

Neregulární jazyky

Ne všechny jazyky jsou regulární.

Existují jazyky, pro které neexistuje žádný konečný automat, který by je rozpoznával.

Příklady neregulárních jazyků:

- $L_1 = \{a^n b^n \mid n \geq 0\}$
- $L_2 = \{ww \mid w \in \{a, b\}^*\}$
- $L_3 = \{ww^R \mid w \in \{a, b\}^*\}$

Poznámka: Existence neregulárních jazyků vyplývá již z faktu, že automatů pracujících nad nějakou abecedou Σ je jen spočetně mnoho, zatímco jazyků nad abecedou Σ je nespočetně mnoho.

Jak dokázat o nějakém jazyce L , že není regulární?

Jazyk není regulární, jestliže neexistuje (tj. není možné sestrojít) konečný automat, který by ho rozpoznával.

Jak ale dokázat, že něco neexistuje?

Jak dokázat o nějakém jazyce L , že není regulární?

Jazyk není regulární, jestliže neexistuje (tj. není možné sestrojít) konečný automat, který by ho rozpoznával.

Jak ale dokázat, že něco neexistuje?

Odpověď: Sporem.

Např. předpokládat, že existuje nějaký automat A rozpoznávající jazyk L , a ukázat, že tento předpoklad vede k logickému sporu.

Ukážeme, že jazyk $L = \{a^n b^n \mid n \geq 0\}$ není regulární.

Důkaz sporem.

Předpokládejme, že existuje DKA $A = (Q, \Sigma, \delta, q_0, F)$ takový, že $L(A) = L$.

Neregulární jazyky

Ukážeme, že jazyk $L = \{a^n b^n \mid n \geq 0\}$ není regulární.

Důkaz sporem.

Předpokládejme, že existuje DKA $A = (Q, \Sigma, \delta, q_0, F)$ takový, že $L(A) = L$.

Řekněme, že $|Q| = n$.

Neregulární jazyky

Ukážeme, že jazyk $L = \{a^n b^n \mid n \geq 0\}$ není regulární.

Důkaz sporem.

Předpokládejme, že existuje DKA $A = (Q, \Sigma, \delta, q_0, F)$ takový, že $L(A) = L$.

Řekněme, že $|Q| = n$.

Vezměme si slovo $z = a^n b^n$.

Neregulární jazyky

Ukážeme, že jazyk $L = \{a^n b^n \mid n \geq 0\}$ není regulární.

Důkaz sporem.

Předpokládejme, že existuje DKA $A = (Q, \Sigma, \delta, q_0, F)$ takový, že $L(A) = L$.

Řekněme, že $|Q| = n$.

Vezměme si slovo $z = a^n b^n$.

Protože $z \in L$, musí existovat přijímající výpočet automatu A

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} \cdots \xrightarrow{a} q_{n-1} \xrightarrow{a} q_n \xrightarrow{b} q_{n+1} \xrightarrow{b} \cdots \xrightarrow{b} q_{2n-1} \xrightarrow{b} q_{2n}$$

kde q_0 je počáteční stav a $q_{2n} \in F$.

Vezměme si nyní prvních $n + 1$ stavů ve výpočtu

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} \cdots \xrightarrow{a} q_{n-1} \xrightarrow{a} q_n \xrightarrow{b} q_{n+1} \xrightarrow{b} \cdots \xrightarrow{b} q_{2n-1} \xrightarrow{b} q_{2n}$$

tj. posloupnost stavů q_0, q_1, \dots, q_n .

Je zřejmé, že všechny stavy v této posloupnosti nemohou být navzájem různé, protože $|Q| = n$ a tato posloupnost má $n + 1$ prvků.

To znamená, že existuje nějaký stav $q \in Q$, který se v této posloupnosti vyskytuje (alespoň) dvakrát.

Neregulární jazyky

Vezměme si nyní prvních $n + 1$ stavů ve výpočtu

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} \cdots \xrightarrow{a} q_{n-1} \xrightarrow{a} q_n \xrightarrow{b} q_{n+1} \xrightarrow{b} \cdots \xrightarrow{b} q_{2n-1} \xrightarrow{b} q_{2n}$$

tj. posloupnost stavů q_0, q_1, \dots, q_n .

Je zřejmé, že všechny stavy v této posloupnosti nemohou být navzájem různé, protože $|Q| = n$ a tato posloupnost má $n + 1$ prvků.

To znamená, že existuje nějaký stav $q \in Q$, který se v této posloupnosti vyskytuje (alespoň) dvakrát.

Jde o aplikaci tzv. **holubníkového principu (pigeonhole principle)**.

Holubníkový princip

Jestliže mám $n + 1$ holubů rozmístěných do n klecí, pak jsou alespoň v jedné kleci minimálně dva holubi.

Neregulární jazyky

Vezměme si nyní prvních $n + 1$ stavů ve výpočtu

$$q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2 \xrightarrow{a} \cdots \xrightarrow{a} q_{n-1} \xrightarrow{a} q_n \xrightarrow{b} q_{n+1} \xrightarrow{b} \cdots \xrightarrow{b} q_{2n-1} \xrightarrow{b} q_{2n}$$

tj. posloupnost stavů q_0, q_1, \dots, q_n .

Je zřejmé, že všechny stavy v této posloupnosti nemohou být navzájem různé, protože $|Q| = n$ a tato posloupnost má $n + 1$ prvků.

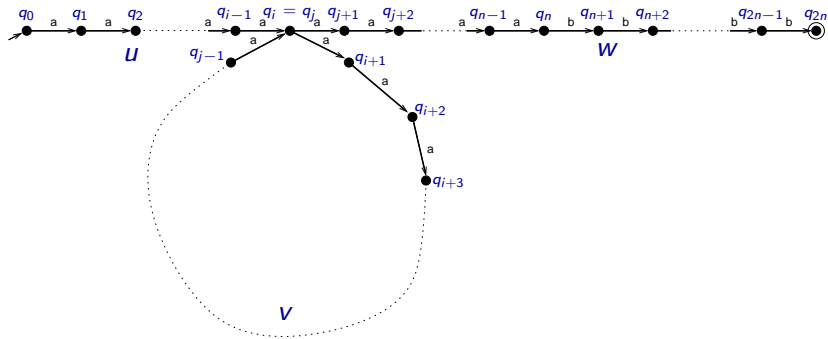
To znamená, že existuje nějaký stav $q \in Q$, který se v této posloupnosti vyskytuje (alespoň) dvakrát.

Tj. existují indexy i, j takové, že $0 \leq i < j \leq n$ a

$$q_i = q_j$$

což znamená, že automat A při čtení symbolů a ve slově $z = a^n b^n$ projde cyklem.

Neregulární jazyky



Slovo $z = a^n b^n$ můžeme rozdělit na tři části u, v, w takové, že $z = uvw$:

$$u = a^j$$

$$v = a^{j-i}$$

$$w = a^{n-j} b^n$$

Neregulární jazyky

Pro slova $u = a^i$, $v = a^{j-i}$ a $w = a^{n-j} b^n$ platí

$$q_0 \xrightarrow{u} q_i \qquad q_i \xrightarrow{v} q_j \qquad q_j \xrightarrow{w} q_{2n}$$

Označme r délku slova v , tj. $r = j - i$ (zjevně $r > 0$, protože $i < j$).

Protože $q_i = q_j$, tak automat přijme slovo $uw = a^{n-r} b^n$, které nepatří do jazyka L :

$$q_0 \xrightarrow{u} q_i \xrightarrow{w} q_{2n}$$

Rovněž slovo $uvvw = a^{n+r} b^n$, které také nepatří do L , bude přijato:

$$q_0 \xrightarrow{u} q_i \xrightarrow{v} q_i \xrightarrow{v} q_i \xrightarrow{w} q_{2n}$$

Neregulární jazyky

Podobně můžeme zdůvodnit, že každé slovo tvaru $uvvvv \cdots vvw$, tj. tvaru $uv^k w$ pro nějaké $k \geq 0$, bude automatem A přijato:

$$q_0 \xrightarrow{u} q_i \xrightarrow{v} q_j \xrightarrow{v} q_i \xrightarrow{v} \cdots \xrightarrow{v} q_i \xrightarrow{v} q_i \xrightarrow{w} q_{2n}$$

Slovo tvaru $uv^k w$ vypadá následovně: $a^{n-r+rk} b^n$.

Protože $r > 0$, tak následující rovnost platí jen pro $k = 1$:

$$n - r + rk = n$$

Pokud je tedy $k \neq 1$, tak slovo $uv^k w$ nepatří do jazyka L .

Automat A však každé takové slovo přijme, což je spor s předpokladem, že $L(A) = \{a^n b^n \mid n \geq 0\}$.

Bezkontextové gramatiky

Příklad:

$\langle \text{STMT} \rangle$	\rightarrow	$\langle \text{IF-STMT} \rangle \mid \langle \text{WHILE-STMT} \rangle \mid \langle \text{BLOCK-STMT} \rangle \mid \langle \text{ASSG-STMT} \rangle$
$\langle \text{IF-STMT} \rangle$	\rightarrow	if $\langle \text{BOOL-EXPR} \rangle$ then $\langle \text{STMT} \rangle$ else $\langle \text{STMT} \rangle$
$\langle \text{WHILE-STMT} \rangle$	\rightarrow	while $\langle \text{BOOL-EXPR} \rangle$ do $\langle \text{STMT} \rangle$
$\langle \text{BLOCK-STMT} \rangle$	\rightarrow	begin $\langle \text{STMT-LIST} \rangle$ 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	$\langle \text{VAR} \rangle \mid \langle \text{CONST} \rangle \mid$ $(\langle \text{ARITH-EXPR} \rangle \langle \text{ARITH-OP} \rangle \langle \text{ARITH-EXPR} \rangle)$
$\langle \text{ARITH-OP} \rangle$	\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	$a \mid b \mid c \mid \dots \mid x \mid y \mid z$

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

Pravidla popisují, jaké řetězce může daný neterminál reprezentovat.

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

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

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

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

$\langle \text{STMT} \rangle$

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

$\langle \text{STMT} \rangle$

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

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 $x \leq y$ **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 $x \leq y$ **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 $x \leq y$ **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 $x \leq y$ **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 $x \leq y$ **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 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** $\langle \text{STMT} \rangle$

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** $\langle \text{STMT} \rangle$

while $x \leq y$ **do** $\langle \text{BLOCK-STMT} \rangle$

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

⟨STMT⟩

⟨WHILE-STMT⟩

while ⟨BOOL-EXPR⟩ **do** ⟨STMT⟩

while ⟨ARITH-EXPR⟩⟨COMPARE-OP⟩⟨ARITH-EXPR⟩ **do** ⟨STMT⟩

while ⟨VAR⟩⟨COMPARE-OP⟩⟨ARITH-EXPR⟩ **do** ⟨STMT⟩

while ⟨VAR⟩ \leq ⟨ARITH-EXPR⟩ **do** ⟨STMT⟩

while ⟨VAR⟩ \leq ⟨VAR⟩ **do** ⟨STMT⟩

while $x \leq$ ⟨VAR⟩ **do** ⟨STMT⟩

while $x \leq y$ **do** ⟨STMT⟩

while $x \leq y$ **do** ⟨BLOCK-STMT⟩

while $x \leq y$ **do begin** ⟨STMT-LIST⟩ **end**

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** $\langle \text{STMT} \rangle$

while $x \leq y$ **do** $\langle \text{BLOCK-STMT} \rangle$

while $x \leq y$ **do begin** $\langle \text{STMT-LIST} \rangle$ **end**

...

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**

Poznámky:

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

$$A \rightarrow \alpha$$

A – levá strana pravidla

α – pravá strana pravidla

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

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

$$A \rightarrow aBBb$$

$$A \rightarrow AaA$$

$$B \rightarrow \varepsilon$$

$$B \rightarrow bCA$$

$$C \rightarrow AB$$

$$C \rightarrow a$$

$$C \rightarrow b$$

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

$$A \rightarrow \alpha_1$$

$$A \rightarrow \alpha_2$$

$$A \rightarrow \alpha_3$$

můžeme je stručněji zapsat jako

$$A \rightarrow \alpha_1 \mid \alpha_2 \mid \alpha_3$$

Například pravidla dříve uvedené gramatiky můžeme zapsat jako

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

Gramatiky slouží ke generování slov.

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

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

$$\underline{A} \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

$$\underline{A} \rightarrow \underline{aBBb} \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

$$A \rightarrow aBBb \mid AaA$$

$$\underline{B} \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

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

Gramatiky slouží ke generování slov.

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

$$A \rightarrow aBBb \mid AaA$$

$$\underline{B} \rightarrow \varepsilon \mid \underline{bCA}$$

$$C \rightarrow AB \mid a \mid b$$

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

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

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

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

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

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

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

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

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

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

$$A \rightarrow aBBb \mid AaA$$

$$\underline{B} \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

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

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

$$A \rightarrow aBBb \mid AaA$$

$$\underline{B} \rightarrow \underline{\varepsilon} \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

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

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

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

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

Gramatiky slouží ke generování slov.

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

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$\underline{C} \rightarrow AB \mid a \mid b$$

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

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$\underline{C} \rightarrow AB \mid a \mid \underline{b}$$

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

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

$$A \rightarrow aBBb \mid AaA$$

$$\underline{B} \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

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

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

$$A \rightarrow aBBb \mid AaA$$

$$\underline{B} \rightarrow \underline{\varepsilon} \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

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

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

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

$$A \rightarrow aBBb \mid AaA$$

$$\underline{B} \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

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

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

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

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

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

$$A \rightarrow aBBb \mid AaA$$

$$B \rightarrow \varepsilon \mid bCA$$

$$C \rightarrow AB \mid a \mid b$$

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

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

Na řetězcích z $(\Pi \cup \Sigma)^*$ 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 α je možné jedním krokem odvodit α' , a to tak, že výskyt nějakého neterminálu A v α nahradíme pravou stranou nějakého pravidla $A \rightarrow \gamma$, kde se A vyskytuje na levé straně.

Na řetězcích z $(\Pi \cup \Sigma)^*$ 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

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

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

Derivace délky n je posloupnost $\beta_0, \beta_1, \beta_2, \dots, \beta_n$, kde $\beta_i \in (\Pi \cup \Sigma)^*$ a kde $\beta_{i-1} \Rightarrow \beta_i$ pro všechna $1 \leq i \leq n$, což můžeme stručněji zapsat

$$\beta_0 \Rightarrow \beta_1 \Rightarrow \beta_2 \Rightarrow \dots \Rightarrow \beta_{n-1} \Rightarrow \beta_n$$

Skutečnost, že pro dané $\alpha, \alpha' \in (\Pi \cup \Sigma)^*$ a $n \in \mathbb{N}$ existuje nějaká derivace $\beta_0 \Rightarrow \beta_1 \Rightarrow \beta_2 \Rightarrow \dots \Rightarrow \beta_{n-1} \Rightarrow \beta_n$, kde $\alpha = \beta_0$ a $\alpha' = \beta_n$, zapisujeme

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

Skutečnost, že $\alpha \Rightarrow^n \alpha'$ pro nějaké $n \geq 0$, zapisujeme

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

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

Větné formy jsou ty $\alpha \in (\Pi \cup \Sigma)^*$, pro které platí

$$S \Rightarrow^* \alpha$$

kde S je počáteční neterminál.

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.

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

Příklad: Chceme vytvořit gramatiku generující jazyk

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

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

$$S \rightarrow aSb \mid \varepsilon$$

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

$$S \rightarrow aSb \mid \varepsilon$$

$$S \Rightarrow \varepsilon$$

$$S \Rightarrow aSb \Rightarrow ab$$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaaaSbbbb \Rightarrow aaaabbbb$$

...

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.

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:

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

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:

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

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

Příklad: Chceme vytvořit gramatiku generující jazyk L tvořený všemi dobře uzávorkovanými sekvencemi symbolů '(' a ')'.
Například $((()())()) \in L$, ale $)() \notin L$.

Příklad: Chceme vytvořit gramatiku generující jazyk L tvořený všemi dobře uzávorkovanými sekvencemi symbolů '(' a ')'.
Například $((()())()) \in L$, ale $)() \notin L$.

Řešení:

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

Příklad: Chceme vytvořit gramatiku generující jazyk L tvořený všemi dobře uzávorkovanými sekvencemi symbolů '(' a ')'.
Například $((()())()) \in L$, ale $)() \notin L$.

Řešení:

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

$$\begin{aligned} S &\Rightarrow SS \Rightarrow (S)S \Rightarrow (S)(S) \Rightarrow (SS)(S) \Rightarrow ((S)S)(S) \Rightarrow \\ &(()S)(S) \Rightarrow (()S)()S \Rightarrow (()())(S) \Rightarrow (()())((S)) \Rightarrow \\ &(()())(()) \end{aligned}$$

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

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:

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

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:

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

$$\begin{aligned} E &\Rightarrow E + E \Rightarrow E * E + E \Rightarrow (E) * E + E \Rightarrow (E + E) * E + E \Rightarrow \\ &(a + E) * E + E \Rightarrow (a + a) * E + E \Rightarrow (a + a) * a + E \Rightarrow (a + a) * a + (E) \Rightarrow \\ &(a + a) * a + (E * E) \Rightarrow (a + a) * a + (a * E) \Rightarrow (a + a) * a + (a * a) \end{aligned}$$

$A \rightarrow aBBb \mid AaA$

$B \rightarrow \varepsilon \mid bCA$

$C \rightarrow AB \mid a \mid b$

A

$A \rightarrow aBBb \mid AaA$

$B \rightarrow \varepsilon \mid bCA$

$C \rightarrow AB \mid a \mid b$

A

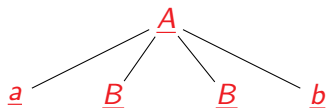
A

A $\rightarrow aBBb \mid AaA$

$B \rightarrow \varepsilon \mid bCA$

$C \rightarrow AB \mid a \mid b$

A

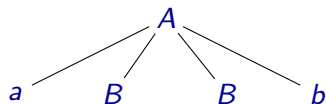


A \rightarrow aBBb | AaA

B \rightarrow ε | bCA

C \rightarrow AB | a | b

A \Rightarrow aBBb

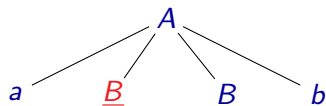


$A \rightarrow aBBb \mid AaA$

$B \rightarrow \varepsilon \mid bCA$

$C \rightarrow AB \mid a \mid b$

$A \Rightarrow aBBb$



$A \rightarrow aBBb \mid AaA$

$\underline{B} \rightarrow \varepsilon \mid bCA$

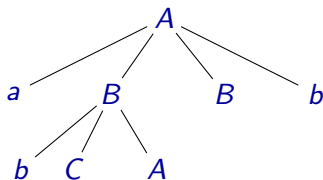
$C \rightarrow AB \mid a \mid b$

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

$A \rightarrow aBBb \mid AaA$

$B \rightarrow \varepsilon \mid bCA$

$C \rightarrow AB \mid a \mid b$

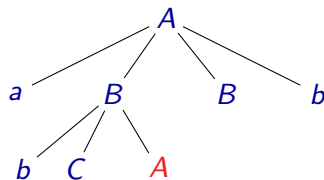


$A \Rightarrow aBBb \Rightarrow abCABb$

$\underline{A} \rightarrow aBBb \mid AaA$

$B \rightarrow \varepsilon \mid bCA$

$C \rightarrow AB \mid a \mid b$

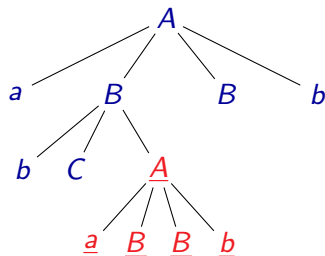


$A \Rightarrow aBBb \Rightarrow abC\underline{A}Bb$

$\underline{A} \rightarrow \underline{aBBb} \mid AaA$

$B \rightarrow \varepsilon \mid bCA$

$C \rightarrow AB \mid a \mid b$

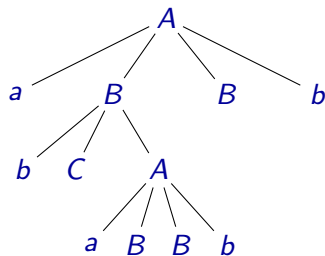


$A \Rightarrow aBBb \Rightarrow abC\underline{A}Bb \Rightarrow abC\underline{aBBb}Bb$

$A \rightarrow aBBb \mid AaA$

$B \rightarrow \varepsilon \mid bCA$

$C \rightarrow AB \mid a \mid b$



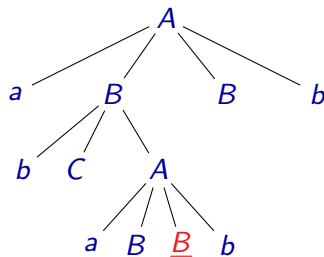
$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb$

Derivační strom

$A \rightarrow aBBb \mid AaA$

$\underline{B} \rightarrow \varepsilon \mid bCA$

$C \rightarrow AB \mid a \mid b$

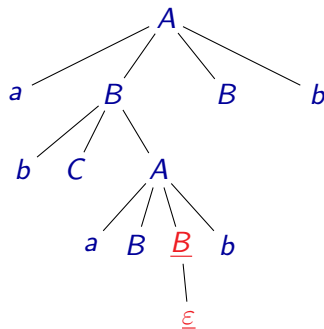


$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCa\underline{B}bBb$

$A \rightarrow aBBb \mid AaA$

$\underline{B} \rightarrow \underline{\epsilon} \mid bCA$

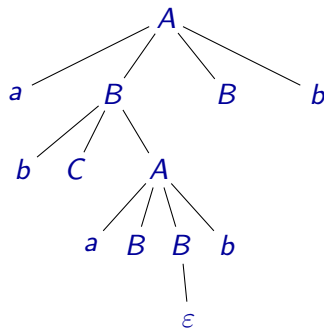
$C \rightarrow AB \mid a \mid b$



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

Derivační strom

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



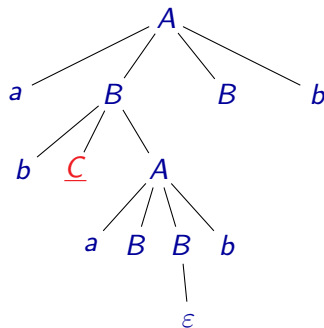
$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb$

Derivační strom

$A \rightarrow aBBb \mid AaA$

$B \rightarrow \varepsilon \mid bCA$

$\underline{C} \rightarrow AB \mid a \mid b$



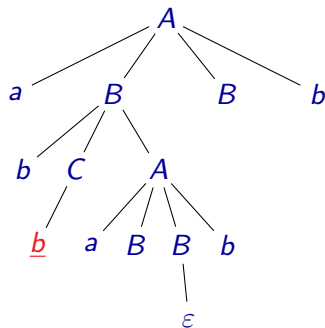
$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow ab\underline{C}aBbBb$

Derivační strom

$A \rightarrow aBBb \mid AaA$

$B \rightarrow \varepsilon \mid bCA$

$\underline{C} \rightarrow AB \mid a \mid \underline{b}$



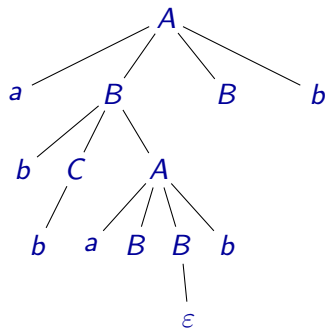
$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow ab\underline{C}aBbBb \Rightarrow ab\underline{b}aBbBb$

Derivační strom

$A \rightarrow aBBb \mid AaA$

$B \rightarrow \varepsilon \mid bCA$

$C \rightarrow AB \mid a \mid b$



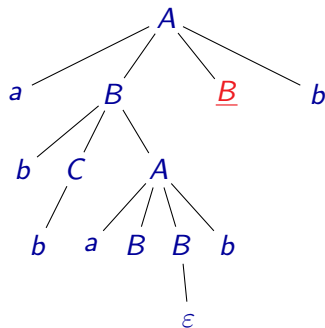
$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb$

Derivační strom

$A \rightarrow aBBb \mid AaA$

$\underline{B} \rightarrow \varepsilon \mid bCA$

$C \rightarrow AB \mid a \mid b$



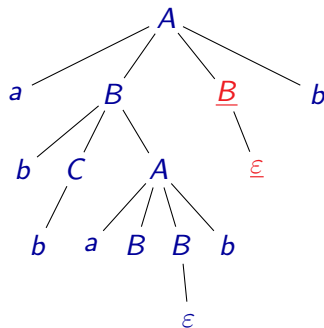
$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBb\underline{B}b$

Derivační strom

$A \rightarrow aBBb \mid AaA$

$\underline{B} \rightarrow \underline{\epsilon} \mid bCA$

$C \rightarrow AB \mid a \mid b$



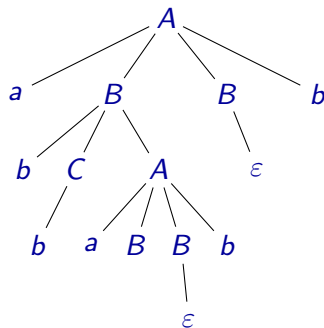
$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBb\underline{B}b \Rightarrow abbaBbb$

Derivační strom

$A \rightarrow aBBb \mid AaA$

$B \rightarrow \varepsilon \mid bCA$

$C \rightarrow AB \mid a \mid b$



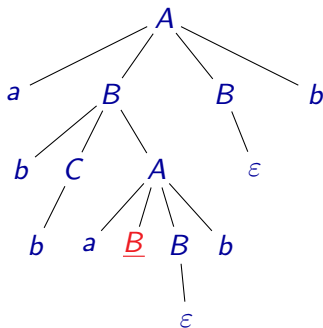
$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abbaBbb$

Derivační strom

$A \rightarrow aBBb \mid AaA$

$\underline{B} \rightarrow \varepsilon \mid bCA$

$C \rightarrow AB \mid a \mid b$



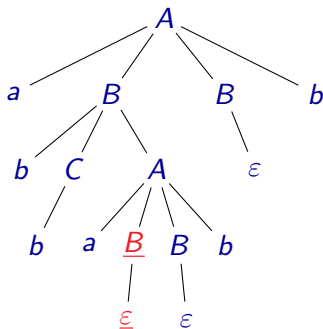
$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abba\underline{B}bb$

Derivační strom

$A \rightarrow aBBb \mid AaA$

$\underline{B} \rightarrow \underline{\epsilon} \mid bCA$

$C \rightarrow AB \mid a \mid b$



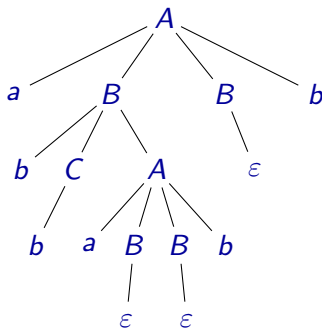
$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abba\underline{B}bb \Rightarrow abbabb$

Derivační strom

$A \rightarrow aBBb \mid AaA$

$B \rightarrow \varepsilon \mid bCA$

$C \rightarrow AB \mid a \mid b$



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

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 $A \rightarrow \alpha$.

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

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:

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

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

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^* .

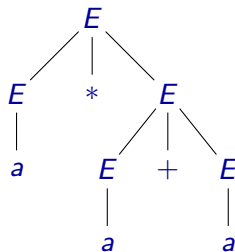
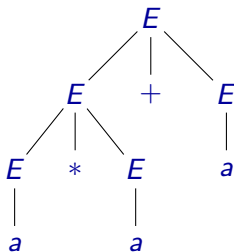
Nejednoznačné gramatiky

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

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

můžeme nahradit ekvivalentní gramatikou

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

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

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 $L(G) = L(G_1)L(G_2)$:

$$G = (\Pi_1 \cup \Pi_2 \cup \{S\}, \Sigma, S, P_1 \cup P_2 \cup \{S \rightarrow S_1S_2\})$$

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

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

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

$$G = (\Pi_1 \cup \{S\}, \Sigma, S, P_1 \cup \{S \rightarrow \varepsilon, S \rightarrow S_1S\})$$