Bezkontextové gramatiky

Bezkontextové gramatiky

Příklad: Chtěli bychom popsat jazyk aritmetických výrazů obsahující výrazy jako například:

`175` `(9+15)` `(((10-4)*((1+34)+2))/(3+(-37)))`

Pro jednoduchost předpokládejme:

- Výrazy jsou plně uzávorkované.
- Jediné aritmetické operace jsou "`+`", "`-`", "`*`", "`/`" a unární "`-`".
- Hodnoty operandů jsou přirozená čísla zapsaná v desítkové soustavě — zápis čísla je neprázdná posloupnost číslic.

Abeceda jazyka: $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, *, /, (,)\}$

Bezkontextové gramatiky

Příklad (pokr.): Popis pomocí induktivní definice:

- **Číslice** je libovolný ze znaků 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- **Číslo** je neprázdná posloupnost číslic, tj.:
  - Pokud je α číslice, tak α je číslo.
  - Pokud α je číslice a β je číslo, tak i $\alpha\beta$ je číslo.
- **Výraz** je libovolná posloupnost symbolů vytvořená podle následujících pravidel:
  - Pokud je α číslo, tak α je výraz.
  - Pokud α je výraz, tak i $(-\alpha)$ je výraz.
  - Pokud α a β jsou výrazy, tak i $(\alpha+\beta)$ je výraz.
  - Pokud α a β jsou výrazy, tak i $(\alpha-\beta)$ je výraz.
  - Pokud α a β jsou výrazy, tak i $(\alpha*\beta)$ je výraz.
  - Pokud α a β jsou výrazy, tak i (α/β) je výraz.

Bezkontextové gramatiky

Příklad (pokr.): Způsob zápisu téže informace jako v předchozí induktivní definici pomocí **bezkontextové gramatiky**:

Zavedeme následující pomocné symboly — těmto symbolům se říká **neterminály**:

- D — zastupuje libovolnou číslici
- C — zastupuje libovolné číslo
- E — zastupuje libovolný výraz

$$D \to 0 \quad D \to 1 \quad D \to 2 \quad D \to 3 \quad D \to 4$$

$$D \to 5 \quad D \to 6 \quad D \to 7 \quad D \to 8 \quad D \to 9$$

$$C \to D \quad C \to DC$$

$$E \to C \quad E \to (-E) \quad E \to (E{+}E) \quad E \to (E{-}E) \quad E \to (E{*}E) \quad E \to (E/E)$$

Bezkontextové gramatiky

Příklad (pokr.): Stručnější způsob zápisu:

$$D \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$
$$C \to D \mid DC$$
$$E \to C \mid (-E) \mid (E+E) \mid (E-E) \mid (E*E) \mid (E/E)$$

Bezkontextové gramatiky

Příklad: Jazyk, kde slova jsou (případně i prázdné) posloupnosti výrazů popsaných v předchozím příkladě, kde jednotlivé výrazy jsou odděleny čárkami (abecedu je třeba rozšířit o symbol ","):

$$
\begin{aligned}
S &\to T \mid \varepsilon \\
T &\to E \mid E\,,T \\
D &\to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
C &\to D \mid DC \\
E &\to C \mid (-E) \mid (E{+}E) \mid (E{-}E) \mid (E{*}E) \mid (E/E)
\end{aligned}
$$

Bezkontextové gramatiky

Příklad: Příkazy nějakého programovacího jazyka (fragment gramatiky):

$$
\begin{aligned}
S &\to E\texttt{;} \mid T \mid \texttt{if (}E\texttt{)}\ S \mid \texttt{if (}E\texttt{)}\ S\ \texttt{else}\ S \\
&\quad \mid \texttt{while (}E\texttt{)}\ S \mid \texttt{do}\ S\ \texttt{while (}E\texttt{);} \mid \texttt{for (}F\texttt{;}\,F\texttt{;}\,F\texttt{)}\ S \\
&\quad \mid \texttt{return}\ F\texttt{;} \\
T &\to \{\ U\ \} \\
U &\to \varepsilon \mid SU \\
F &\to \varepsilon \mid E \\
E &\to \quad \ldots \\
&\qquad \ldots
\end{aligned}
$$

Poznámka:

- S — příkaz
- T — blok příkazů
- U — sekvence příkazů
- E — výraz
- F — výraz, který je možno vynechat

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

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

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

$\underline{A}$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 $(A \to \gamma) \in P$.

Příklad: Jestliže $(B \to bCA) \in P$, 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 $A \to \gamma$, kde se A vyskytuje na levé straně.

Bezkontextové gramatiky

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 $(A \to \gamma) \in P$.

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 α je možné jedním krokem odvodit α', a to tak, že výskyt nějakého neterminálu A v α nahradíme pravou stranou nějakého pravidla $A \to \gamma$, kde se A vyskytuje na levé straně.

Bezkontextové gramatiky

Derivace délky n je posloupnost $\beta_0, \beta_1, \beta_2, \cdots, \beta_n$, 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

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

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

Bezkontextové gramatiky

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

$$S \Rightarrow^* \alpha$$

kde S je počáteční neterminál.

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:

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

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

Derivační strom

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

Derivační strom

A

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

A

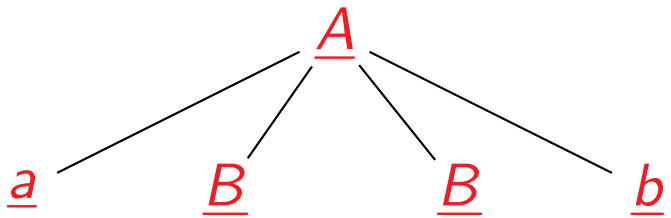
Derivační strom

$\underline{A}$

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

$\underline{A}$

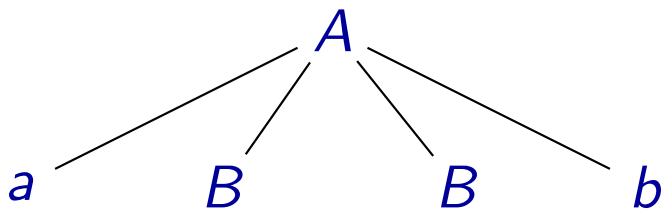
Derivační strom



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

$A \Rightarrow aBBb$

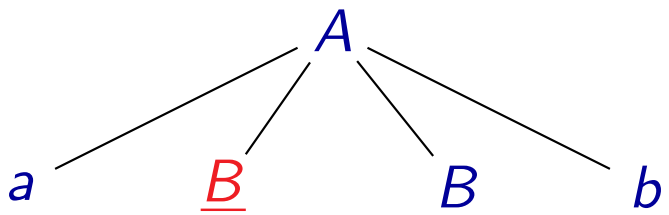
Derivační strom



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

$A \Rightarrow aBBb$

Derivační strom

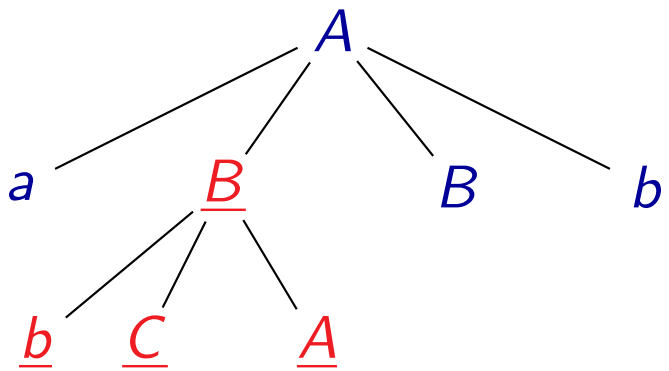


$A \to aBBb \mid AaA$
$\underline{B} \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$

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

Derivační strom

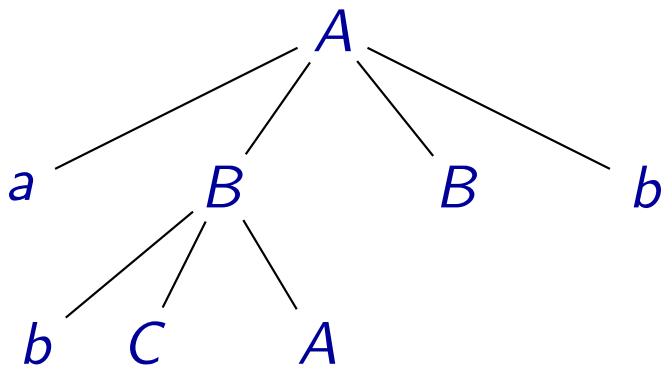
$A \to aBBb \mid AaA$
$\underline{B} \to \varepsilon \mid \underline{bCA}$
$C \to AB \mid a \mid b$



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

Derivační strom

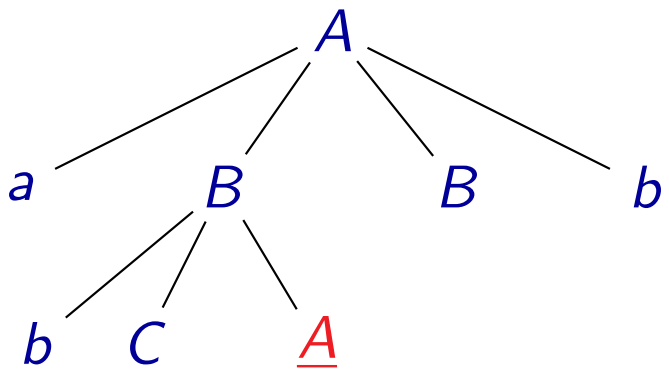
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb$

Derivační strom

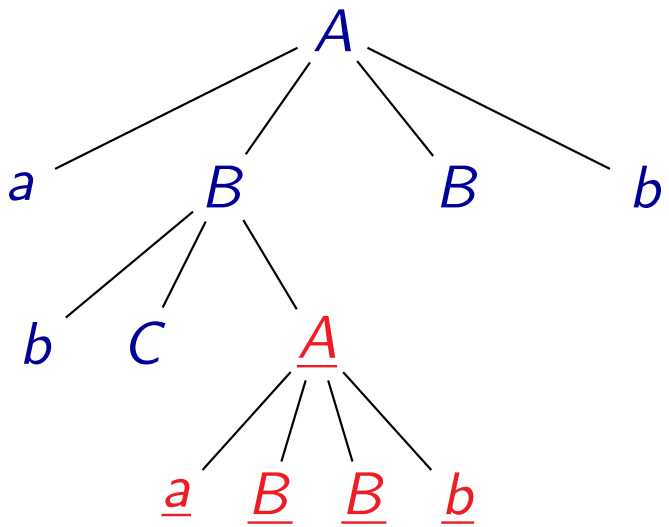
$\underline{A} \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



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

Derivační strom

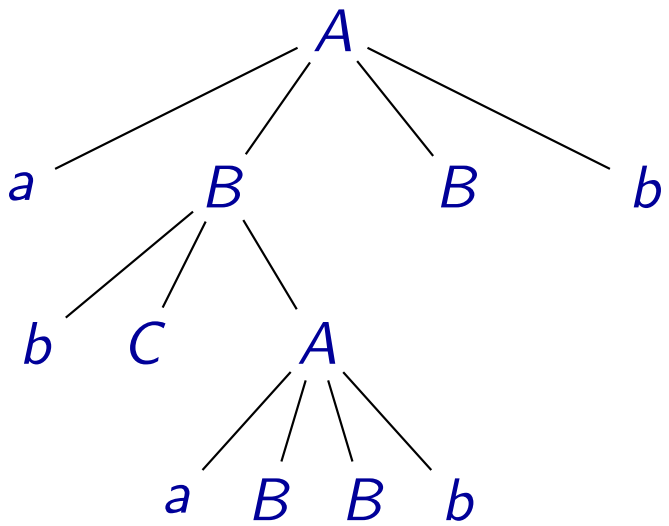
$\underline{A} \to \underline{aBBb} \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



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

Derivační strom

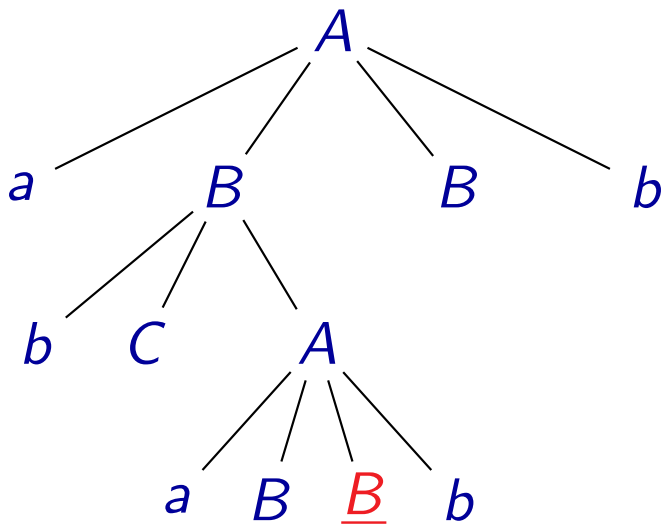
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



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

Derivační strom

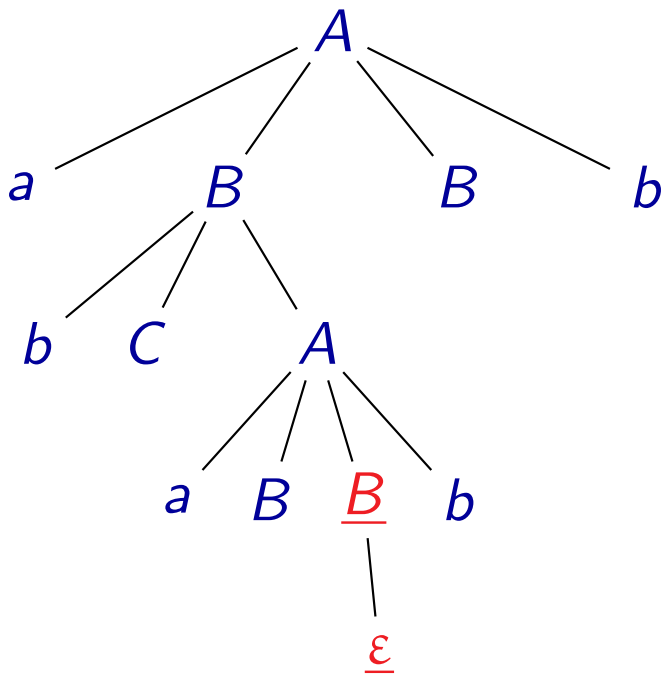
$A \to aBBb \mid AaA$
$\underline{B} \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



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

Derivační strom

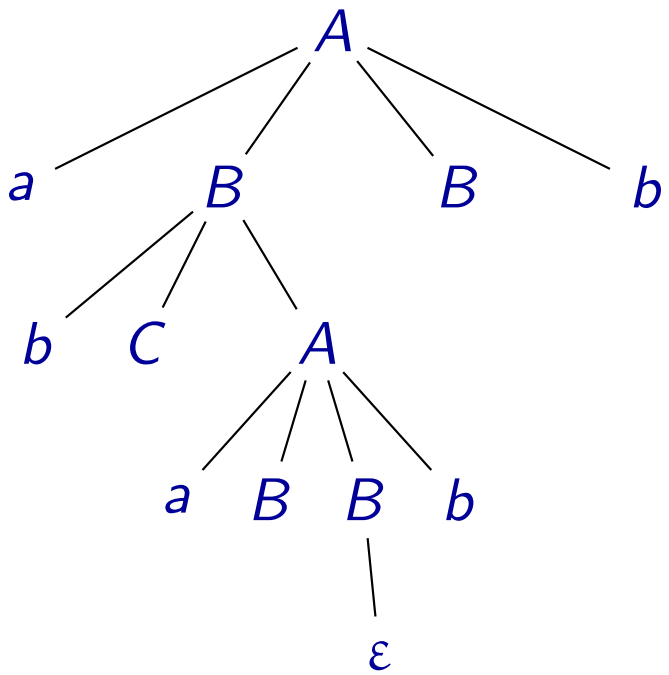
$A \to aBBb \mid AaA$
$\underline{B} \to \underline{\varepsilon} \mid bCA$
$C \to AB \mid a \mid b$



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

Derivační strom

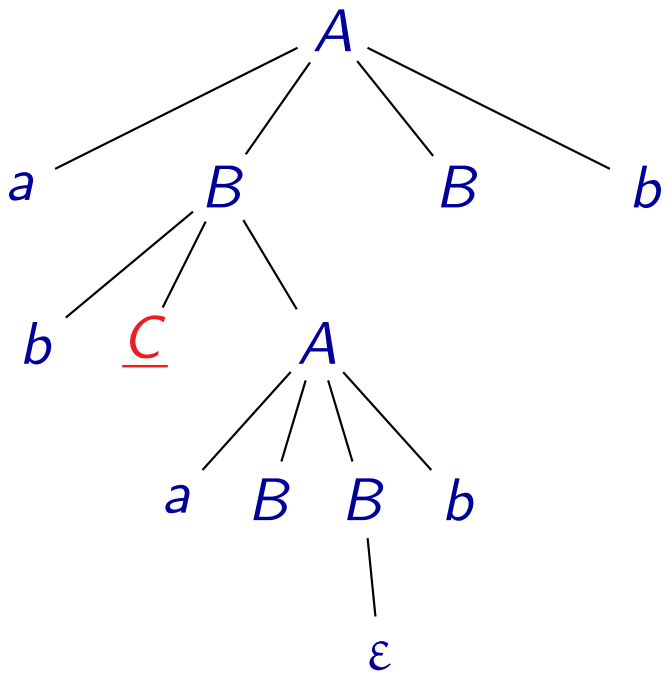
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



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

Derivační strom

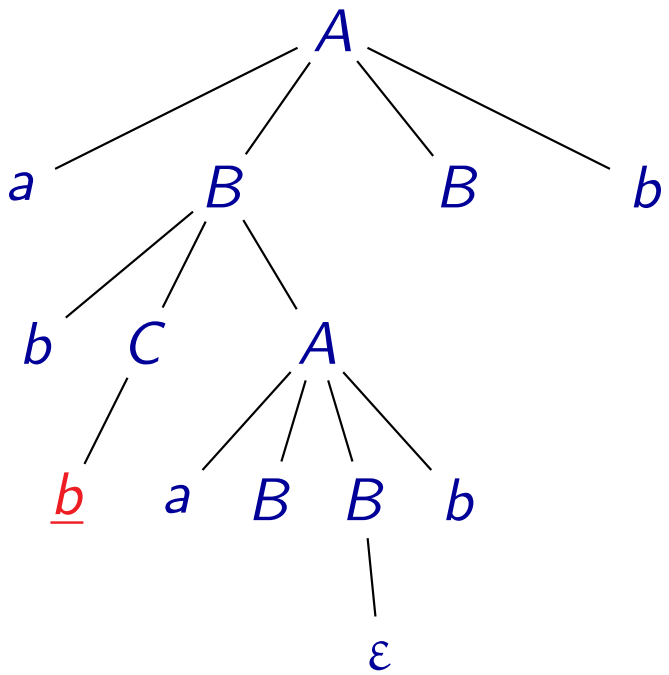
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$\underline{C} \to AB \mid a \mid b$



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

Derivační strom

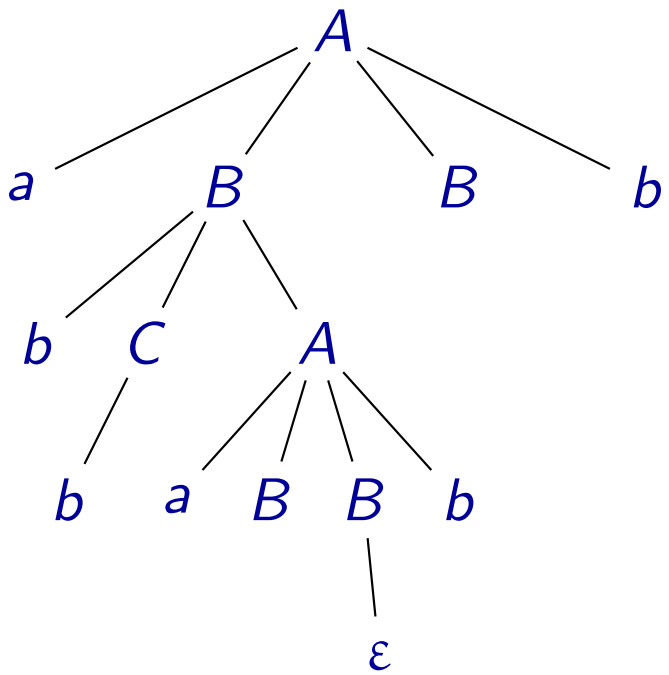
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$\underline{C} \to AB \mid a \mid \underline{b}$



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

Derivační strom

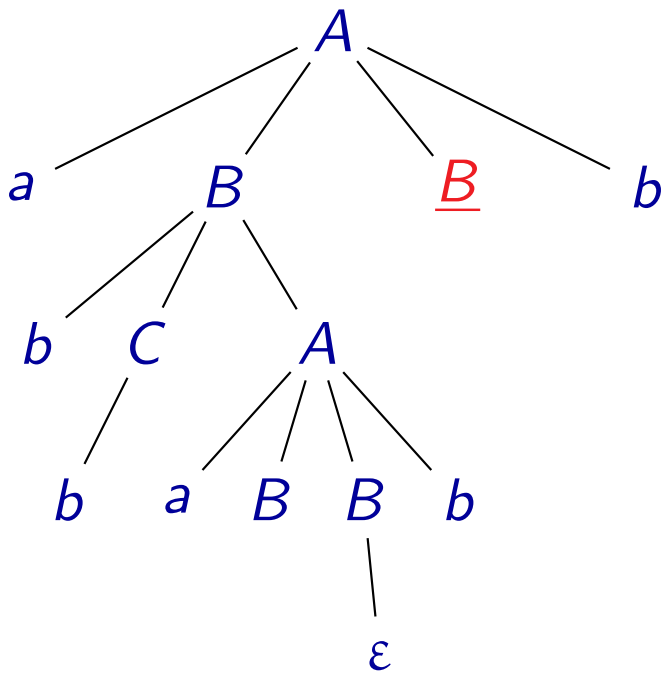
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



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

Derivační strom

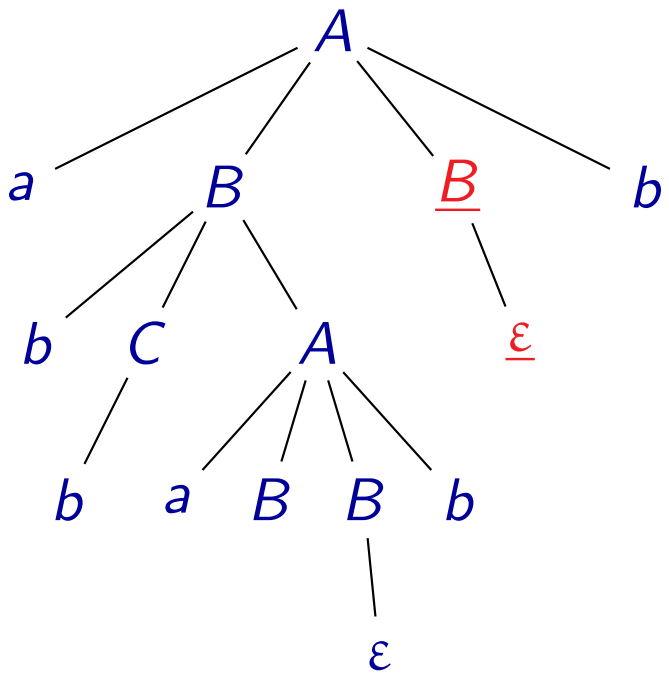
$A \to aBBb \mid AaA$
$\underline{B} \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



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

Derivační strom

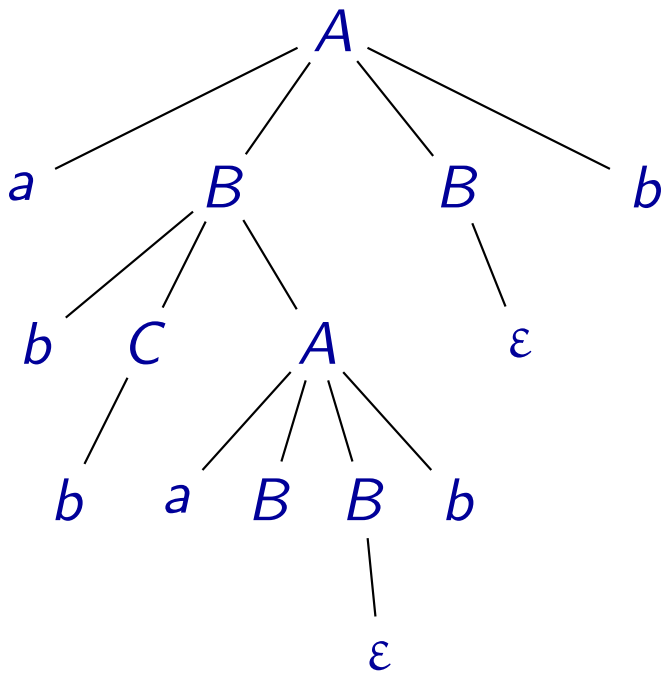
$A \to aBBb \mid AaA$
$\underline{B} \to \underline{\varepsilon} \mid bCA$
$C \to AB \mid a \mid b$



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

Derivační strom

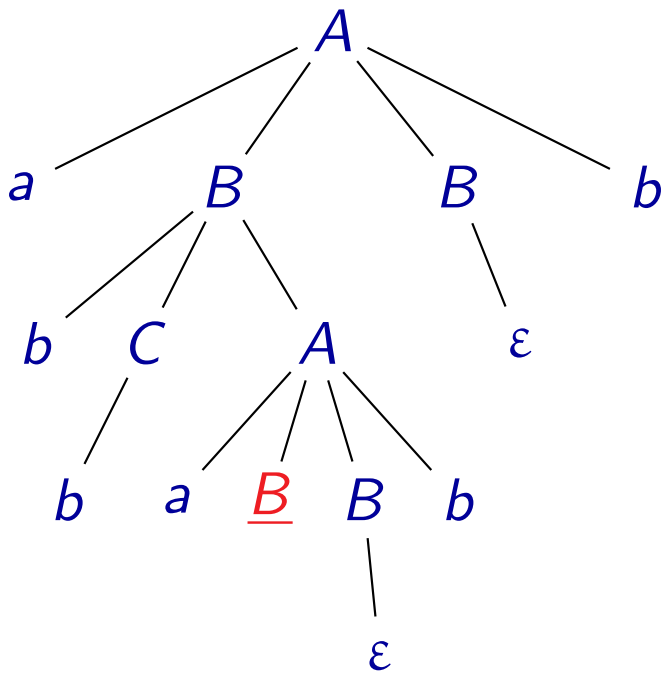
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



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

Derivační strom

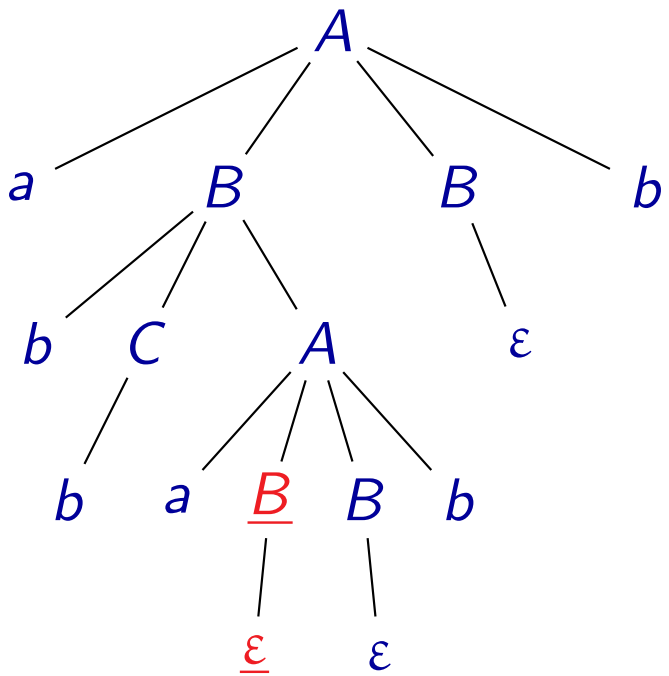
$A \to aBBb \mid AaA$
$\underline{B} \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



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

Derivační strom

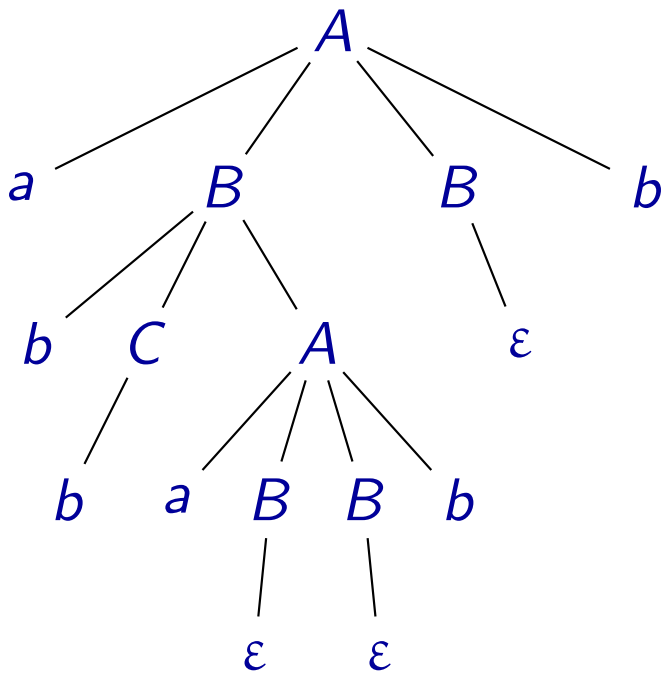
$A \to aBBb \mid AaA$
$\underline{B} \to \underline{\varepsilon} \mid bCA$
$C \to AB \mid a \mid b$



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

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.

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^*.

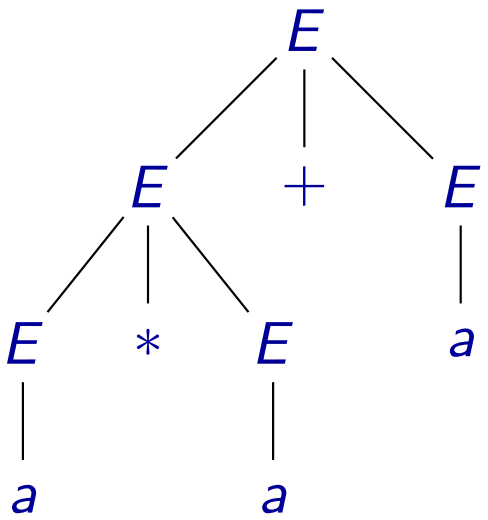
Nejednoznačné gramatiky

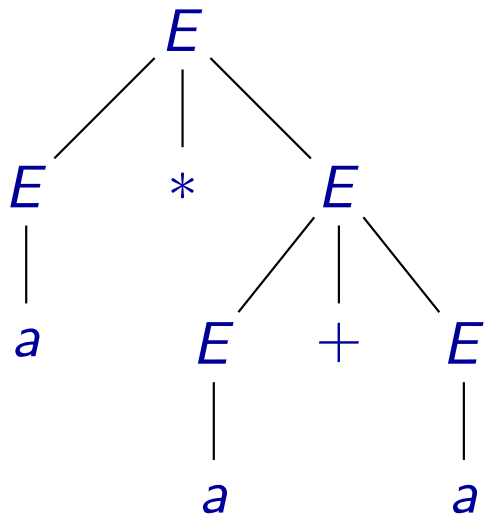
Gramatika G je **nejednoznačná**, jestliže existuje nějaké slovo $w \in L(G)$, kterému přísluší dva různé derivační stromy, resp. dvě různé levé či dvě různé pravé derivace.

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**.

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 $L(G) = L(G_1)^*$:

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

Lexikální a syntaktická analýza — příklad

Příklad: Chtěli bychom rozpoznávat jazyk aritmetických výrazů obsahující výrazy jako například:

```
34          x+1          -x * 2 + 128 * (y - z / 3)
```

- Ve výrazech se mohou vyskytovat číselné konstanty — posloupnosti číslic `0`, `1`, . . . , `9`.
- Ve výrazech se mohou vyskytovat jména proměnných — posloupnosti tvořené písmeny, číslicemi a znakem "`_`", které nezačínají číslicí.
- Výrazy mohou obsahovat základní aritmetické operace — "`+`", "`-`", "`*`", "`/`" a unární "`-`".
- Je možné používat závorky — "`(`" a "`)`" a běžnou prioritu aritmetických operací.

Lexikální a syntaktická analýza — příklad

- **Vstup:** posloupnost znaků (např. řetězec, textový soubor, apod.)
- **Výstup:** abstraktní syntaktický strom reprezentující strukturu daného výrazu nebo informace o syntaktické chybě ve výrazu

Lexikální a syntaktická analýza — příklad

Vytváření **abstraktního syntaktického stromu**:

- Výčtový typ reprezentující binární aritmetické operace:

  enum *Bin_op* { **Add**, **Sub**, **Mul**, **Div** }

- Výčtový typ reprezentující unární aritmetické operace:

  enum *Un_op* { **Un_minus** }

- Funkce pro vytváření různých typů vrcholů abstraktního syntaktického stromu:
  - MK-VAR(*ident*) — vytvoření listu reprezentujícího proměnnou
  - MK-NUM(*num*) — vytvoření listu reprezentujícího číselnou konstantu
  - MK-UNARY(*op*, *e*) — vytvoření vrcholu s jedním potomkem *e*, na který je aplikována unární operace *op* (typu *Un_op*)
  - MK-BINARY(*op*, *e1*, *e2*) — vytvoření vrcholu se dvěma potomky *e1* a *e2*, na které je aplikována binární operace *op* (typu *Bin_op*)

Lexikální analýza

Výčtový typ *Token_kind* reprezentující různé druhy **tokenů**:

- **T_EOF** — konec vstupu
- **T_Ident** — identifikátor
- **T_Number** — číselná konstanta
- **T_LParen** — "("
- **T_RParen** — ")"
- **T_Plus** — "+"
- **T_Minus** — "-"
- **T_Star** — "*"
- **T_Slash** — "/"

Lexikální analýza

Proměnná c : naposledy načtený znak (resp. speciální hodnota $\langle eof \rangle$ reprezentující, že byl dosažen konec vstupu):

- na začátku se do proměnné c načte první znak ze vstupu
- funkce NEXT-CHAR() vrací následující znak ze vstupu

Některé pomocné funkce:

- ERROR() — vypíše informaci o syntaktické chybě a ukončí načítání výrazu
- **is-ident-start-char**(c) — testuje, zda c je znak, kterým může začínat identifikátor
- **is-ident-normal-char**(c) — testuje, zda c je znak, který se může vyskytovat v identifikátoru (jinde než na začátku)
- **is-digit**(c) — testuje, zda je c číslice

Lexikální analýza

Některé další pomocné funkce:

- CREATE-IDENT(s) — vytváří identifikátor z daného řetězce s
- CREATE-NUMBER(s) — vytváří číslo z daného řetězce s

Pomocné proměnné:

- *last-ident* — naposledy načtený identifikátor
- *last-num* — naposledy načtená číselná konstanta

Funkce NEXT-TOKEN() — hlavní část lexikálního analyzátoru, vrací následující token ze vstupu

```
1  NEXT-TOKEN():
2  begin
3      while c ∈ {" ", "\t"} do
4          c := NEXT-CHAR();
5      end
6      if c == ⟨eof⟩ then
7          return T_EOF
8      else
9          switch c do
10             case "(":  c := NEXT-CHAR(); return T_LParen
11             case ")":  c := NEXT-CHAR(); return T_RParen
12             case "+":  c := NEXT-CHAR(); return T_Plus
13             case "-":  c := NEXT-CHAR(); return T_Minus
14             case "*":  c := NEXT-CHAR(); return T_Star
15             case "/":  c := NEXT-CHAR(); return T_Slash
16             otherwise
17                 if is-ident-start-char(c) then
18                     return SCAN-IDENT()
19                 else if is-digit(c) then
20                     return SCAN-NUMBER()
21                 else  ERROR()
22             end
23         endsw
24     end
25 end
```

Lexikální analýza

```
1  SCAN-IDENT ():
2  begin
3      s := c
4      c := NEXT-CHAR()
5      while is-ident-normal-char(c) do
6          s := s · c
7          c := NEXT-CHAR()
8      end
9      last-ident := CREATE-IDENT(s)
10     return T_Ident
11 end
```

Lexikální analýza

```
1  SCAN-NUMBER ():
2  begin
3      s := c
4      c := NEXT-CHAR()
5      while is-digit(c) do
6          s := s · c
7          c := NEXT-CHAR()
8      end
9      last-num := CREATE-NUMBER(s)
10     return T_Number
11 end
```

Syntaktická analýza

Proměnná t :

- naposledy načtený token

Pomocná funkce:

- INIT-SCANNER():
  - inicialuje lexikální analyzátor
  - načte do proměnné c první znak ze vstupu

Syntaktická analýza

Bezkontextová gramatika pro daný jazyk:

$$
\begin{aligned}
S &\to E \langle eof \rangle \\
E &\to T\ G \\
G &\to \varepsilon \mid A\ T\ G \\
A &\to + \mid - \\
T &\to F\ U \\
U &\to \varepsilon \mid M\ F\ U \\
M &\to * \mid / \\
F &\to - F \mid (\ E\ ) \mid \langle ident \rangle \mid \langle num \rangle
\end{aligned}
$$

Syntaktická analýza

Jednou z často používaných metod syntaktické analýzy je tzv. **rekurzivní sestup**:

- Každému neterminálu odpovídá jedna funkce — funkce odpovídající neterminálu A implementuje všechna pravidla s neterminálem A na levé straně.
- Na základě následujícího tokenu se vybírá v rámci dané funkce mezi jednotlivými pravidly.
- Instrukce v těle funkce odpovídají zpracování pravých stran jednotlivých pravidel:
  - výskyt neterminálu B — zavolá se funkce odpovídající neterminálu B
  - výskyt terminálu a — zkontroluje se, že následující token odpovídá terminálu a, pokud odpovídá, načte se další token, pokud neodpovídá, ohlásí se chyba

Syntaktická analýza

$$S \rightarrow E \langle eof \rangle$$

```
1 PARSE ():
2 begin
3     INIT-SCANNER()
4     t := NEXT-TOKEN()
5     e := PARSE-E()
6     if t == T_EOF then  return e
7     else  ERROR()
8 end
```

$$E \rightarrow T\ G$$

```
1 PARSE-E ():
2 begin
3     e1 := PARSE-T()
4     return PARSE-G(e1)
5 end
```

$$G \rightarrow \varepsilon \mid A\ T\ G$$

```
1 PARSE-G (e1):
2 begin
3     if t ∈ {T_Plus, T_Minus} then
4         op := PARSE-A()
5         e2 := PARSE-T()
6         return PARSE-G(MK-BINARY(op, e1, e2))
7     else return e1
8 end
```

$$T \rightarrow F\ U$$

```
1 PARSE-T ():
2 begin
3     e1 := PARSE-F()
4     return PARSE-U(e1)
5 end
```

$$U \rightarrow \varepsilon \mid M\ F\ U$$

```
1 PARSE-U (e1):
2 begin
3     if t ∈ {T_Star, T_Slash} then
4         op := PARSE-M()
5         e2 := PARSE-F()
6         return PARSE-U(MK-BINARY(op, e1, e2))
7     else return e1
8 end
```

Syntaktická analýza

$$A \rightarrow + \mid -$$

```
1  PARSE-A ():
2  begin
3      if t == T_Plus then
4          t := NEXT-TOKEN()
5          return Add
6      else if t == T_Minus then
7          t := NEXT-TOKEN()
8          return Sub
9      else  ERROR()
10 end
```

Syntaktická analýza

$$M \rightarrow * \mid /$$

```
1  PARSE-M ():
2  begin
3      if t == T_Star then
4          t := NEXT-TOKEN()
5          return Mul
6      else if t == T_Slash then
7          t := NEXT-TOKEN()
8          return Div
9      else  ERROR()
10 end
```

$$
\begin{aligned}
F &\longrightarrow -\,F \\
&\mid (\,E\,) \\
&\mid \langle \mathit{ident} \rangle \\
&\mid \langle \mathit{num} \rangle
\end{aligned}
$$

```
1  PARSE-F ():
2  begin
3      switch t do
4          case T_Minus:
5              t := NEXT-TOKEN()
6              e := PARSE-F()
7              return MK-UNARY(Un_minus, e)
8          case T_LParen:
9              t := NEXT-TOKEN()
10             e := PARSE-E()
11             if t == T_RParen then
12                 t := NEXT-TOKEN()
13                 return e
14             else ERROR()
15         case T_Ident:
16             e := MK-VAR(last-ident)
17             t := NEXT-TOKEN()
18             return e
19         case T_Number:
20             e := MK-NUM(last-num)
21             t := NEXT-TOKEN()
22             return e
23         otherwise  ERROR()
24     endsw
25 end
```

Syntaktická analýza

- Pokud funkce končí rekurzivním voláním sebe sama, jako třeba funkce Parse-G(), je možné nahradit tuto rekurzi iterací.
- Funkce Parse-E() a Parse-G() je možné spojit do jediné funkce.
- Podobně ve funkci Parse-U() je možné nahradit rekurzi iterací a funkce Parse-T() a Parse-U() spojit do jediné funkce.

```
1  PARSE-E():
2  begin
3      e1 := PARSE-T()
4      while t ∈ {T_Plus, T_Minus} do
5          op := PARSE-A()
6          e2 := PARSE-T()
7          e1 := MK-BINARY(op, e1, e2)
8      end
9      return e1
10 end
```

```
1  PARSE-T():
2  begin
3      e1 := PARSE-F()
4      while t ∈ {T_Star, T_Slash} do
5          op := PARSE-M()
6          e2 := PARSE-F()
7          e1 := MK-BINARY(op, e1, e2)
8      end
9      return e1
10 end
```