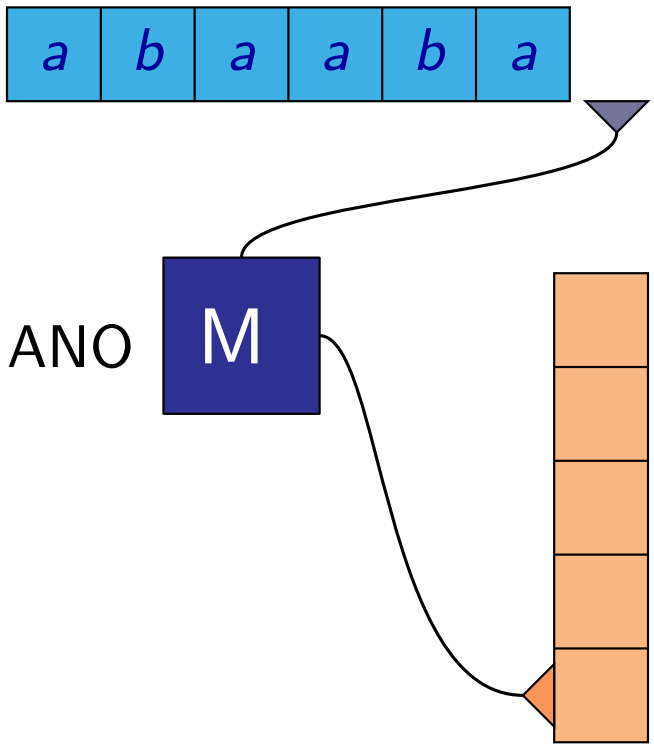
Zásobníkový automat

- Slovo *abaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$
- Při čtení a, stavu U a vrcholu zásobníku B, musí změnit stav na M a uložit A na zásobník



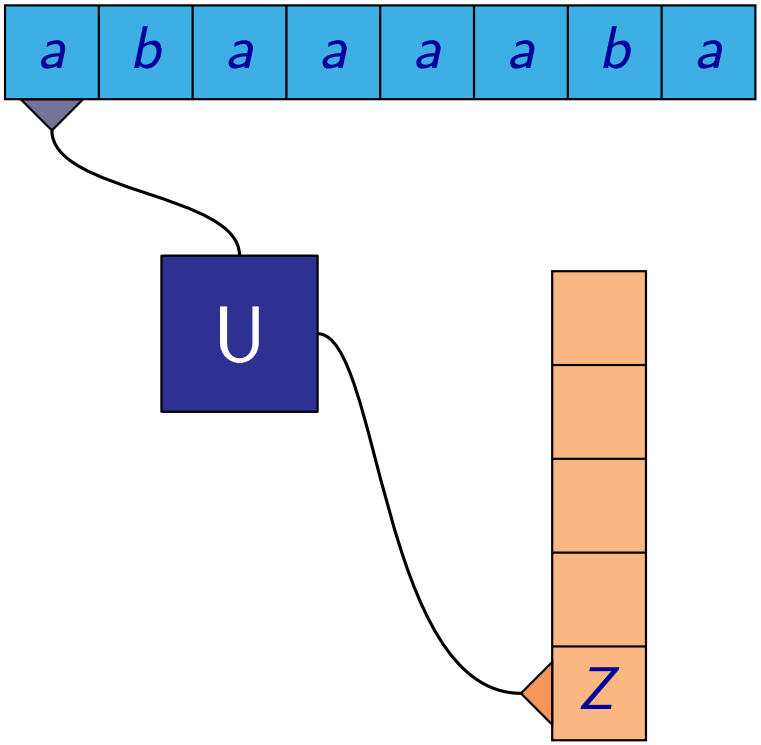
Zásobníkový automat

- Slovo *abaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$
- Při čtení a, stavu U a vrcholu zásobníku B, musí změnit stav na M a uložit A na zásobník



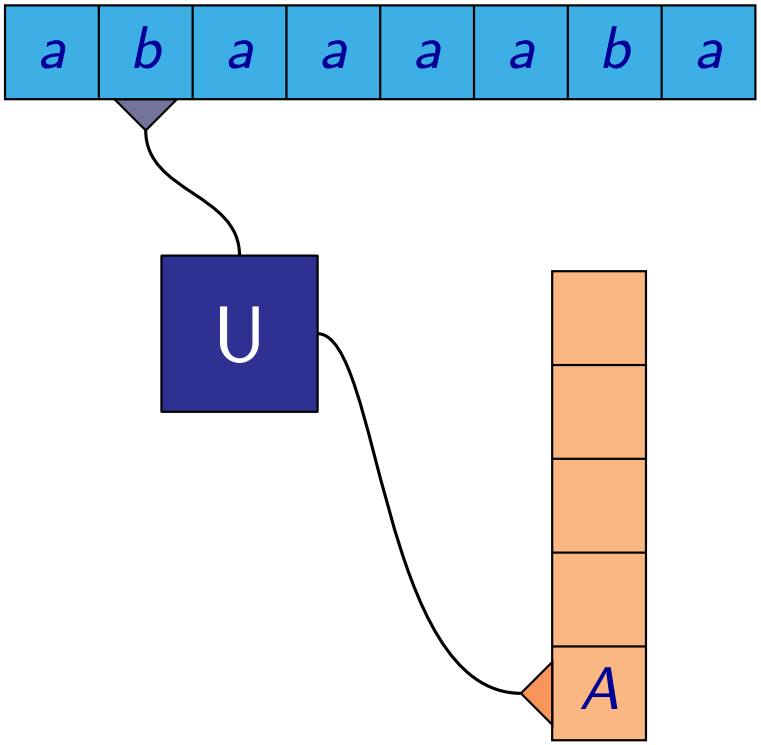
Zásobníkový automat

- Slovo *abaaaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$



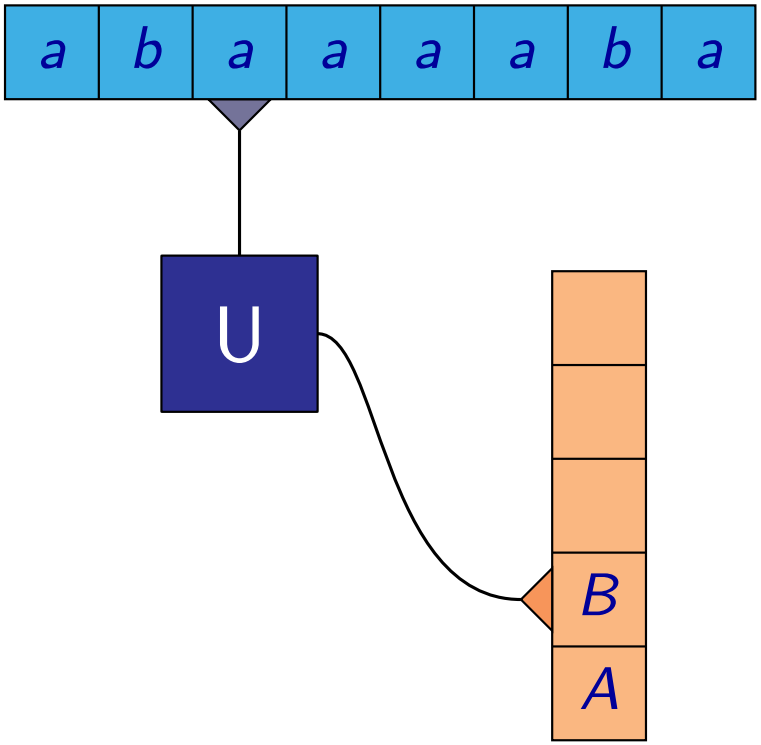
Zásobníkový automat

- Slovo *abaaaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$



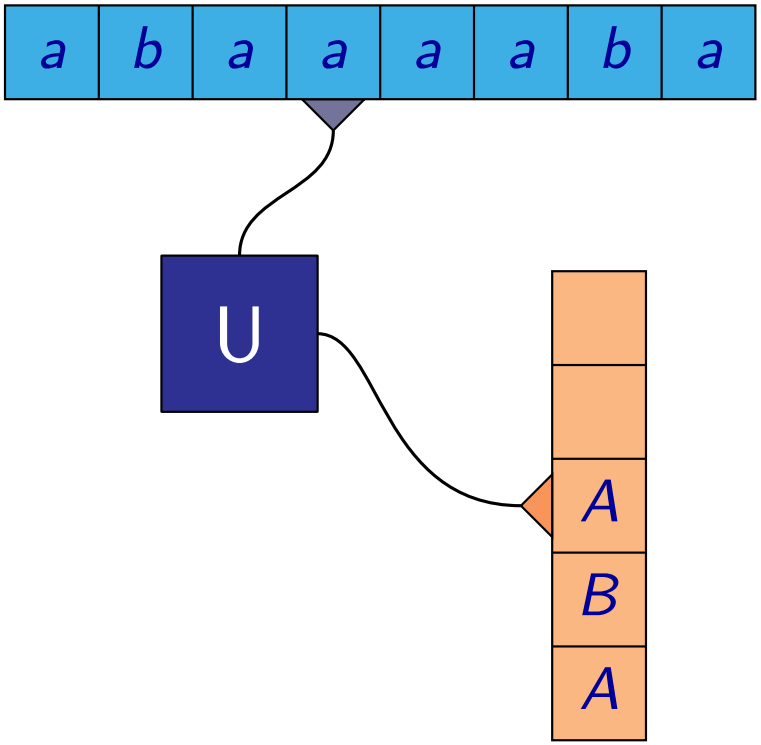
Zásobníkový automat

- Slovo *abaaaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$
- Při čtení a, stavu U a vrcholu zásobníku B, nemění stav a uloží A na zásobník



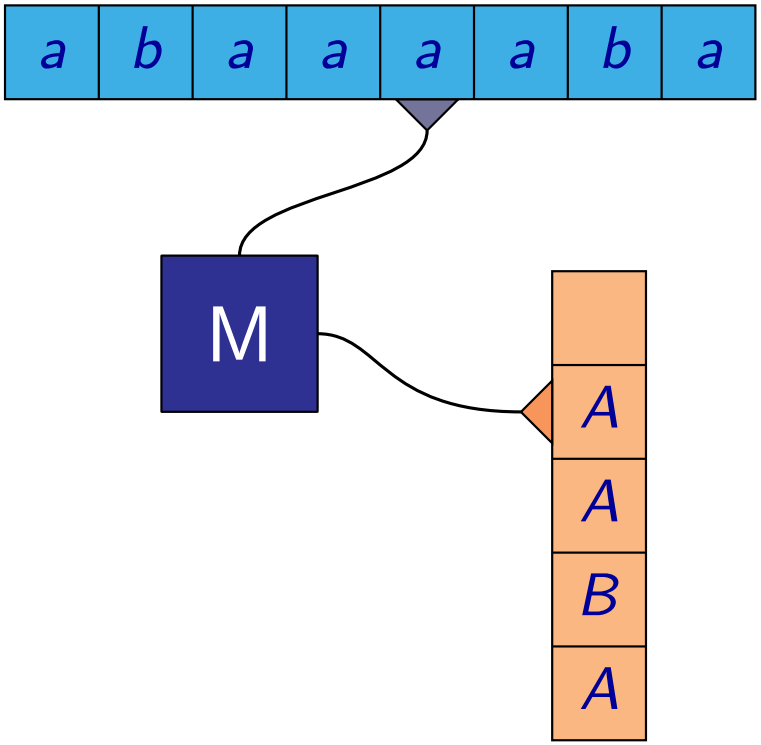
Zásobníkový automat

- Slovo *abaaaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$
- Při čtení a, stavu U a vrcholu zásobníku B, nemění stav a uloží A na zásobník



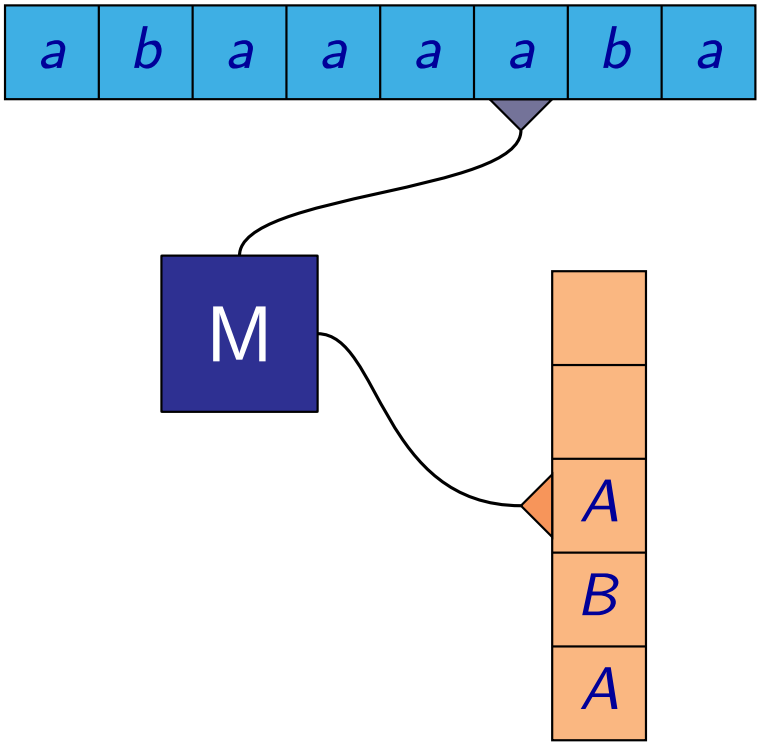
Zásobníkový automat

- Slovo *abaaaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$
- Při čtení a, stavu U a vrcholu zásobníku B, nemění stav a uloží A na zásobník



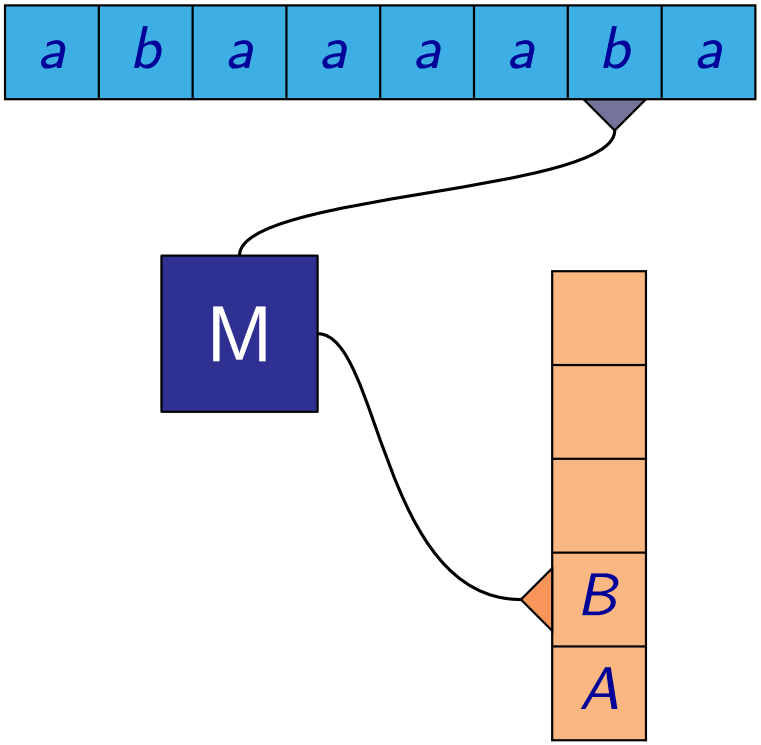
Zásobníkový automat

- Slovo *abaaaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$
- Při čtení a, stavu U a vrcholu zásobníku B, nemění stav a uloží A na zásobník



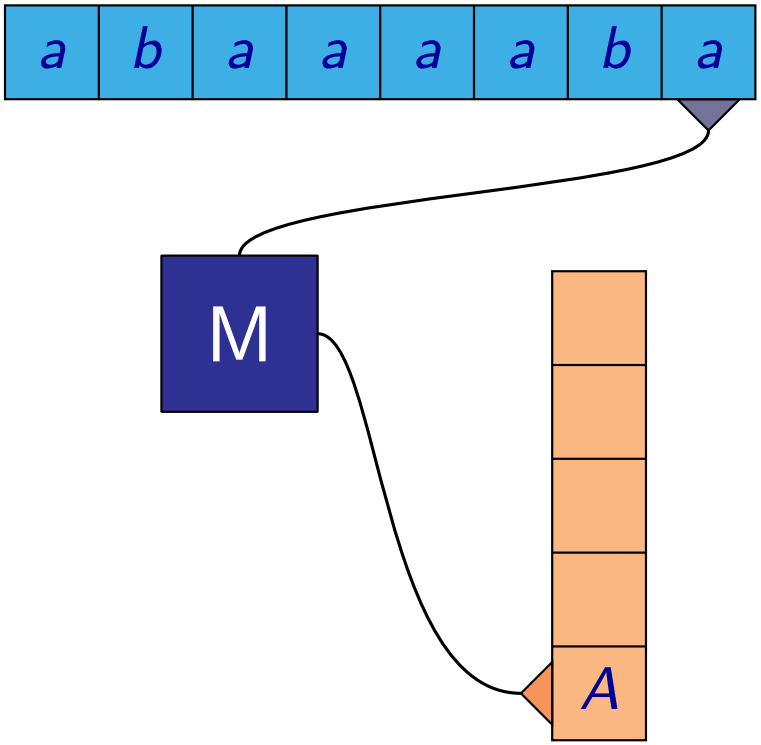
Zásobníkový automat

- Slovo *abaaaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$
- Při čtení a, stavu U a vrcholu zásobníku B, nemění stav a uloží A na zásobník



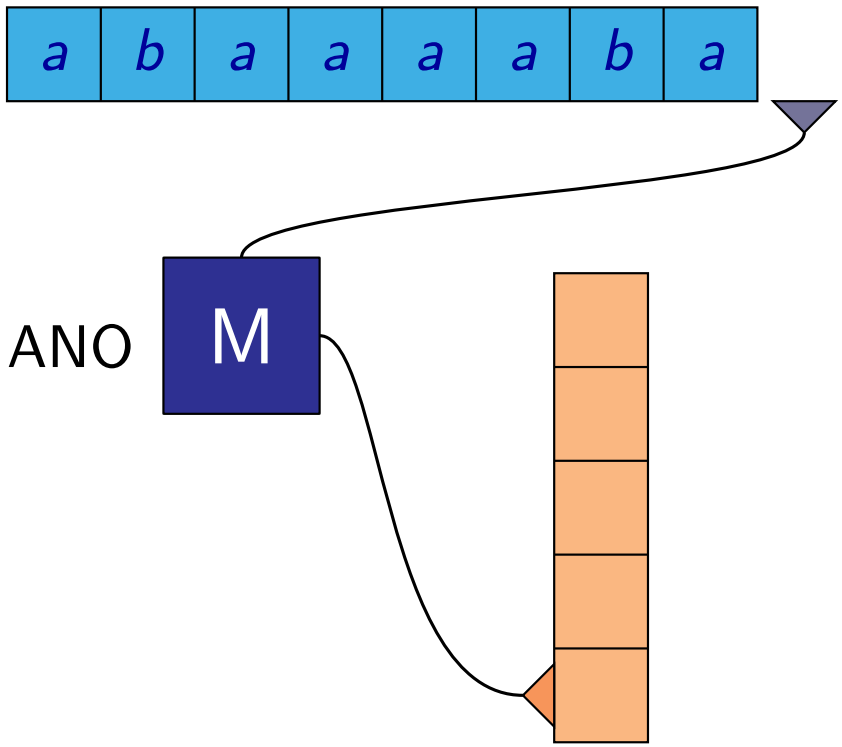
Zásobníkový automat

- Slovo *abaaaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$
- Při čtení a, stavu U a vrcholu zásobníku B, nemění stav a uloží A na zásobník



Zásobníkový automat

- Slovo *abaaaaba* patří do jazyka $L = \{w(w)^R \mid w \in \{a, b\}^*\}$
- Při čtení a, stavu U a vrcholu zásobníku B, nemění stav a uloží A na zásobník



Zásobníkový automat

- Uvedený zásobníkový automat se nemůže jednoznačně rozhodnout, jak má pokračovat. Musí „uhádnout“, kde je půlka slova.
- Na rozdíl od konečných automatů je deterministická verze zásobníkových slabší a proto definujeme přímo nedeterministické

Zásobníkový automat

Definice

Zásobníkový automat je uspořádaná šestice $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$, kde

- Q je konečná neprázdná množina stavů
- Σ je konečná neprázdná množina zvaná vstupní abeceda
- Γ je konečná neprázdná množina zvaná zásobníková abeceda
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$ je (nedeterministická) přechodová funkce
- $q_0 \in Q$ je počáteční stav
- $Z_0 \in \Gamma$ je počáteční zásobníkový symbol

Zásobníkový automat

Používají se dvě různé definice toho, kdy automat přijímá dané slovo:

- Jestliže zásobníkový automat M přijímá **prázdným zásobníkem**, příjme slovo w tehdy, jestliže existuje výpočet automatu M nad slovem w takový, že automat přečte celé slovo w a po jeho přečtení má prázdný zásobník.
- Jestliže zásobníkový automat M přijímá **koncovým stavem**, příjme slovo w tehdy, jestliže existuje výpočet automatu M nad slovem w takový, že automat přečte celé slovo w a po jeho přečtení je řídící jednotka automatu M v některém z koncových stavů z množiny F.

Zásobníkový automat

Příklad: $L = \{a^i b^i \mid i \geq 1\}$

$M = (Q, \Sigma, \Gamma, \delta, q_1, Z)$, kde

- $Q = \{q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{Z, A\}$
- $$\begin{array}{ll} \delta(q_1, a, Z) = \{(q_1, A)\} & \delta(q_1, b, Z) = \emptyset \\ \delta(q_1, a, I) = \{(q_1, AA)\} & \delta(q_1, b, A) = \{q_2, \varepsilon)\} \\ \delta(q_2, a, A) = \emptyset & \delta(q_2, b, A) = \{(q_2, \varepsilon)\} \\ \delta(q_2, a, Z) = \emptyset & \delta(q_2, b, Z) = \emptyset \end{array}$$

Poznámka: Často se uvádí jen ty přechody přechodové funkce, které nejsou do prázdné množiny, tedy kdy je skutečně nějaký přechod definován

Zásobníkový automat

Příklad: $L = \{w(w)^R \mid w \in \{a, b\}^*\}$

$M = (Q, \Sigma, \Gamma, \delta, U, Z)$, kde

- $Q = \{q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{Z, A, B\}$
-

$$\begin{array}{ll}
\delta(q_1, a, Z) = \{(q_1, A)\} & \delta(q_1, b, Z) = \{(q_1, B)\} \\
\delta(q_1, a, A) = \{(q_1, AA), (q_2, AA)\} & \delta(q_1, b, A) = \{(q_1, BA), (q_2, BA)\} \\
\delta(q_1, a, B) = \{(q_1, AB), (q_2, AB)\} & \delta(q_1, b, B) = \{(q_1, BB), (q_2, BB)\} \\
\delta(q_2, a, A) = \{(q_2, \varepsilon)\} & \delta(q_2, b, A) = \emptyset \\
\delta(q_2, a, B) = \emptyset & \delta(q_2, b, B) = \{(q_2, \varepsilon)\} \\
\delta(q_2, a, Z) = \emptyset & \delta(q_2, b, Z) = \emptyset \\
\delta(q_1, \varepsilon, Z) = \{(q_1, \varepsilon)\} & \delta(q_2, \varepsilon, Z) = \emptyset \\
\delta(q_1, \varepsilon, A) = \emptyset & \delta(q_2, \varepsilon, A) = \emptyset \\
\delta(q_1, \varepsilon, B) = \emptyset & \delta(q_2, \varepsilon, B) = \emptyset
\end{array}$$

Ekvivalence bezkontextových gramatik a zás. automatů

Tvrzení

K libovolné bezkontextové gramatice G je možné sestrojit (nedeterministický) zásobníkový automat M takový, že $L(G) = L(M)$.

Tvrzení

K libovolnému zásobníkovému automatu M je možné sestrojit bezkontextovou gramatiku G takovou, že $L(M) = L(G)$.

Omezení bezkontextových jazyků

- Ani bezkontextové gramatiky a zásobníkové automaty nejsou dostatečně silné, aby rozpoznávaly všechny jazyky.
- Příklady jazyků, pro které se dá ukázat, že nejsou bezkontextové (tj. není možné vytvořit bezkontextovou gramatiku, která by je generovala):

$$L_1 = \{a^i b^i c^i \mid i \geq 0\}$$
$$L_2 = \{ww \mid w \in \{a, b\}^*\}$$

Turingův stroj

- Rozšíříme deterministické konečné automaty o možnost zápisu na vstupní pásku, pohyb čtecí hlavy oběma směry a prodloužíme jejich pásku do nekonečna.

Turingův stroj

- Rozšíříme deterministické konečné automaty o možnost zápisu na vstupní pásku, pohyb čtecí hlavy oběma směry a prodloužíme jejich pásku do nekonečna.

Definice

Formálně je **Turingův stroj** definován jako šestice $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F)$ kde:

- Q je konečná množina **stavů**
- Γ je konečná množina **páskových symbolů**
- $\Sigma \subseteq \Gamma, \Sigma \neq \emptyset$ je konečná množina **vstupních symbolů**
- $\delta : (Q - F) \times \Gamma \to Q \times \Gamma \times \{-1, 0, +1\}$ je **přechodová funkce**
- $q_0 \in Q$ je **počáteční stav**
- $F \subseteq Q$ je množina **koncových stavů**

Turingův stroj

- Předpokládáme, že v $\Gamma - \Sigma$ je vždy speciální prvek $\square$ označující prázdný znak.
- Konfigurace je dána slovem na pásce, stavem a pozicí čtecí hlavy.
- Konfigurace je **počáteční**, pokud je hlava na prvním symbolu, stav q_0 a na pásce jsou symboly jen z množiny Σ.
- Konfigurace je **koncová**, je-li stav z množiny F.

Máme-li stav q a b na vstupu, tak pro:

- $\delta(q, b) = (q', b', 0)$ změníme stav na q', změníme na aktuální pozici b na b',
- $\delta(q, b) = (q', b', +1)$ změníme stav na q', změníme na aktuální pozici b na b' a posuneme aktuální pozici o 1 doprava,
- $\delta(q, b) = (q', b', -1)$ změníme stav na q', změníme na aktuální pozici b na b' a posuneme aktuální pozici o 1 doleva.

Turingův stroj

- V koncovém stavu z množiny F výpočet končí.
- Slovo je přijato, pokud na něm stroj někdy dojde do přijímajícího stavu.
- Slovo není přijato, pokud běh stroje nikdy neskončí.
- Jazyk $L(\mathcal{M})$ Turingova stroje $\mathcal{M}$ je množina všech slov, která stroj $\mathcal{M}$ přijímá.

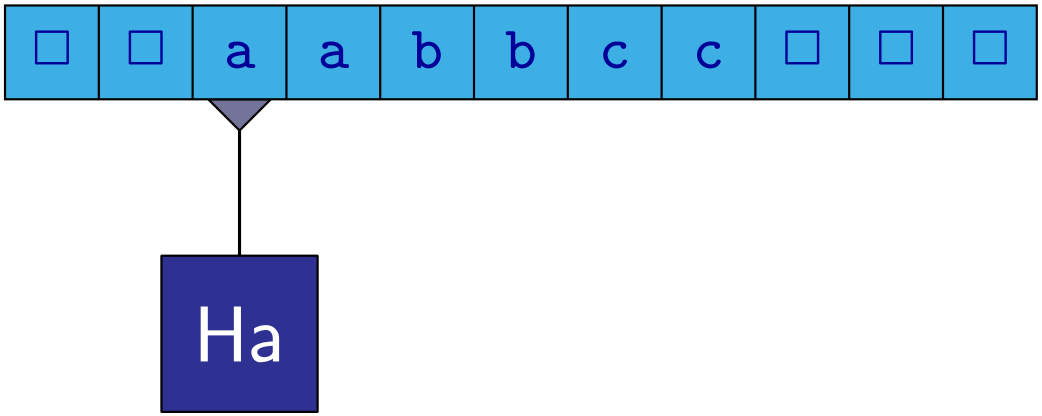
Alternativní definice:

- Množina přijímajících stavů $F = \{q_{\text{ano}}, q_{\text{ne}}\}$.
- Pokud výpočet skončí ve stavu q_{ano}, Turingův stroj slovo přijímá.
- Pokud výpočet skončí ve stavu q_{ne} nebo nikdy neskončí, Turingův stroj slovo nepřijímá.

Turingův stroj

- Turingův stroj nemusí dávat jen odpověď ANO nebo NE, ale může realizovat nějakou funkci, která každému slovu přiřazuje nějaké jiné slovo.
- Slovo přiřazené slovu w je slovo, které zůstane zapsáno na pásce po výpočtu nad slovem w.

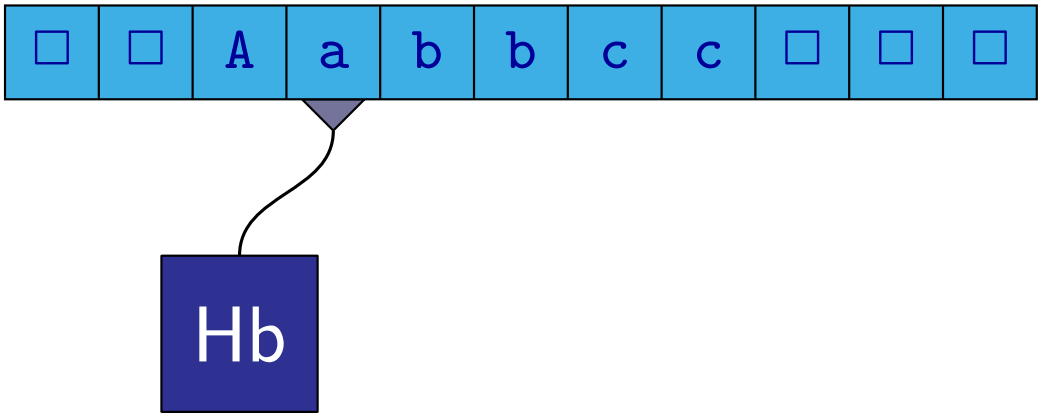
Turingův stroj



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

1. Čte do prvního a, nahradí jej A. Pokud najde dříve b nebo c, dá odpověď Ne.

Turingův stroj

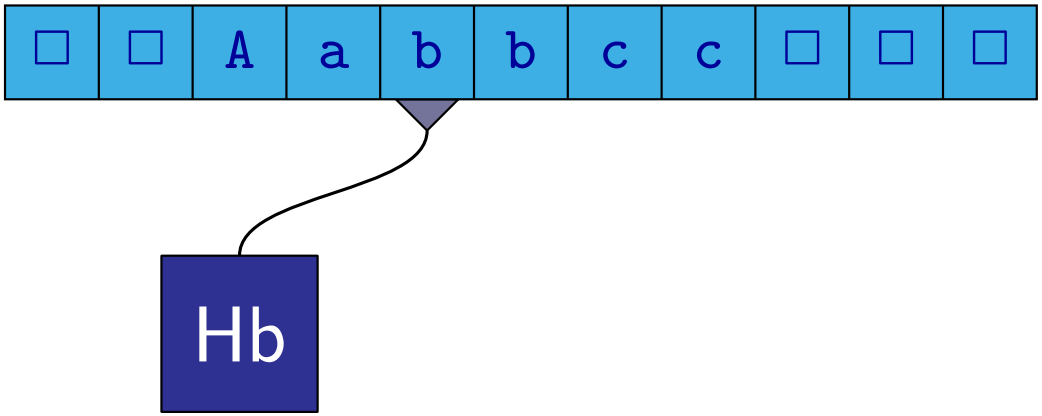


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.

Turingův stroj

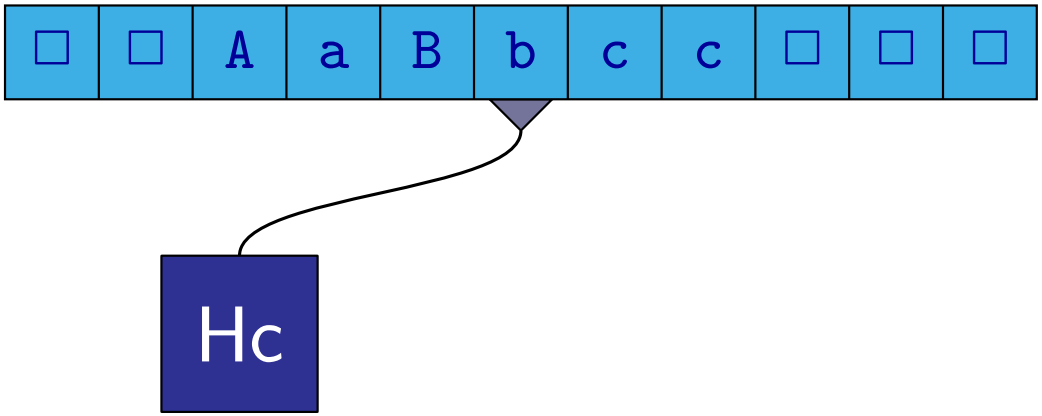


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď Ne.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď Ne.

Turingův stroj

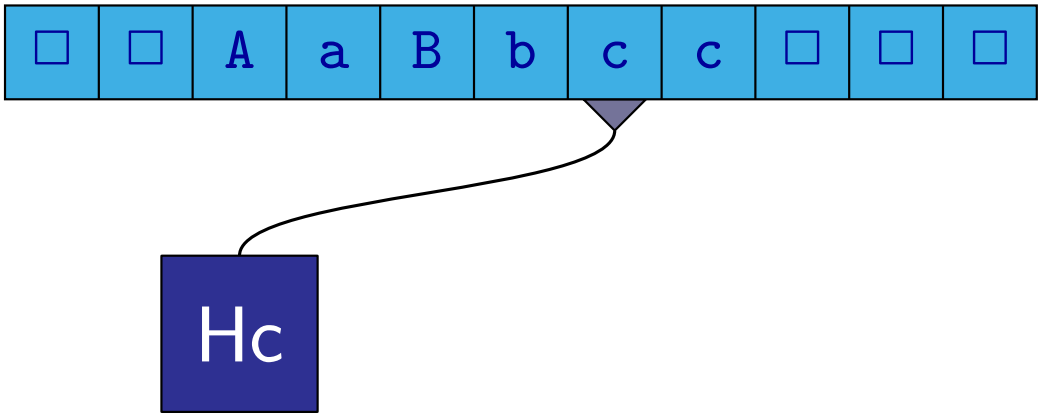


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď NE.

Turingův stroj

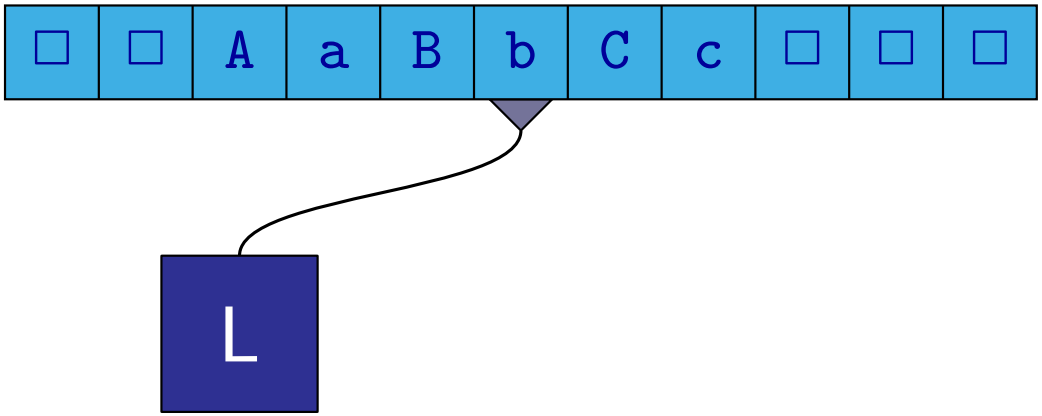


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď NE.

Turingův stroj

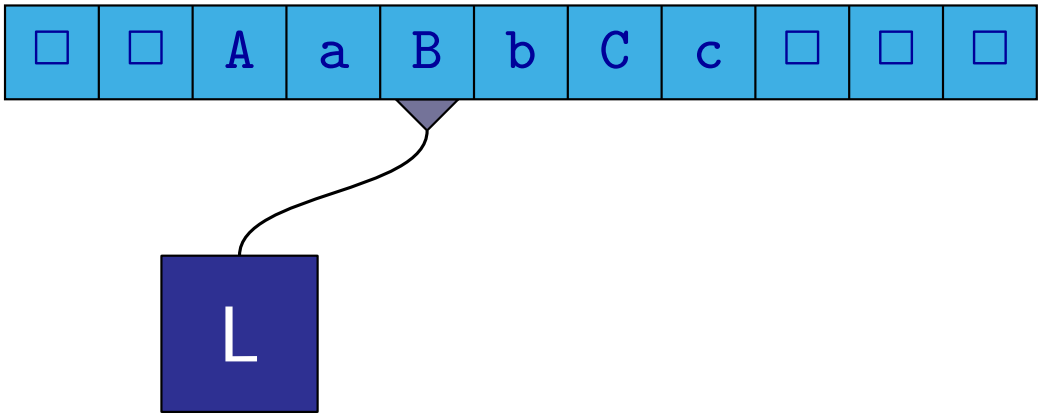


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď NE.
4. Vrací se doleva na nejbližší `A`.

Turingův stroj

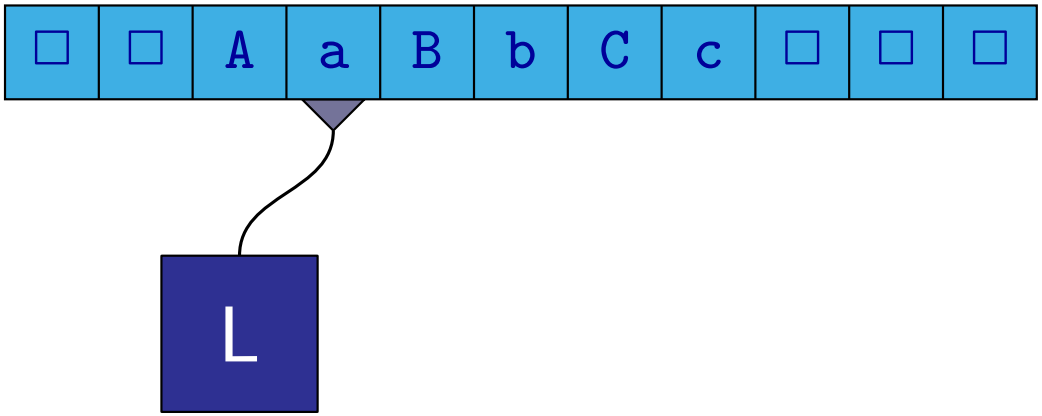


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď NE.
4. Vrací se doleva na nejbližší `A`.

Turingův stroj

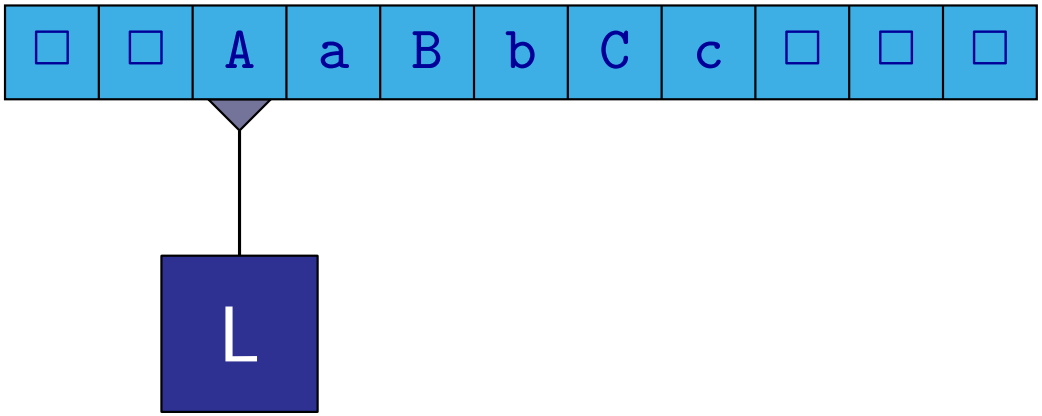


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď Ne.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď Ne.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď Ne.
4. Vrací se doleva na nejbližší `A`.

Turingův stroj

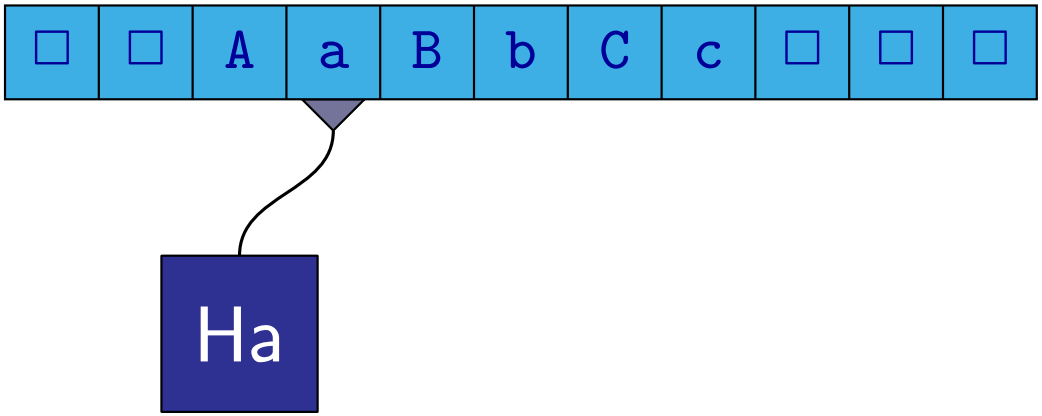


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď NE.
4. Vrací se doleva na nejbližší `A`.

Turingův stroj

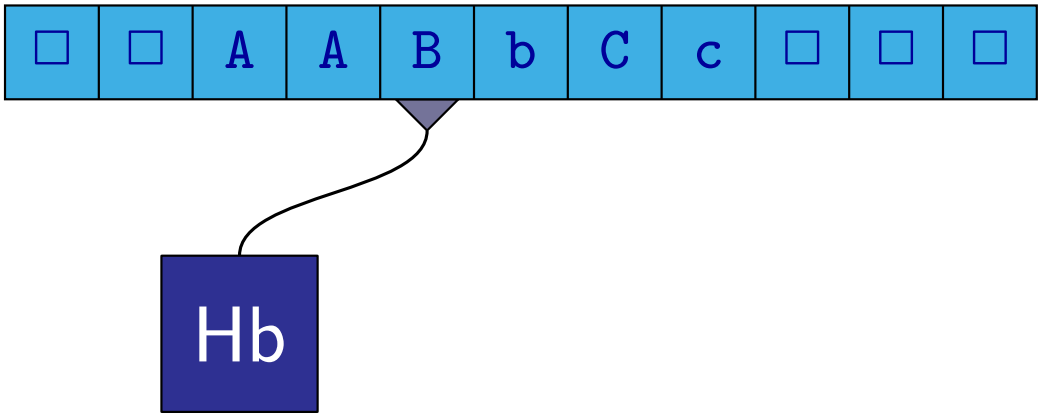


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď Ne.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď Ne.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď Ne.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

Turingův stroj

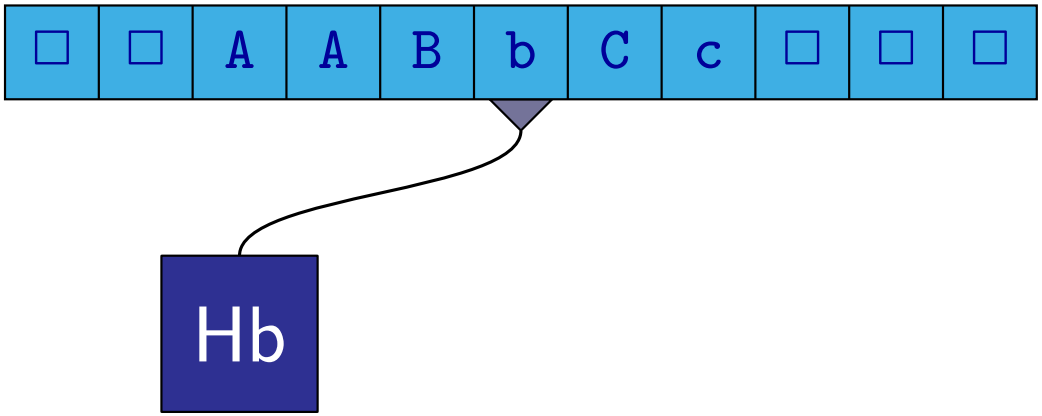


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď NE.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

Turingův stroj

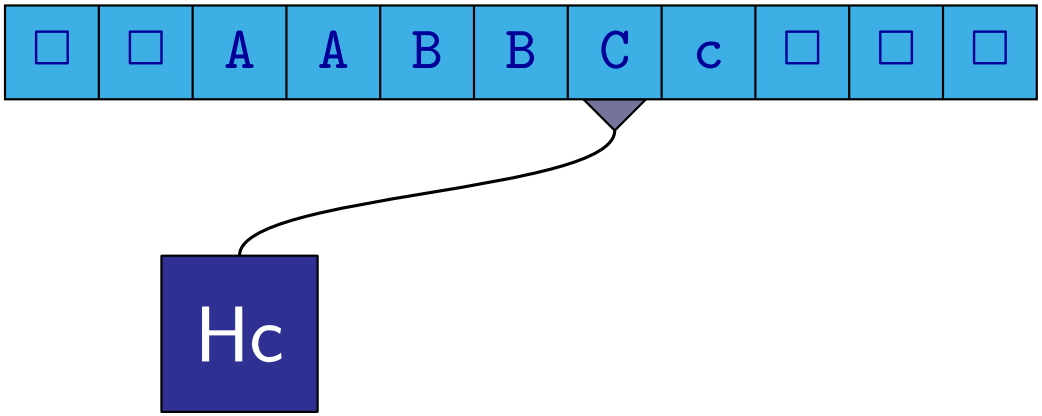


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď NE.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

Turingův stroj

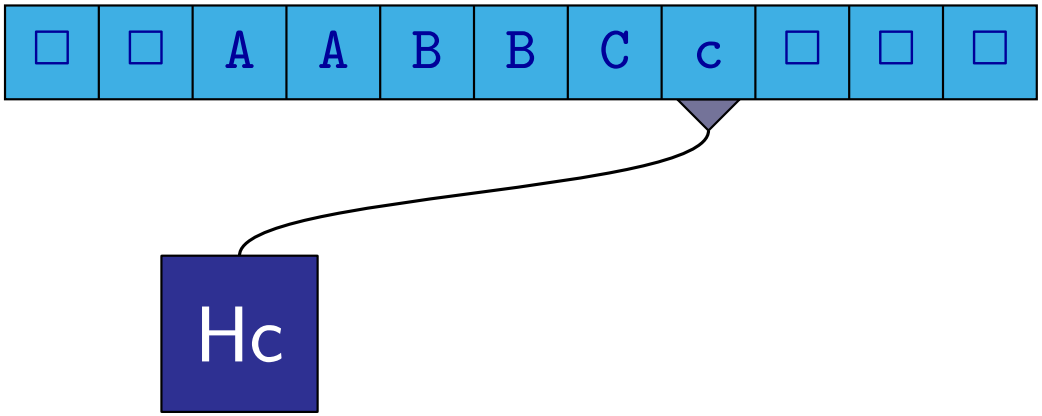


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď NE.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

Turingův stroj

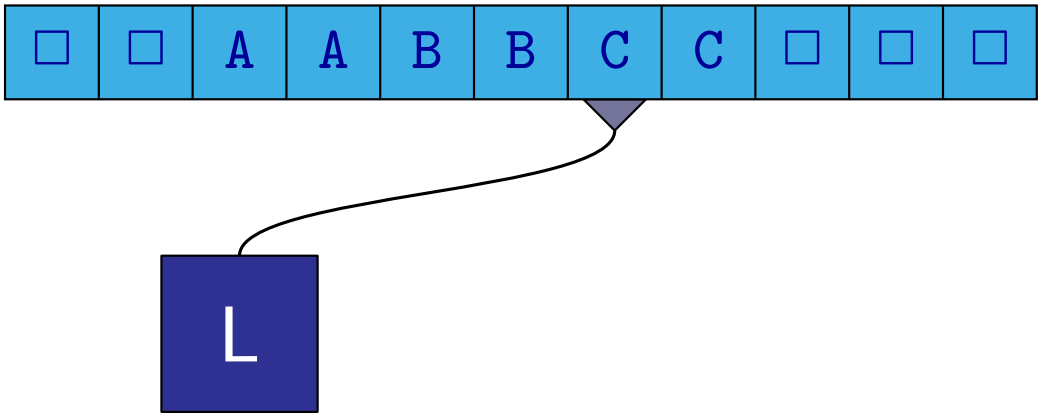


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď NE.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

Turingův stroj

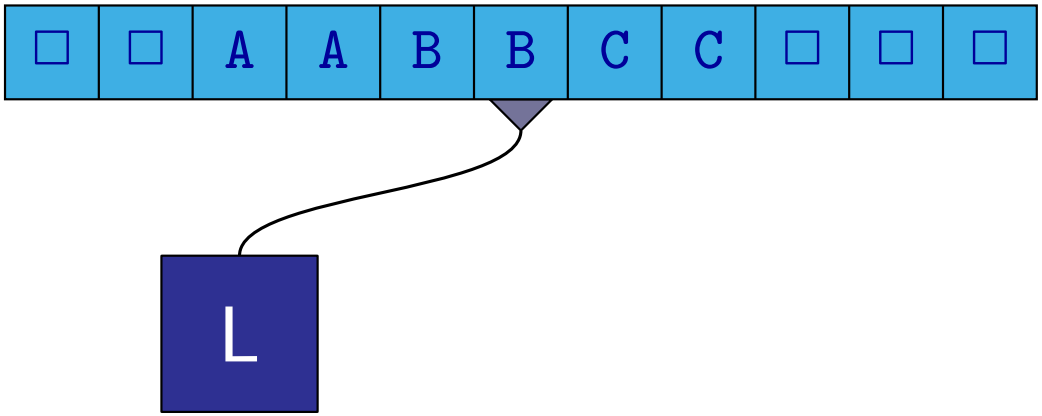


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď NE.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

Turingův stroj

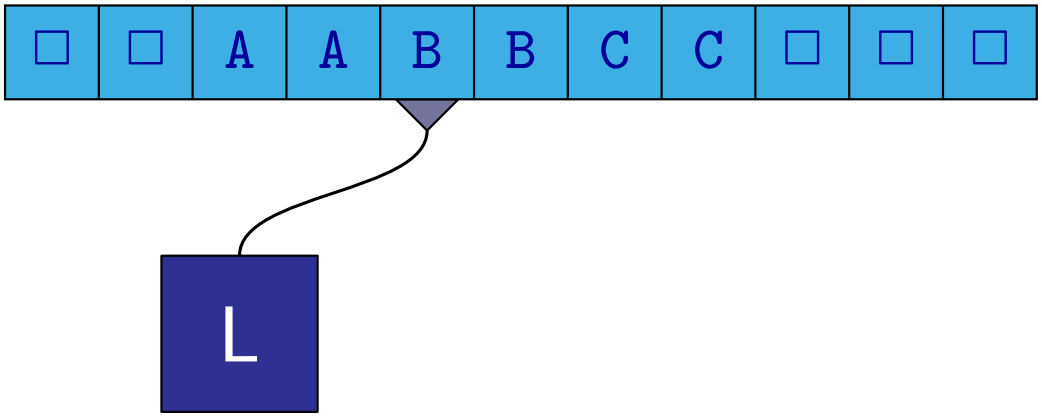


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď NE.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

Turingův stroj

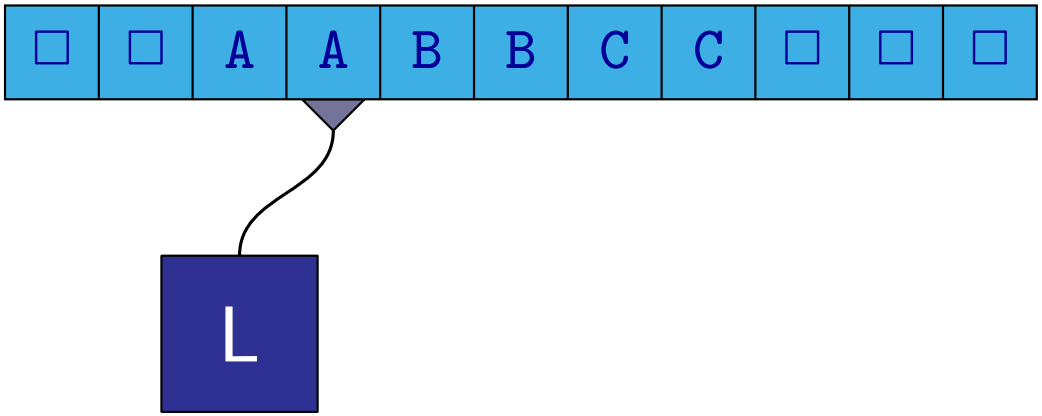


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď NE.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

Turingův stroj

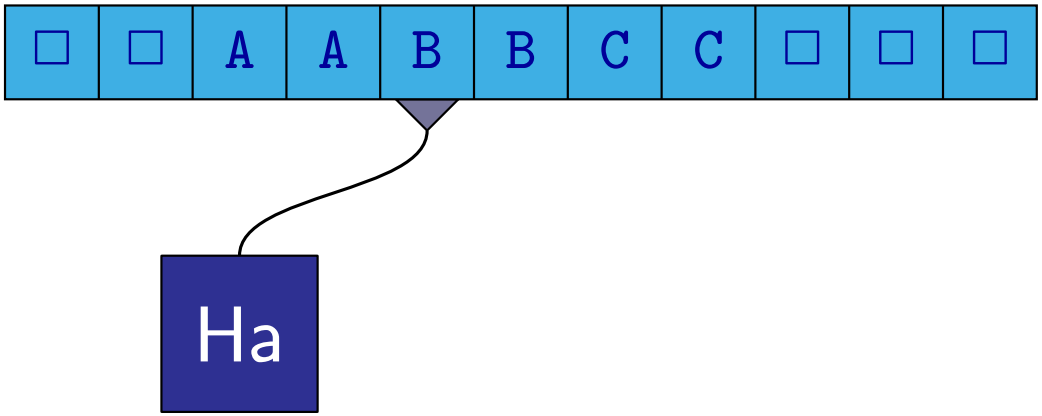


Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE.

2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď NE.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

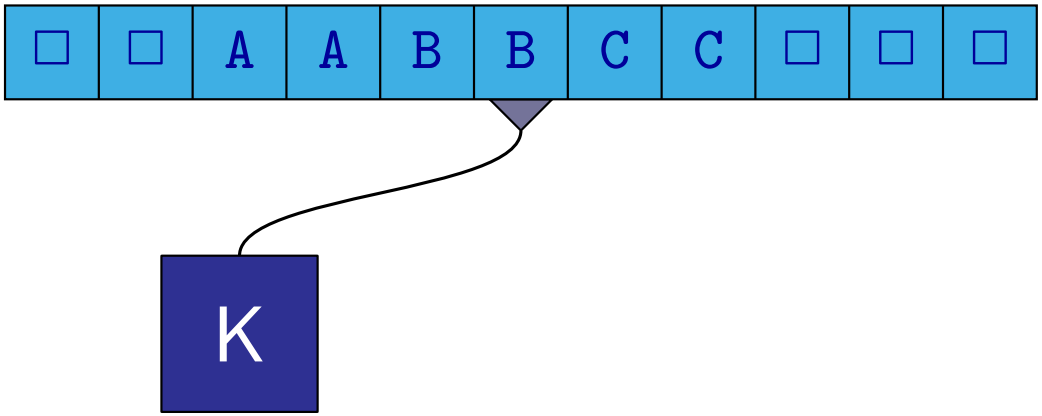
Turingův stroj



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

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď Ne. Pokud už `a` není, zkontroluje, zda už jsou jen velká písmena.
2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď Ne.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď Ne.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

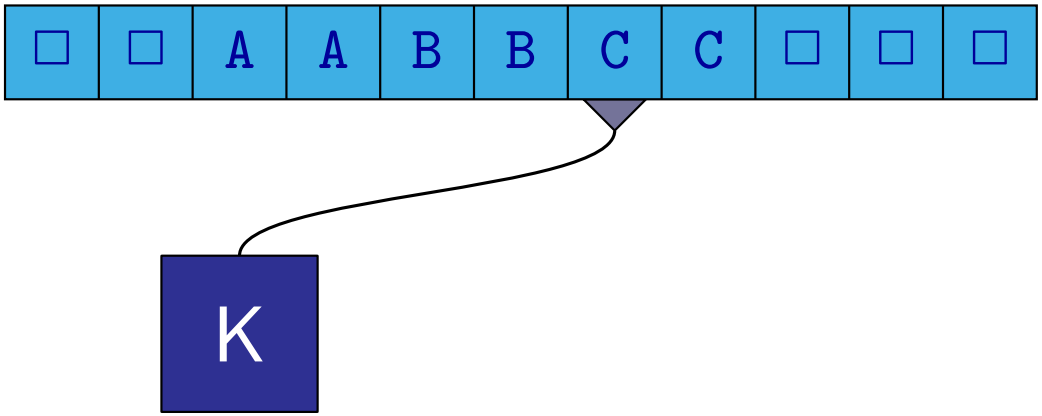
Turingův stroj



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

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď Ne. Pokud už `a` není, zkontroluje, zda už jsou jen velká písmena.
2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď Ne.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď Ne.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

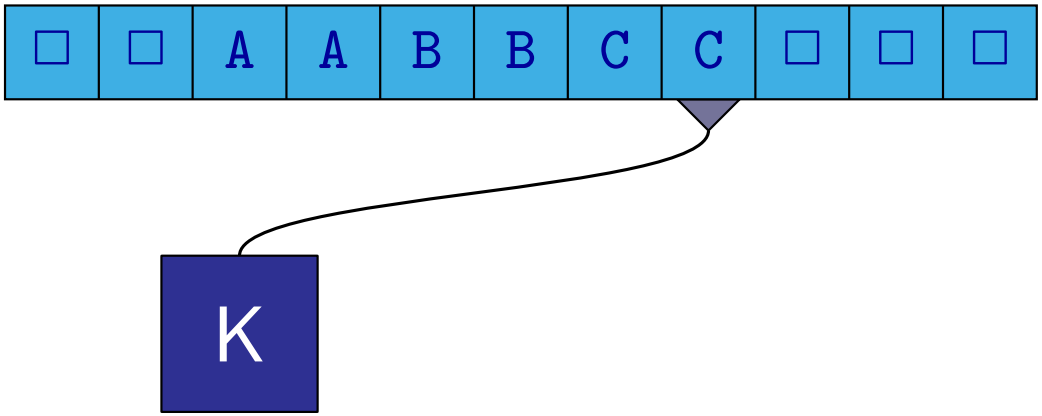
Turingův stroj



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

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď Ne. Pokud už `a` není, zkontroluje, zda už jsou jen velká písmena.
2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď Ne.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď Ne.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

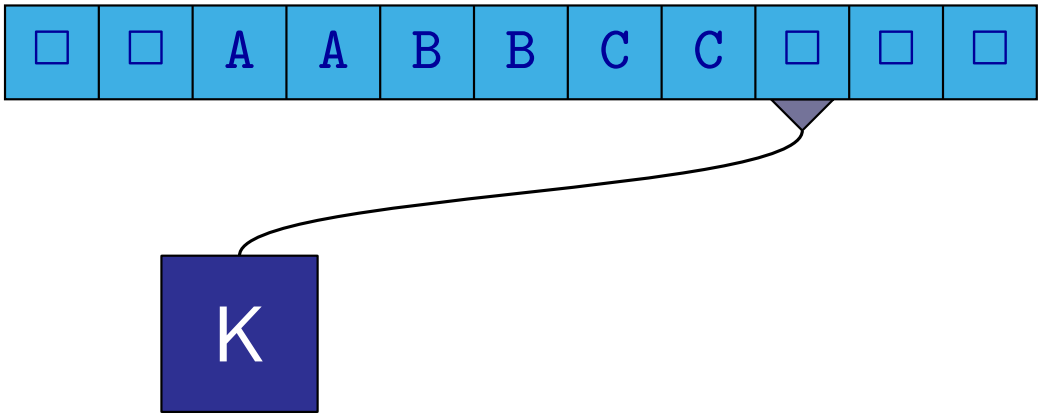
Turingův stroj



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

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď Ne. Pokud už `a` není, zkontroluje, zda už jsou jen velká písmena.
2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď Ne.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď Ne.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

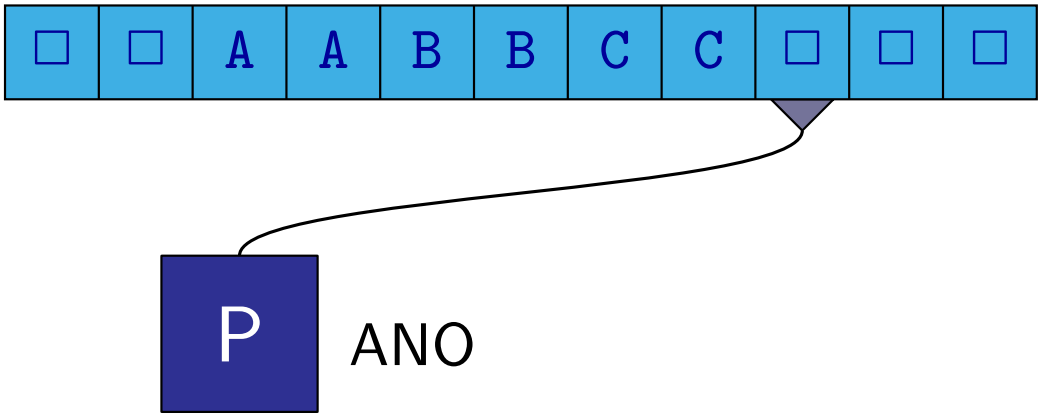
Turingův stroj



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

1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď Ne. Pokud už `a` není, zkontroluje, zda už jsou jen velká písmena.
2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď Ne.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď Ne.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

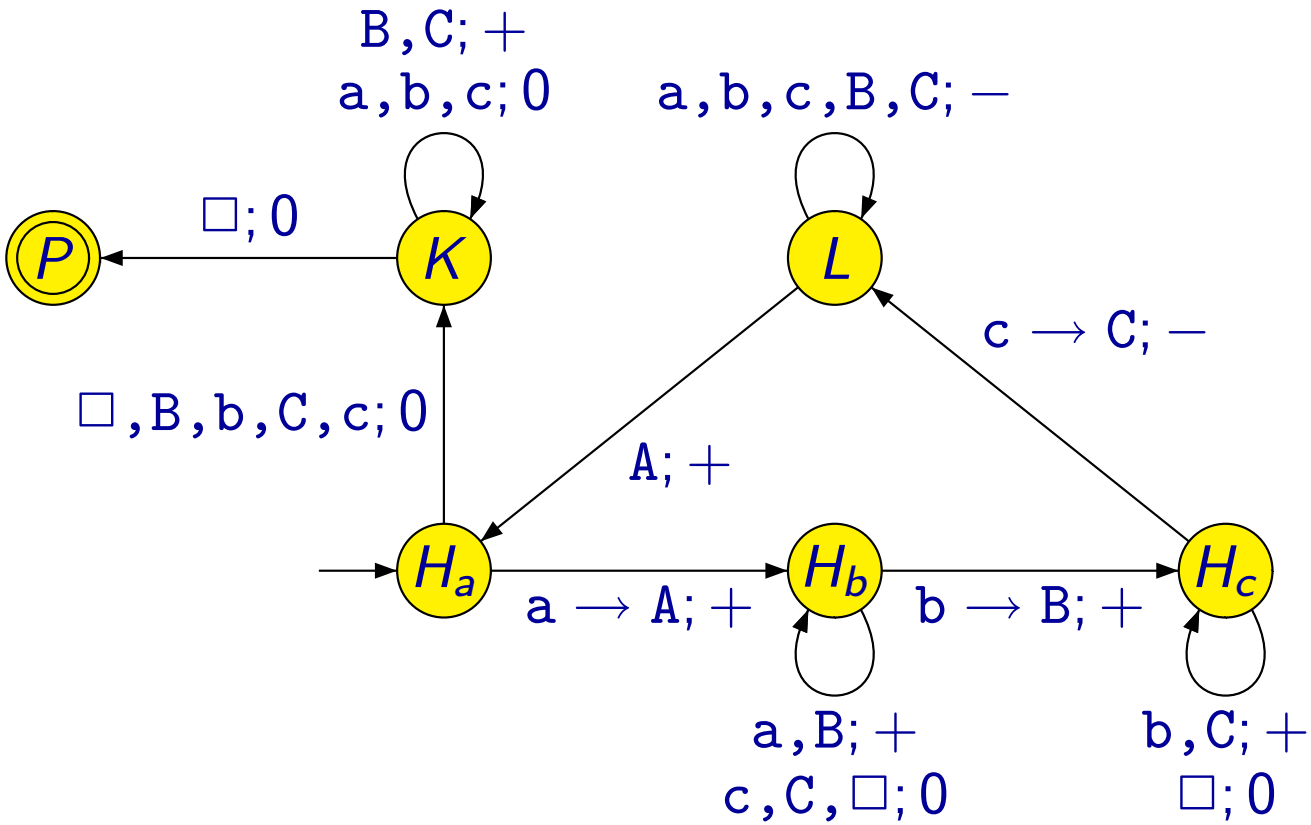
Turingův stroj



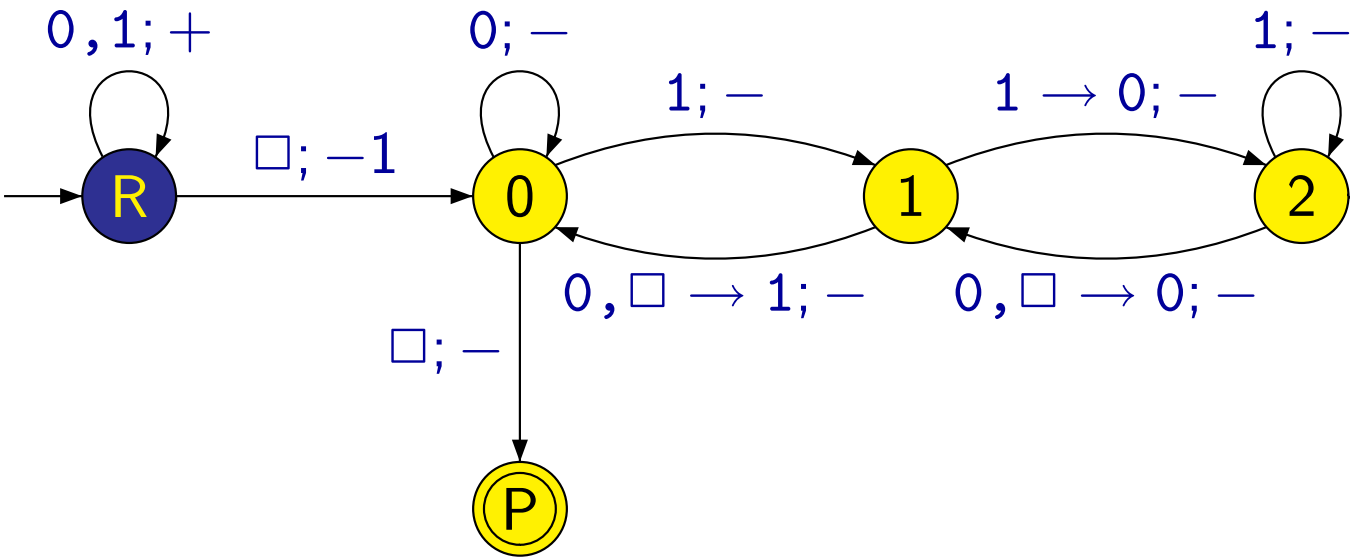
Jazyk $L = \{a^i b^i c^i \mid i \geq 0\}$

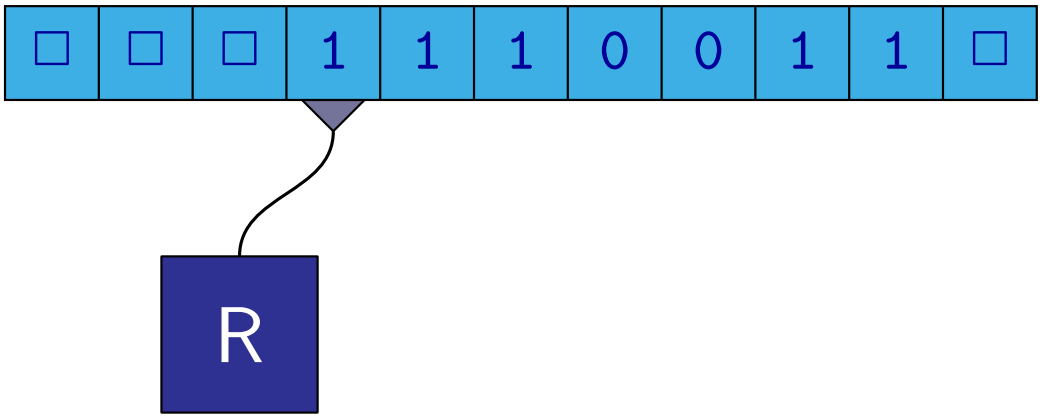
1. Čte do prvního `a`, nahradí jej `A`. Pokud najde dříve `b` nebo `c`, dá odpověď NE. Pokud už `a` není, zkontroluje, zda už jsou jen velká písmena.
2. Čte do prvního `b`, nahradí jej `B`. Na □ nebo `c` dá odpověď NE.
3. Čte do prvního `c`, nahradí jej `C`. Na □ dá odpověď NE.
4. Vrací se doleva na nejbližší `A`.
5. Pokračuje bodem 1

Turingův stroj

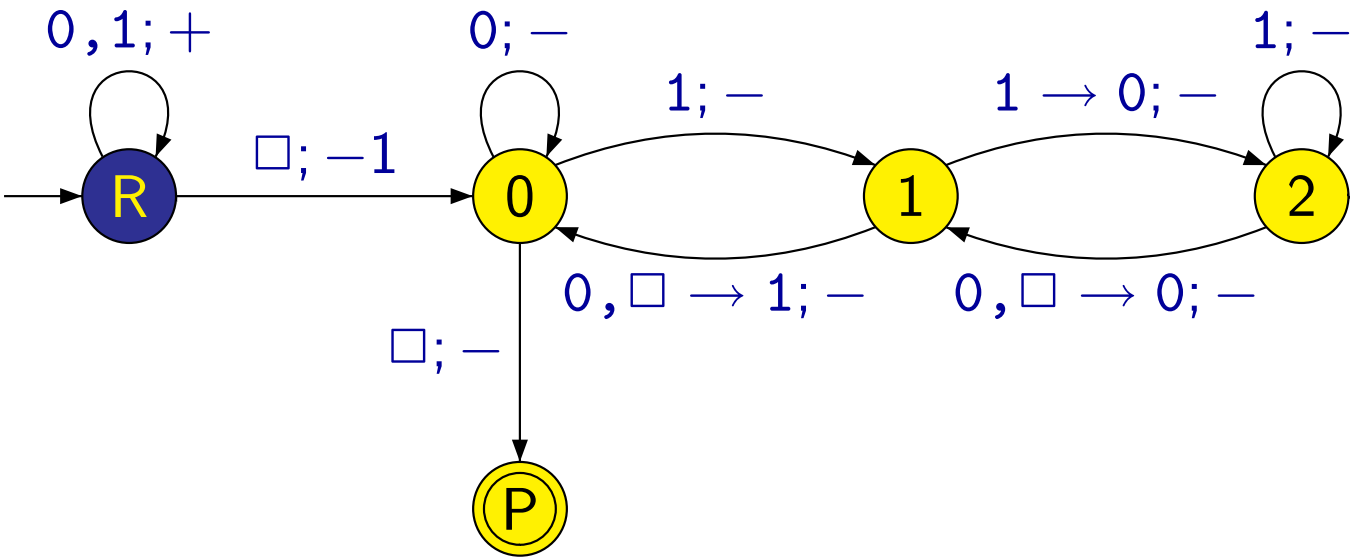


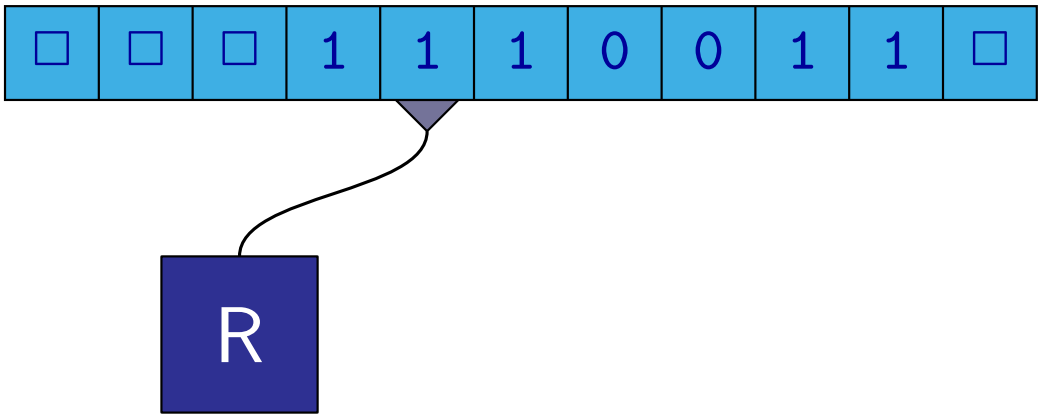
Turingův stroj – násobení třemi



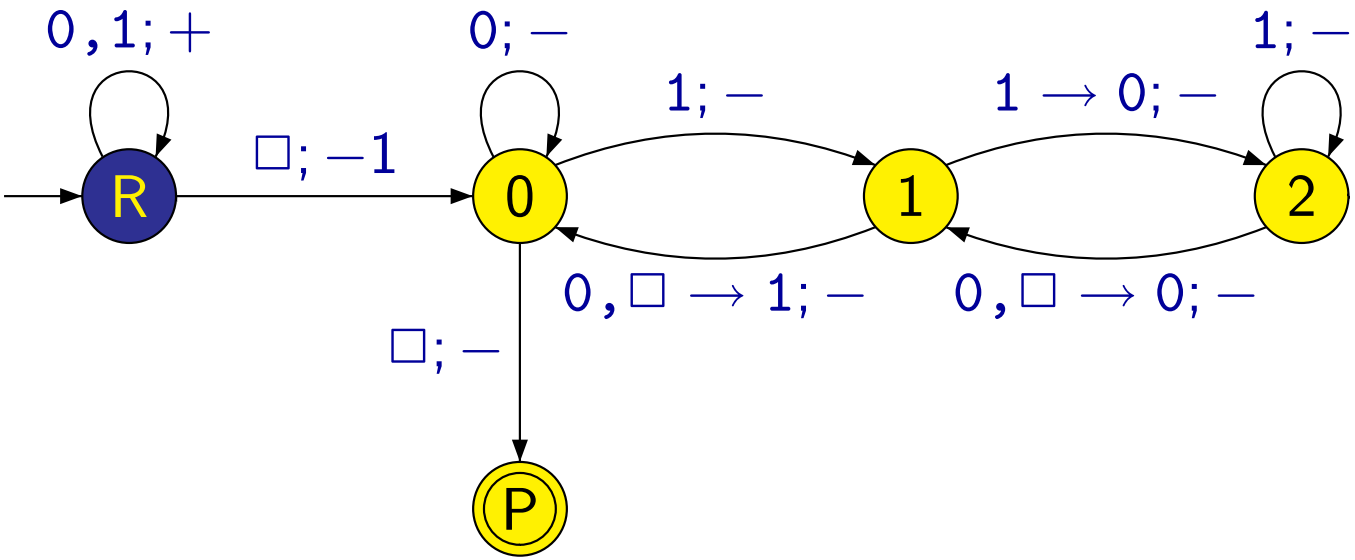


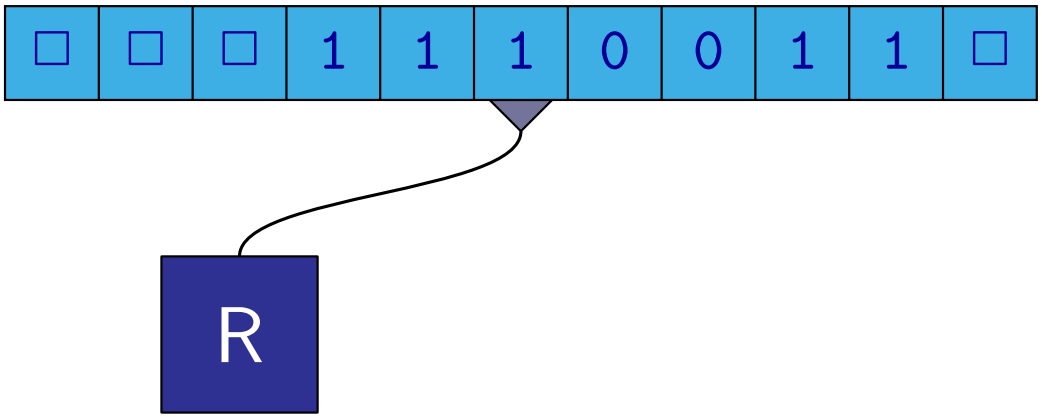
Turingův stroj – násobení třemi



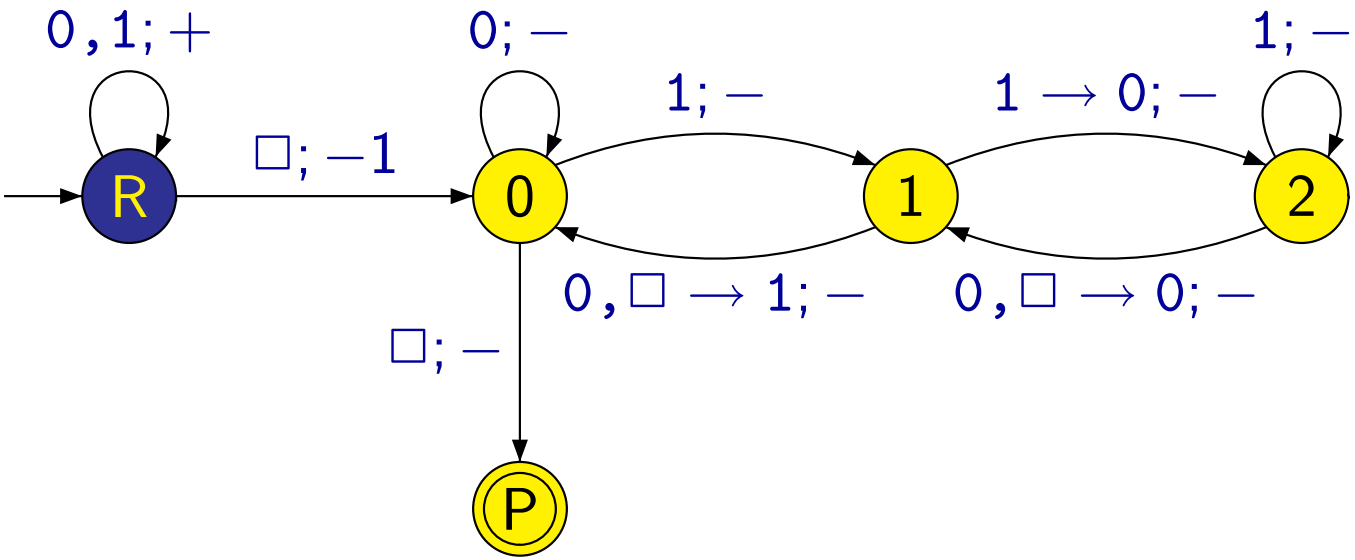


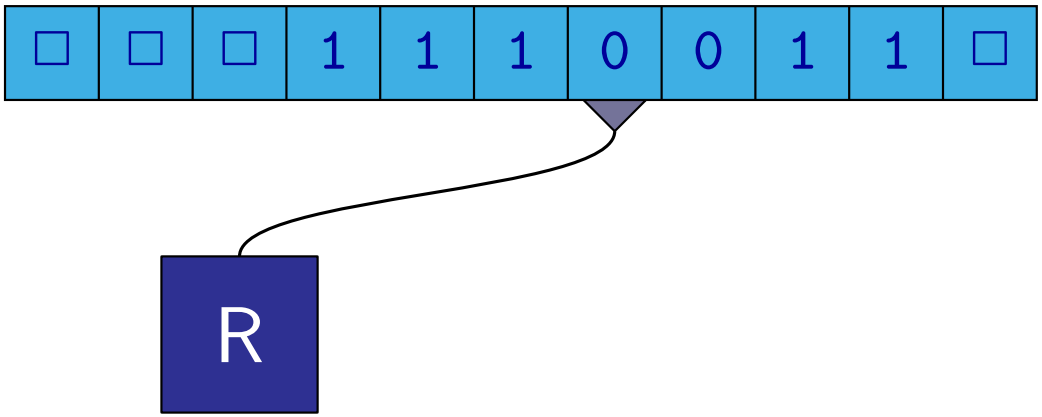
Turingův stroj – násobení třemi



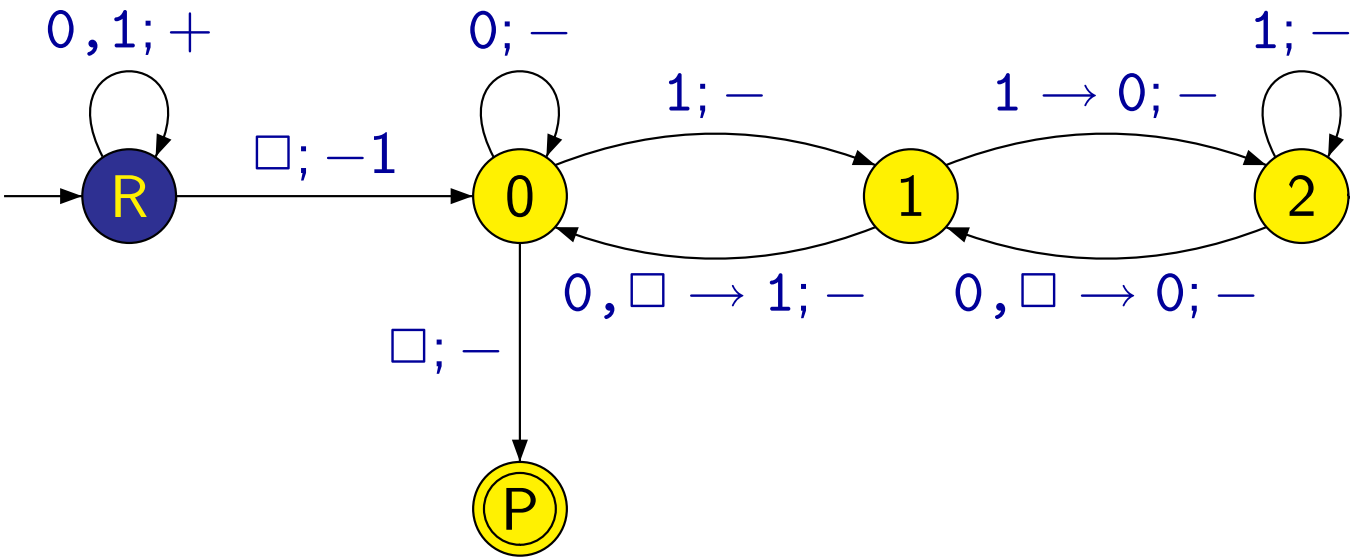


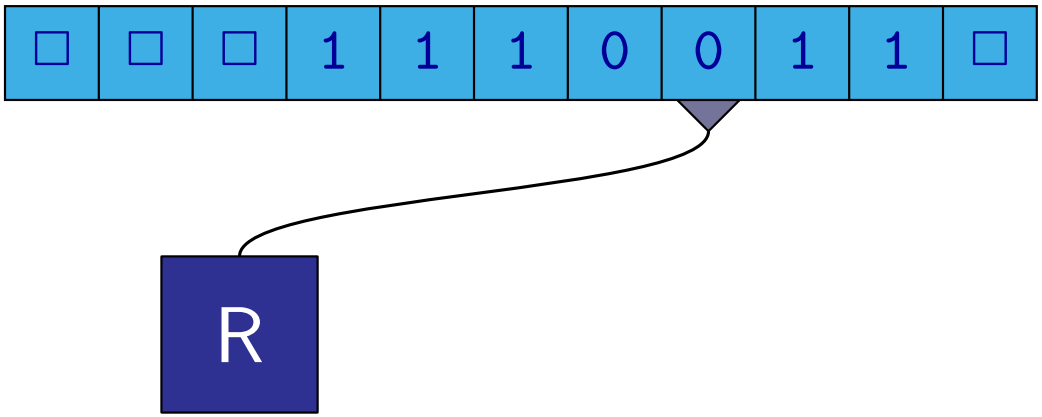
Turingův stroj – násobení třemi



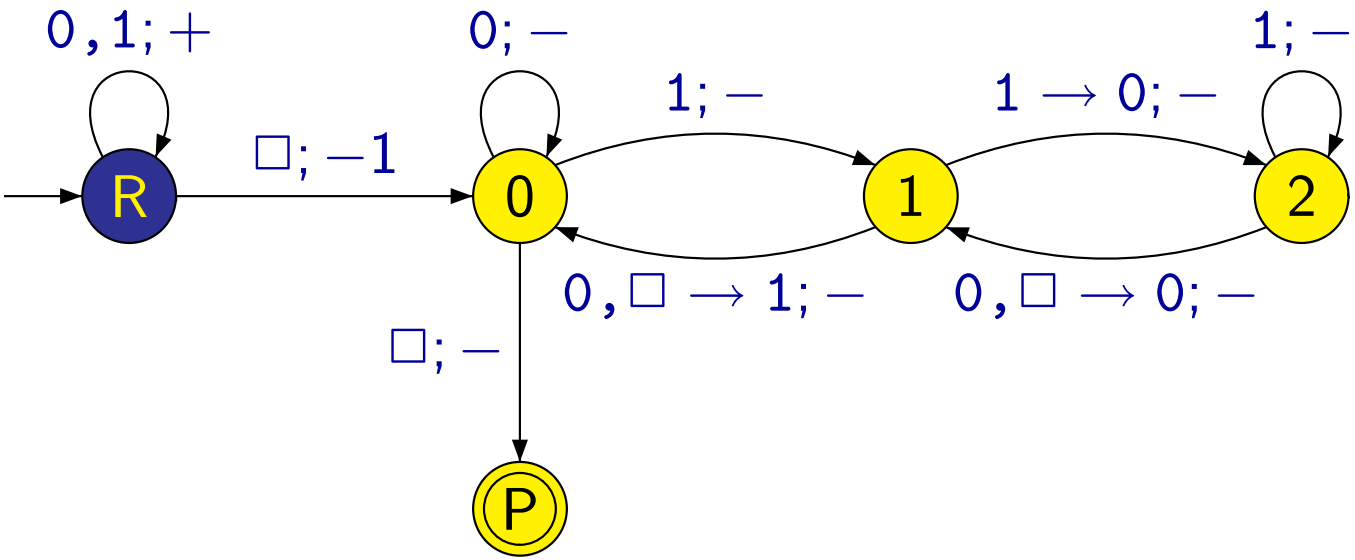


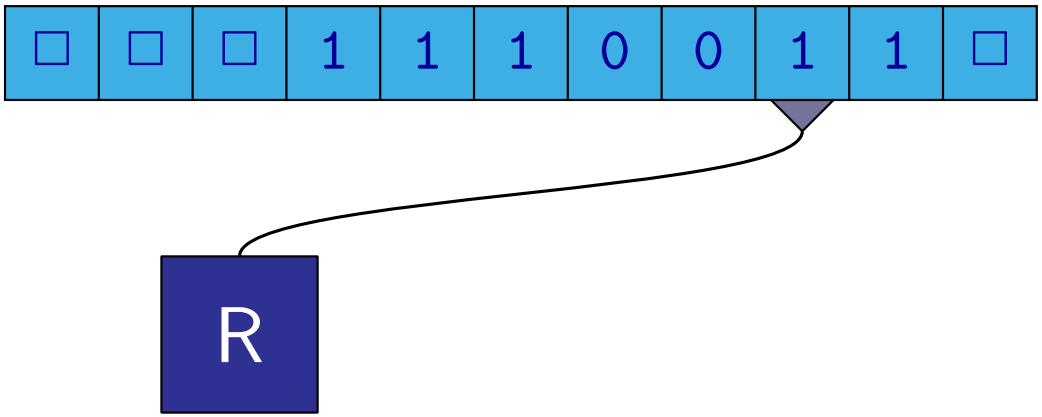
Turingův stroj – násobení třemi



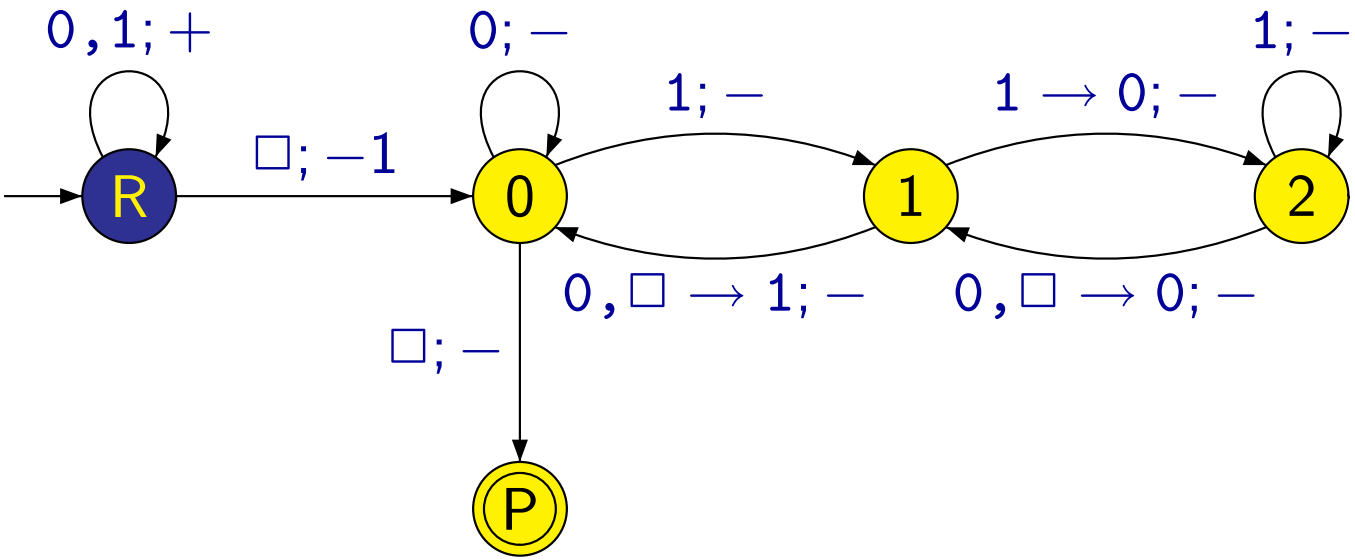


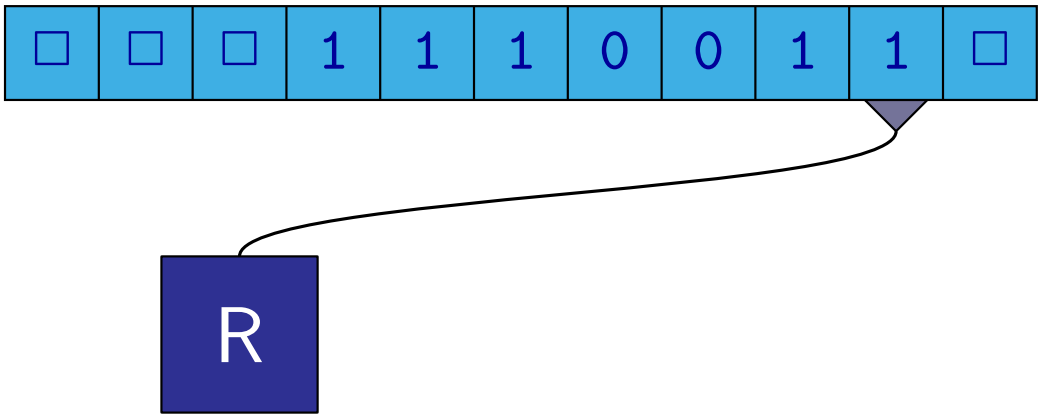
Turingův stroj – násobení třemi



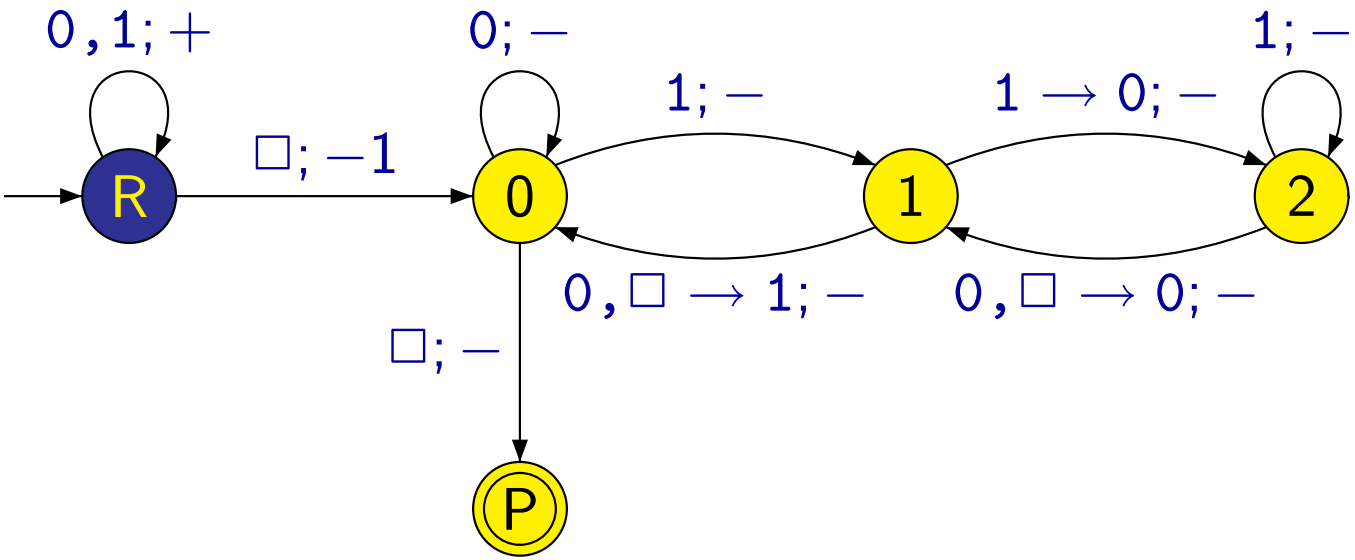


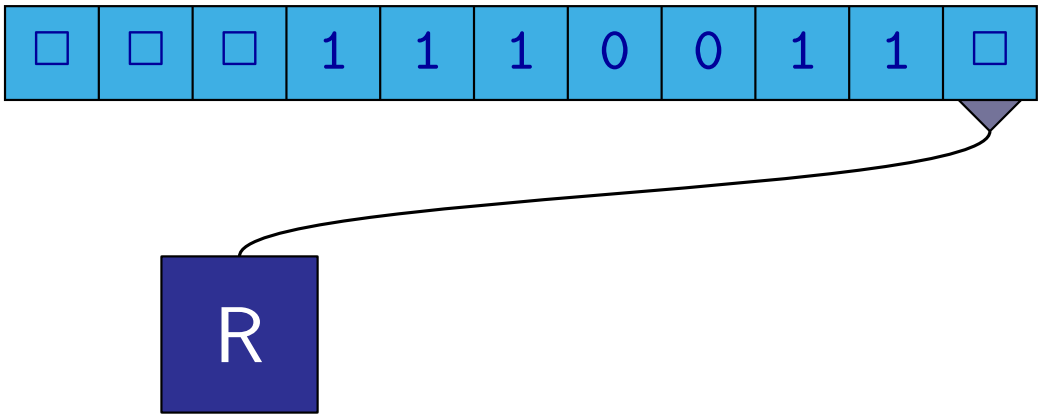
Turingův stroj – násobení třemi



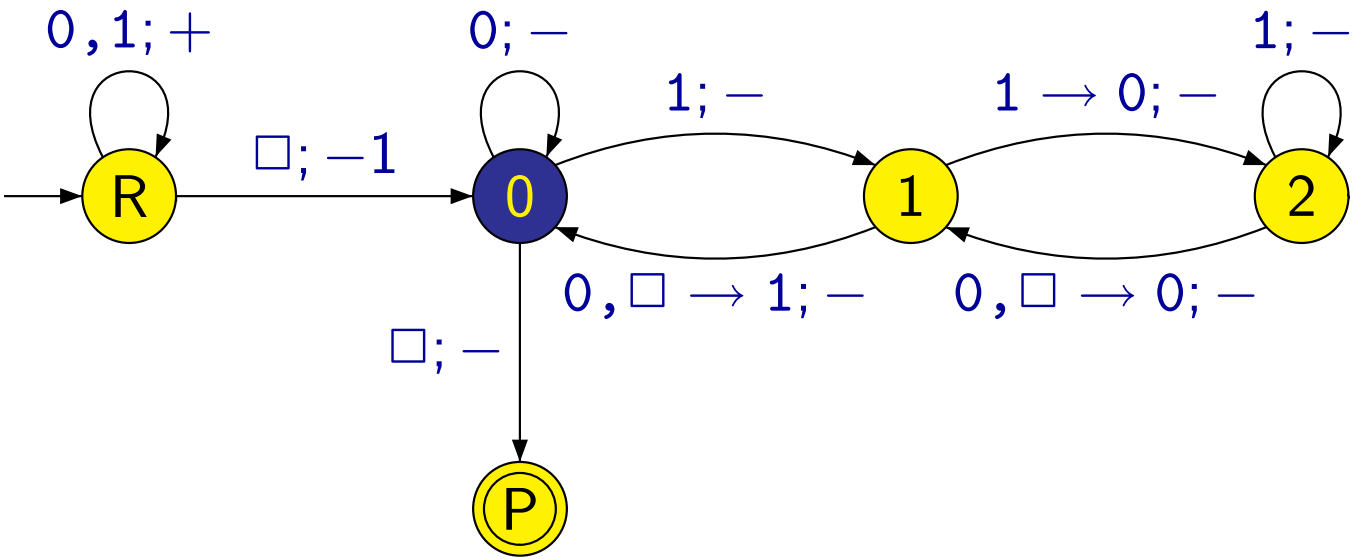


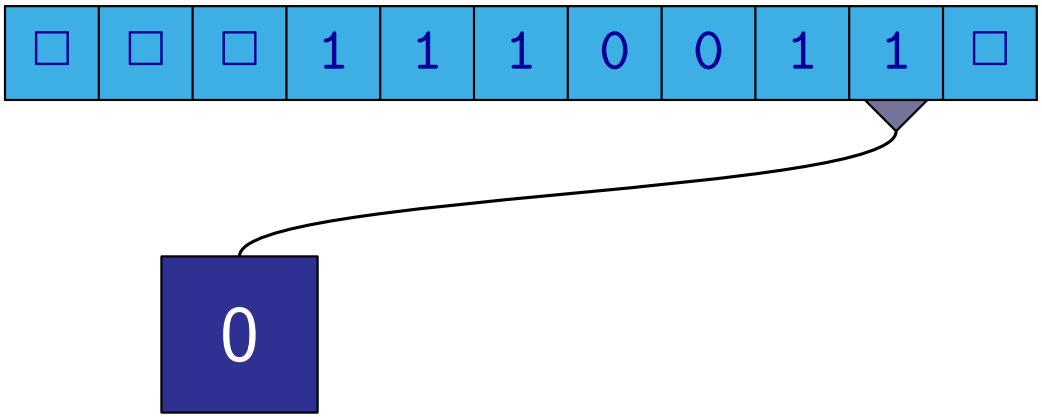
Turingův stroj – násobení třemi



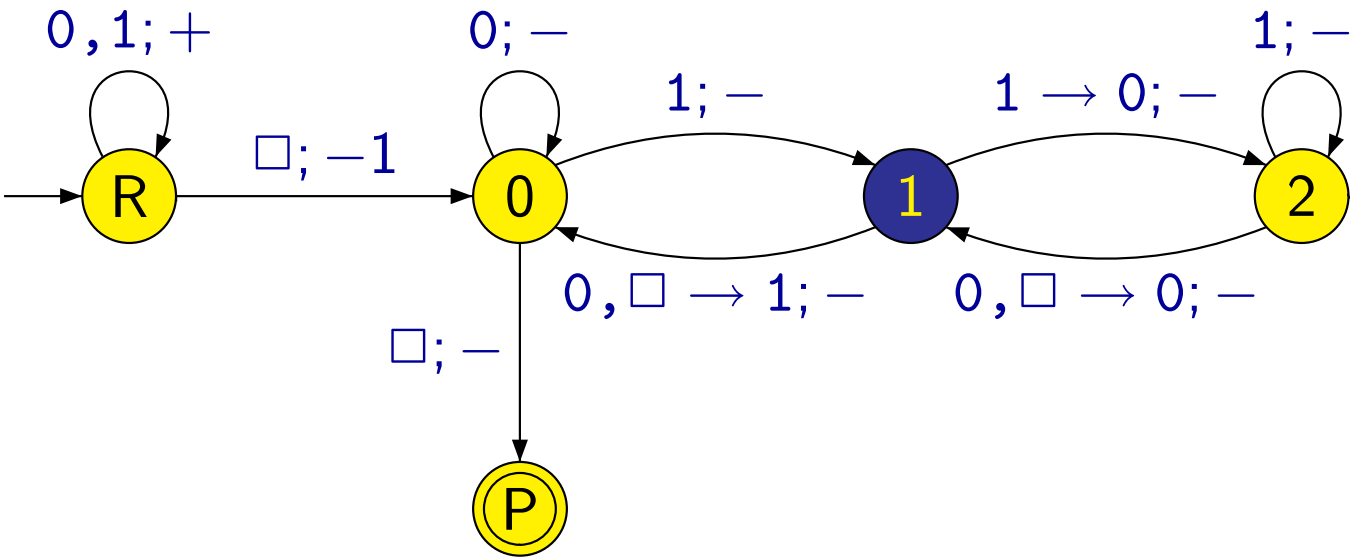


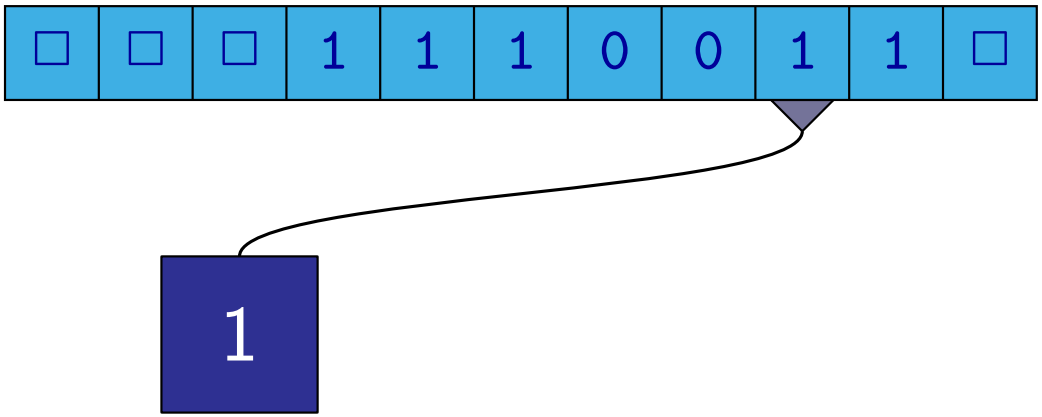
Turingův stroj – násobení třemi



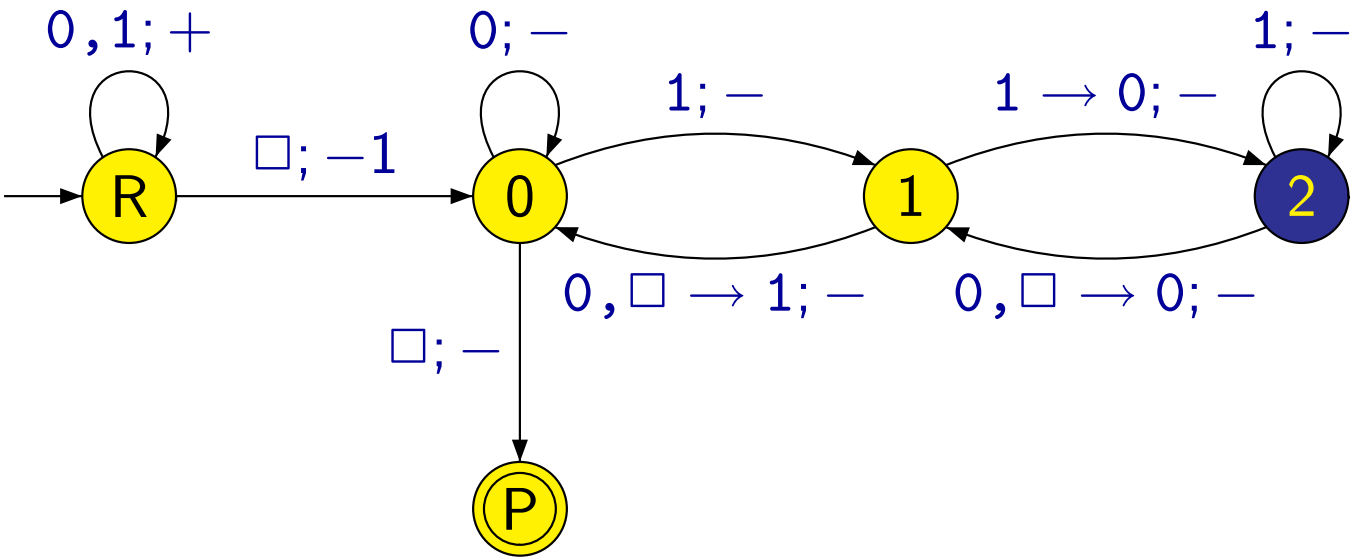


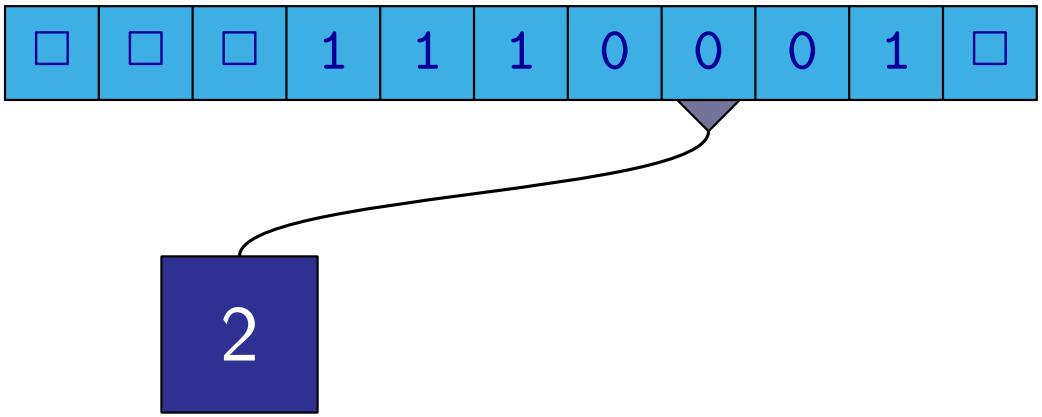
Turingův stroj – násobení třemi



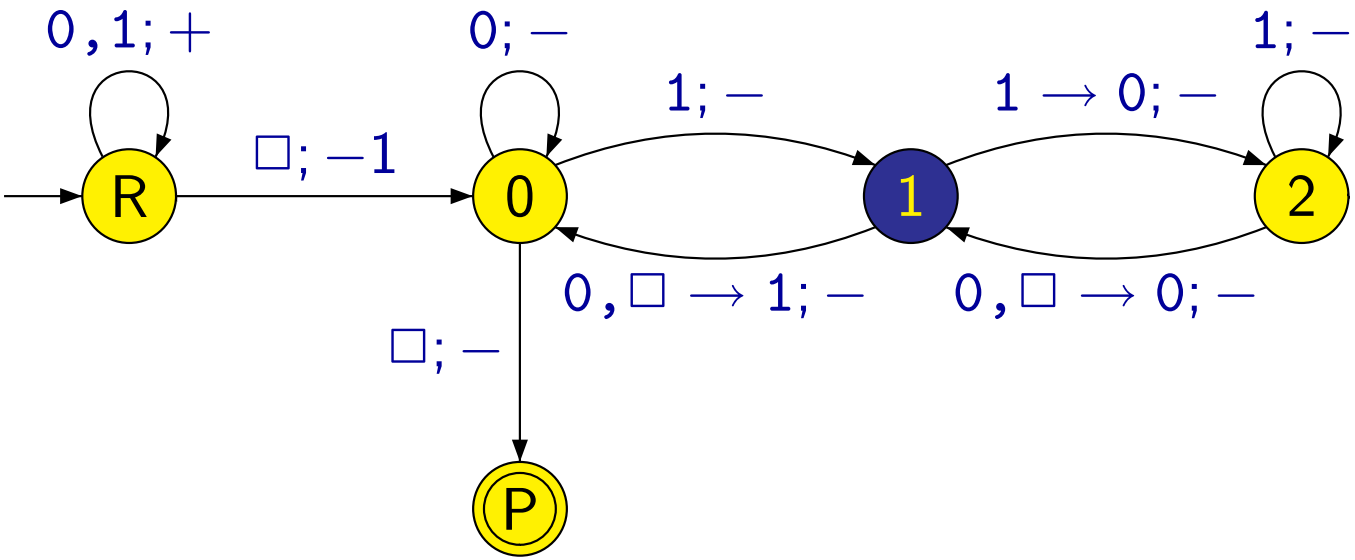


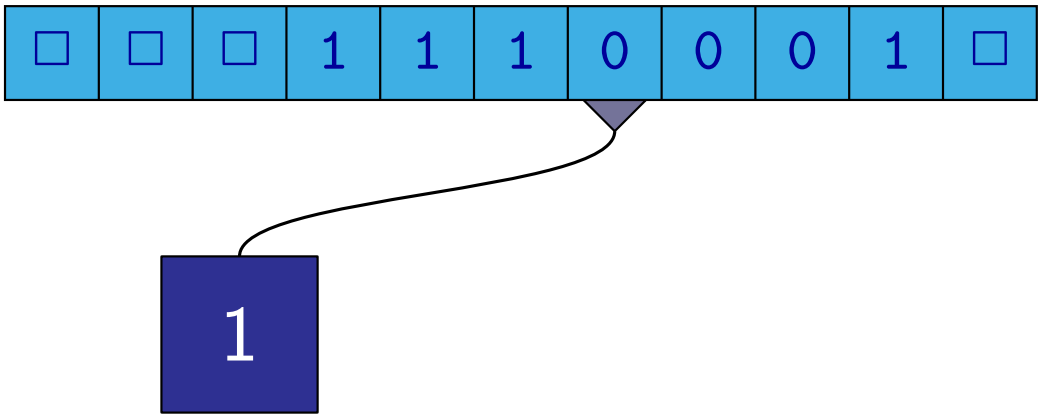
Turingův stroj – násobení třemi



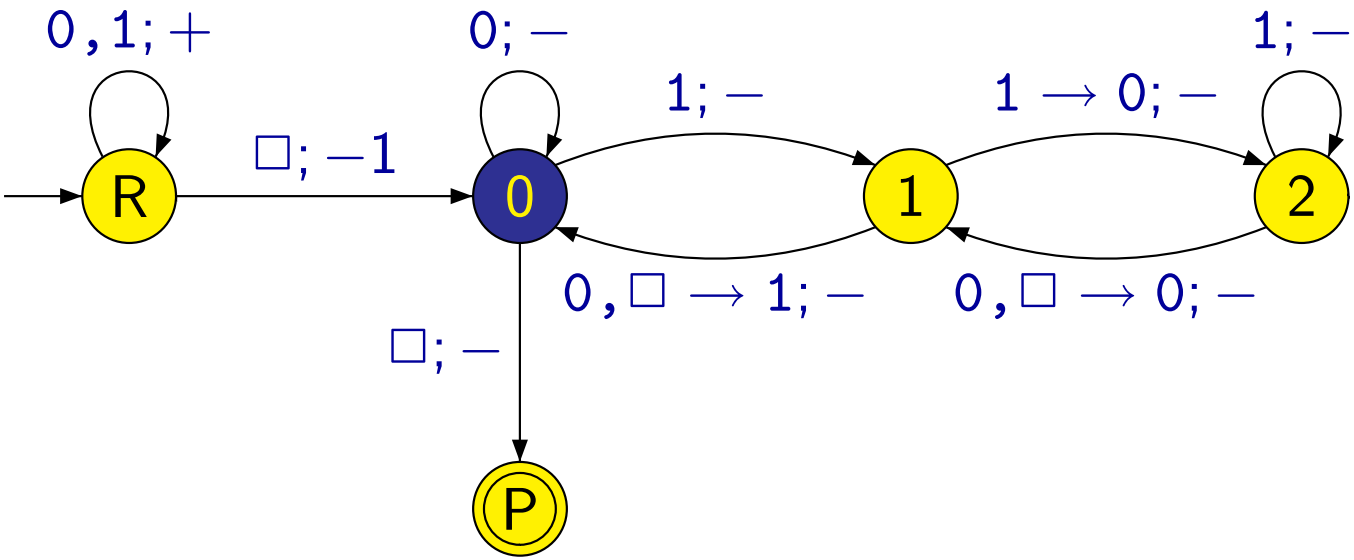


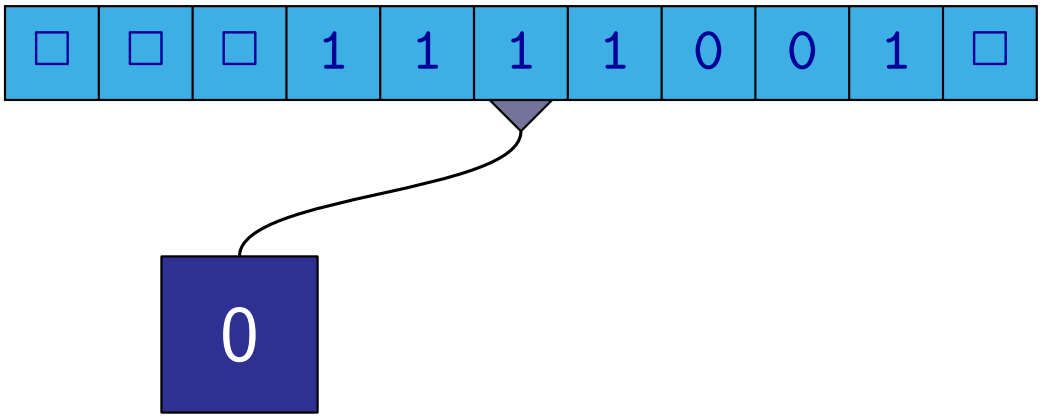
Turingův stroj – násobení třemi



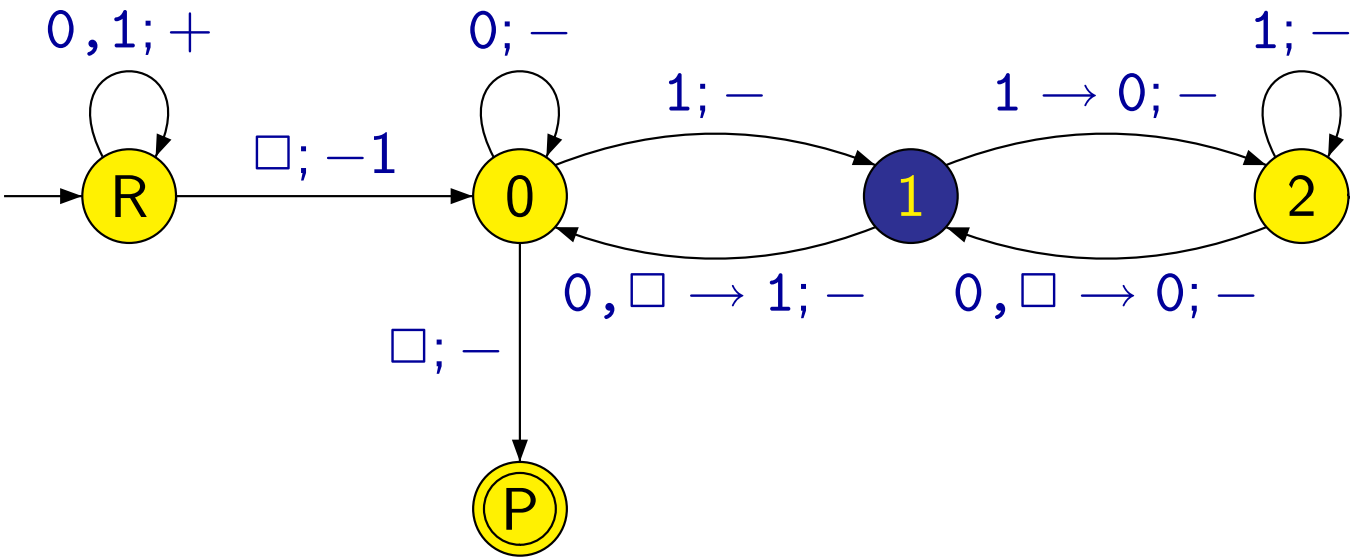


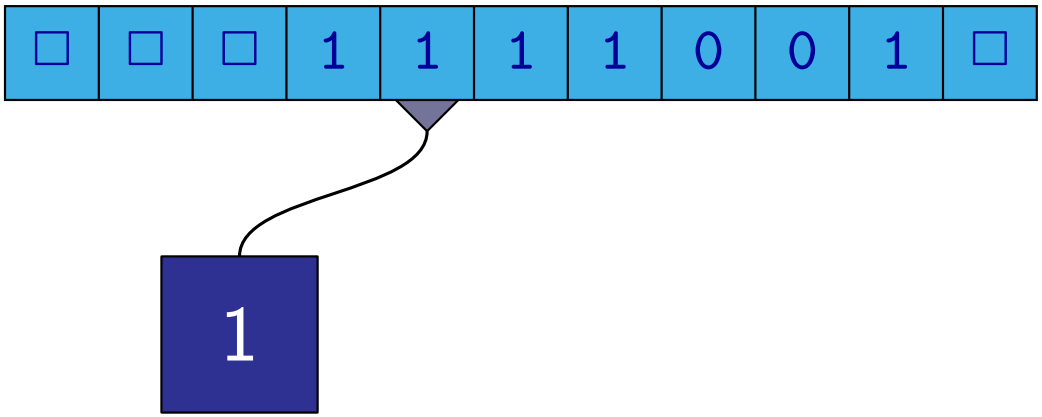
Turingův stroj – násobení třemi



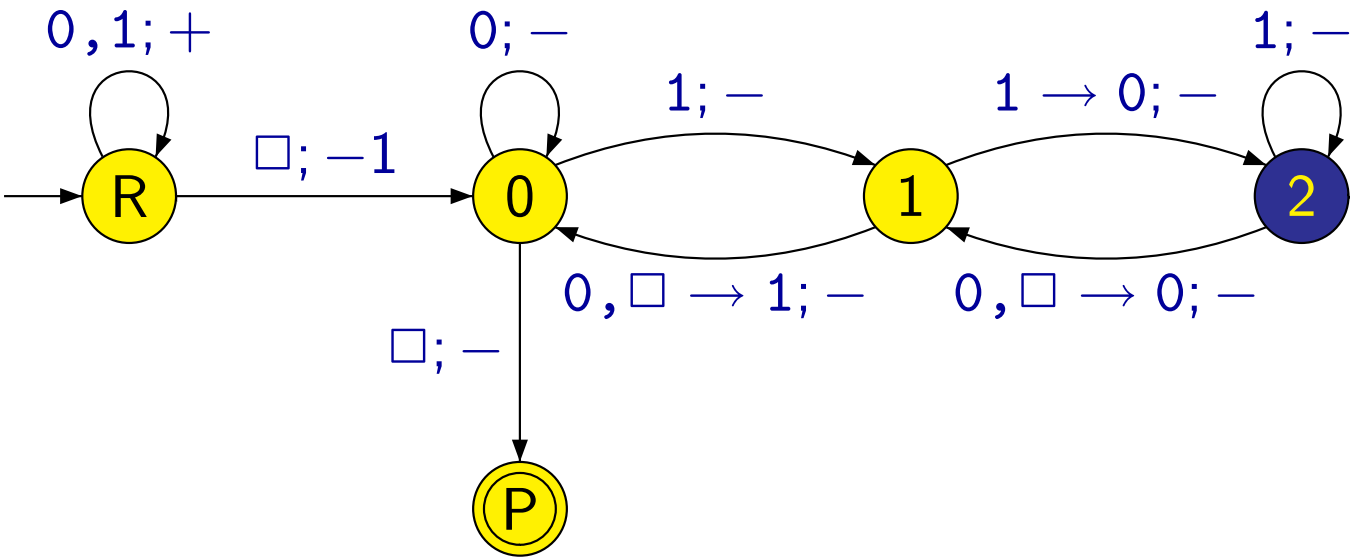


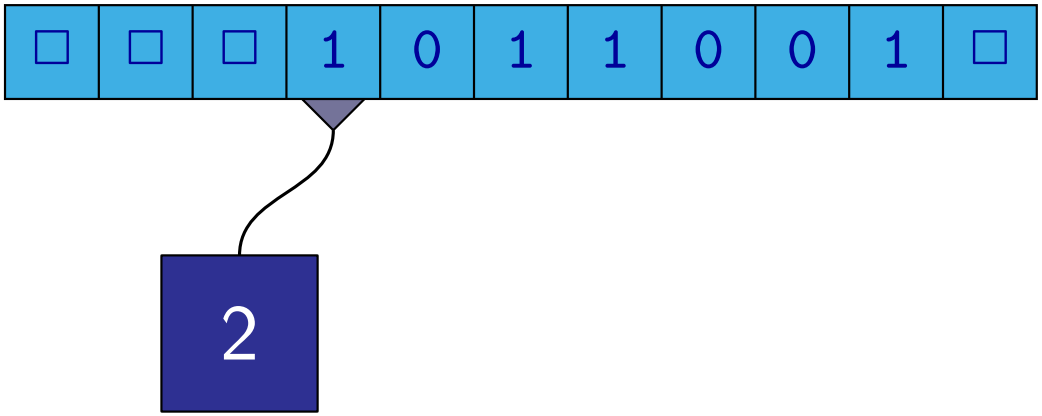
Turingův stroj – násobení třemi



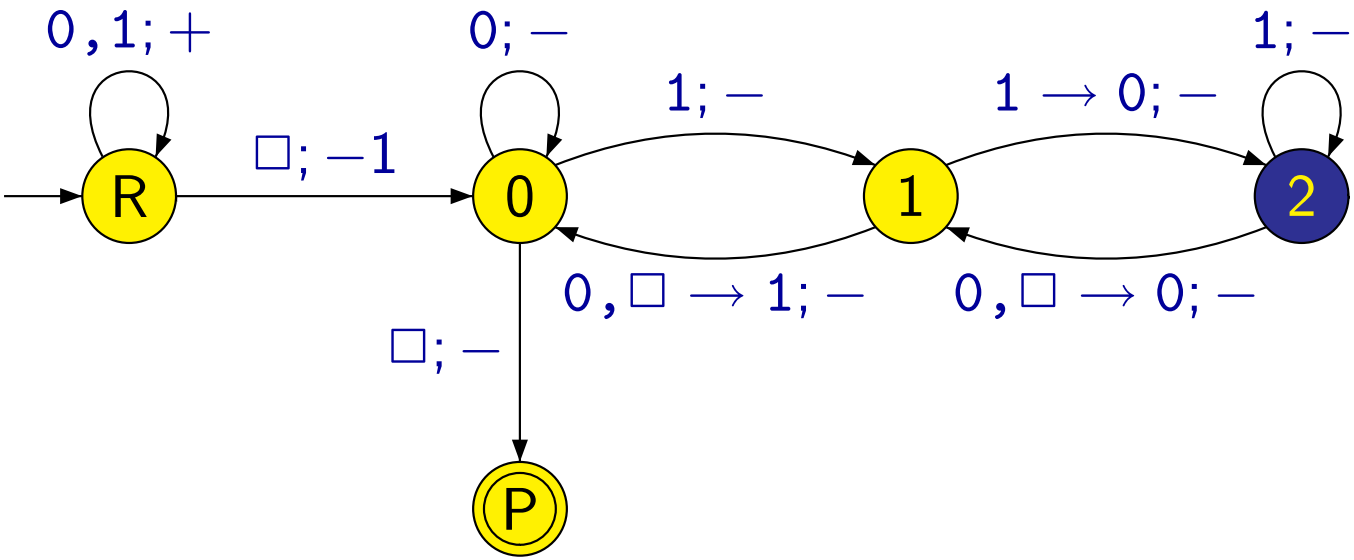


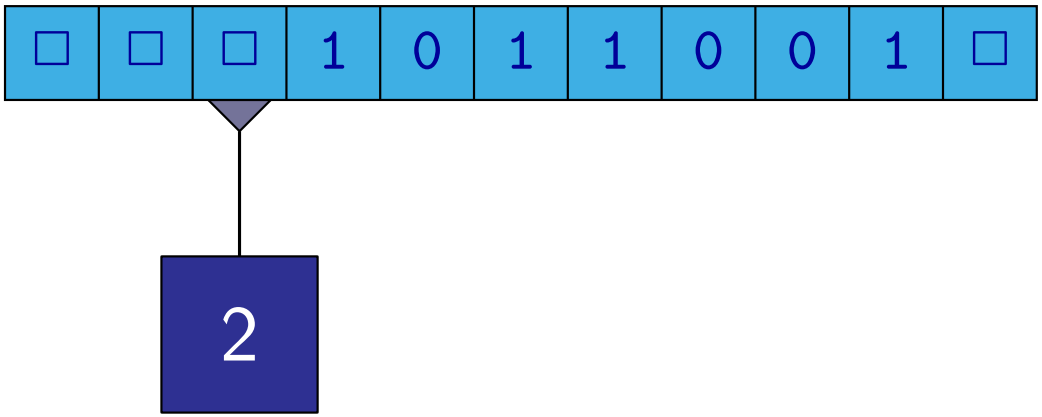
Turingův stroj – násobení třemi



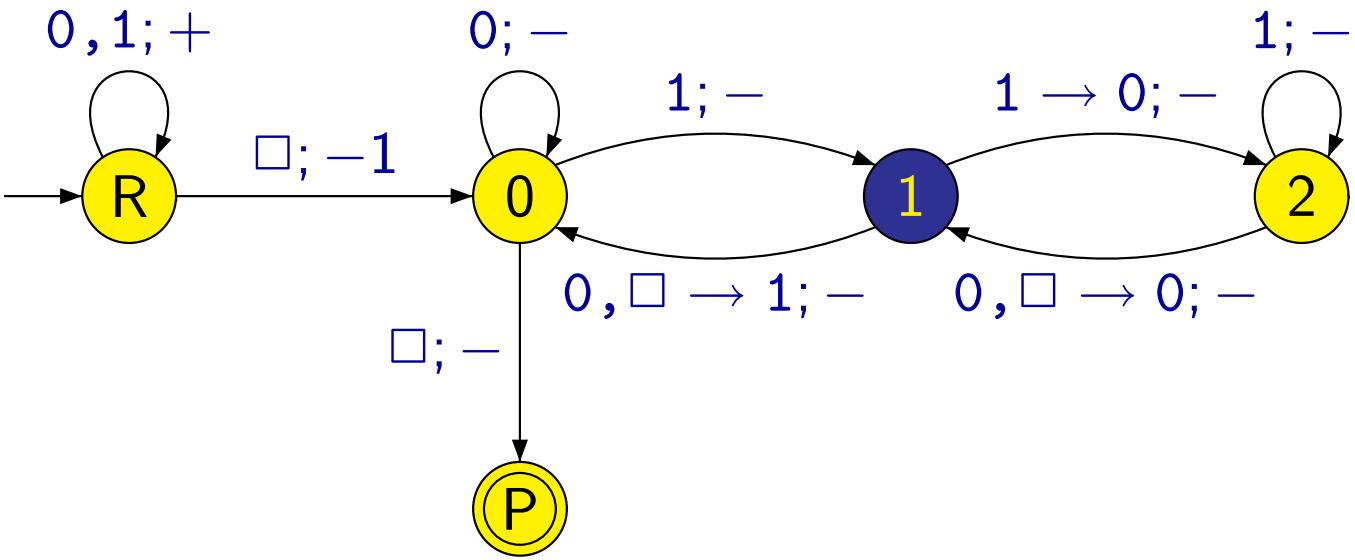


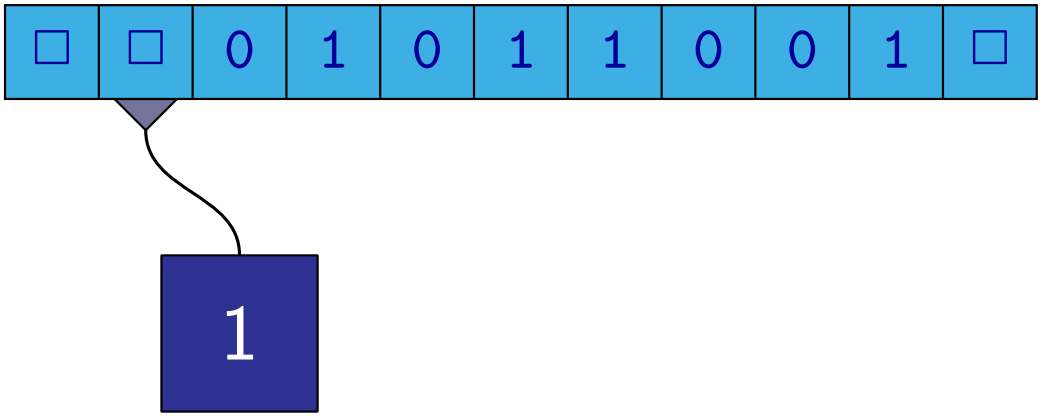
Turingův stroj – násobení třemi



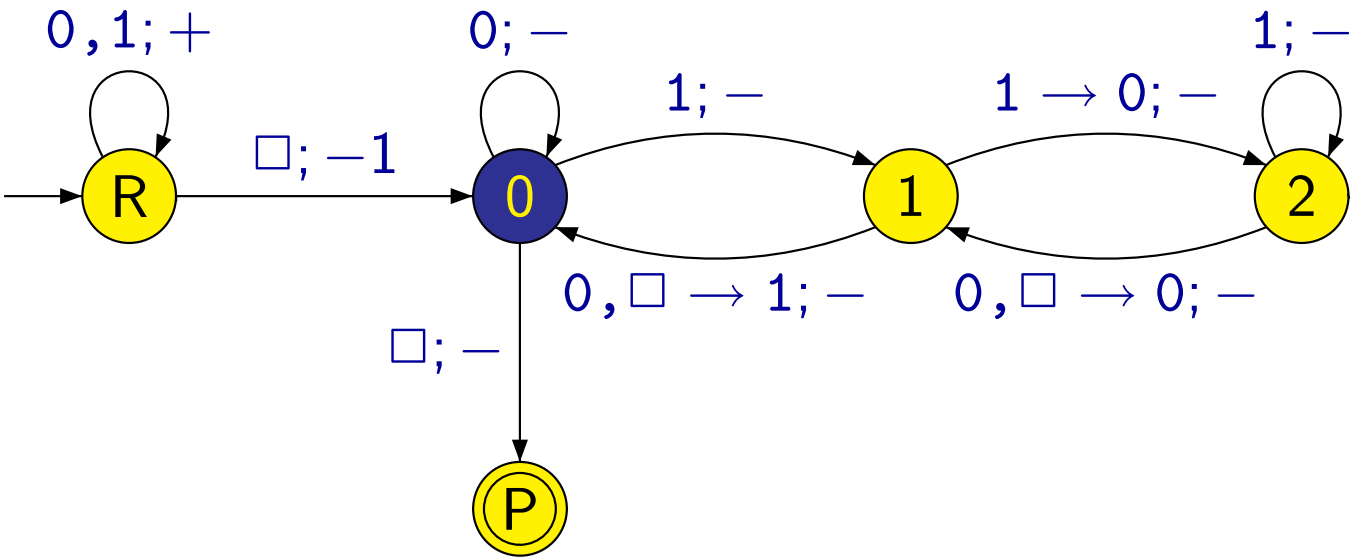


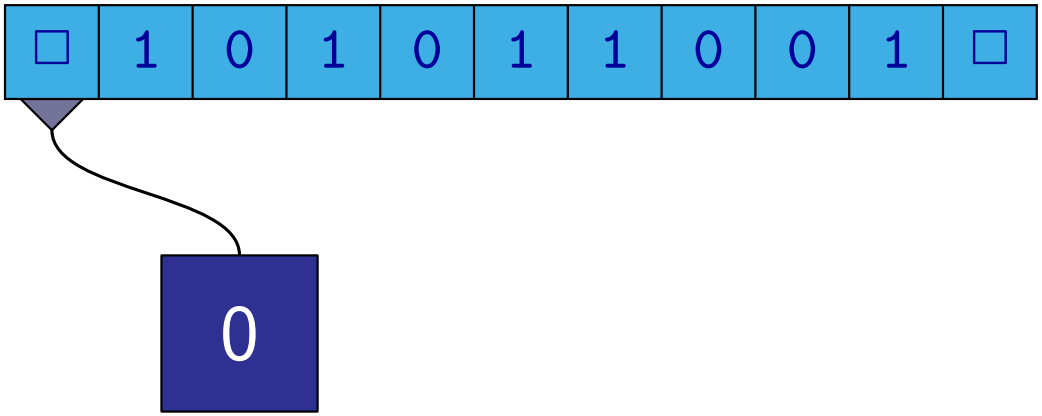
Turingův stroj – násobení třemi



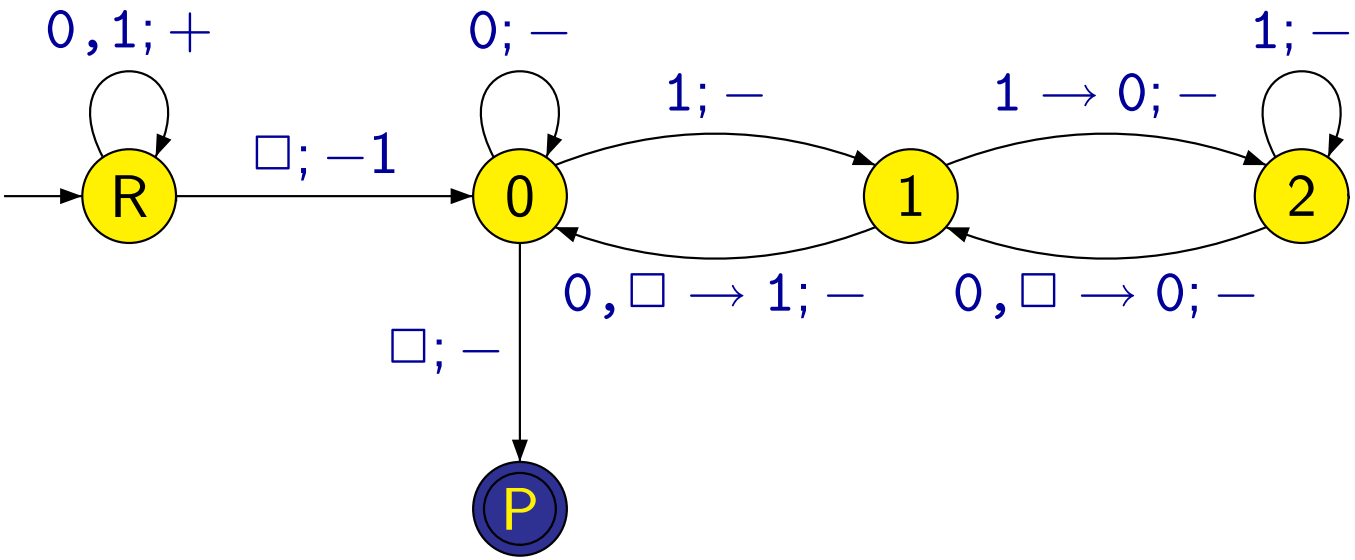


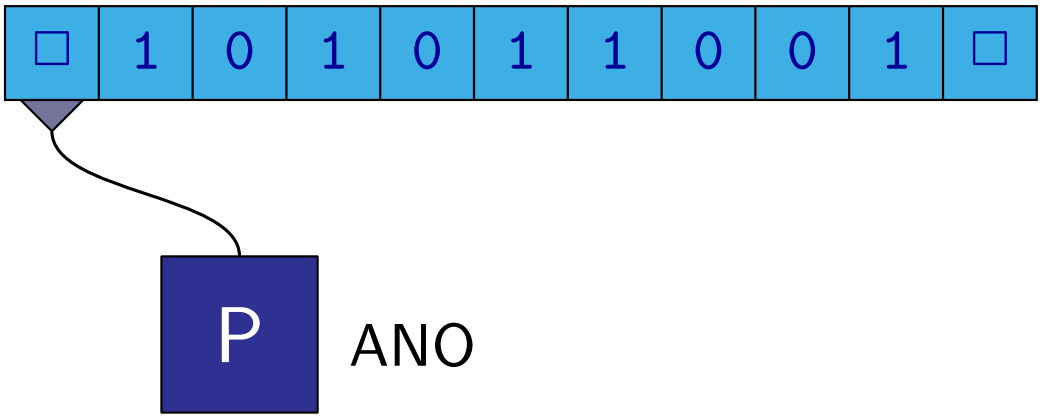
Turingův stroj – násobení třemi





Turingův stroj – násobení třemi





Lineárně omezený automat

Lineárně omezený automat:

- Turingův stroj, který využívá jen úsek pásky, na kterém je zapsáno vstupní slovo.
- Představa levé a pravé zarážky kolem slova, které nemohou být přepsány.
- Z levé zarážky je možný pohyb jen vpravo, z pravé zarážky jen vlevo.
- LBA (linear bounded automata) se uvažují v nedeterministické verzi.
- Je otevřená otázka, jestli jsou deterministické LBA stejně „silné“ jako nedeterministické.

Generativní gramatika

Definice

Generativní gramatika je dána čtveřicí parametrů $G = (\Pi, \Sigma, S, P)$, kde

- Π je konečná množina neterminálů
- Σ je konečná množina terminálů, $\Pi \cap \Sigma = \emptyset$
- $S \in \Pi$ je počáteční neterminál
- P je konečná množina pravidel typu $\alpha \to \beta$, kde $\alpha \in (\Pi \cup \Sigma)^* \Pi (\Pi \cup \Sigma)^*$ a $\beta \in (\Pi \cup \Sigma)^*$.

- $\mu_1 \alpha \mu_2 \Rightarrow_G \mu_1 \beta \mu_2$ pokud $\alpha \to \beta$ je pravidlo v P
- $L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$

Chomského hierarchie

Podle tvaru pravidel, která v gramatice povolíme, dělíme gramatiky na gramatiky typu:

0 - Obecné **generativní gramatiky** - pravidla bez omezení

1 - **Kontextové gramatiky** - pravidla tvaru $\alpha X \beta \to \alpha \gamma \beta$, kde $|\gamma| \geq 1$ (Výjimka $S \to \varepsilon$, ale S pak není na pravé straně žádného pravidla)

2 - **Bezkontextové gramatiky** - pravidla tvaru $X \to \gamma$

3 - **Regulární gramatiky** - pravidla tvaru $X \to wY$ nebo $X \to w$

$$\alpha, \beta, \gamma \in (\Pi \cup \Sigma)^*, X \in \Pi, w \in \Sigma^*$$

Chomského hierarchie

Podle tvaru pravidel, která v gramatice povolíme, dělíme gramatiky na gramatiky typu:

0 - Obecné **generativní gramatiky** - pravidla bez omezení

1 - **Kontextové gramatiky** - pravidla tvaru $\alpha X \beta \to \alpha \gamma \beta$, kde $|\gamma| \geq 1$ (Výjimka $S \to \varepsilon$, ale S pak není na pravé straně žádného pravidla)

2 - **Bezkontextové gramatiky** - pravidla tvaru $X \to \gamma$

3 - **Regulární gramatiky** - pravidla tvaru $X \to wY$ nebo $X \to w$

$$\alpha, \beta, \gamma \in (\Pi \cup \Sigma)^*, X \in \Pi, w \in \Sigma^*$$

- Jazyk je typu i, jestliže jej generuje nějaká gramatika typu i
- Označíme-li $\mathcal{L}_i$ třídu jazyků typu i, pak $\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$

Chomského hierarchie

Pojmenování jazyků jednotlivých typů:

- 0 - Rekurzivně spočetné
- 1 - Kontextové
- 2 - Bezkontextové
- 3 - Regulární

Chomského hierarchie

Pojmenování jazyků jednotlivých typů:

- 0 - Rekurzivně spočetné
- 1 - Kontextové
- 2 - Bezkontextové
- 3 - Regulární

Každé třídě jazyků odpovídá typ stroje nebo automatu, který dokáže rozpoznat všechny jazyky z dané třídy

- 0 - Turingův stroj (deterministický nebo nedeterministický)
- 1 - Lineárně omezený automat (nedeterministický)
- 2 - Zásobníkový automat (nedeterministický)
- 3 - Konečný automat (deterministický nebo nedeterministický)