7. Gramatiky a Chomského hierarchie

Formální jazyky a automaty

Jiří Balun

Obsah

Gramatika

Gramatika

Intuice: gramatika je struktura, která formálně popisuje jazyk pravidly, s jejichž pomocí lze generovat řetězce daného jazyka.

Definice

Formální **gramatika** je reprezentována uspořádanou čtveřicí $\mathcal{G} = (N, \Sigma, P, S)$, kde:

1. N je konečná množina neterminálů,
2. Σ je konečná množina terminálů (abeceda), platí $\Sigma \cap N = \emptyset$,
3. P je množina přepisovacích pravidel ve tvaru $P \subseteq V^*NV^* \times V^*$, kde $V = N \cup \Sigma$,
4. S je počáteční neterminál (platí $S \in N$).

- gramatiky popisují mnohem komplexnější jazyky než konečné automaty a RE
- pravidla v P se místo uspořádaných dvojic většinou zapisují ve tvaru $\alpha \to \beta$
  - pravidlo $\alpha \to \beta$ chápeme tak, že při jeho aplikaci se α přepíše na β
  - levá strana pravidla musí obsahovat alespoň jeden neterminál

Příklad: gramatika pro zápis celých čísel

Příklad

Mějme gramatiku $\mathcal{G} = (N, \Sigma, P, S)$ pro zápis celých čísel s pravidly:

$$S \to -I \mid +I \mid I$$
$$I \to DI \mid D$$
$$D \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

- $N = \{S, I, D\}$, kde neterminály obvykle značíme velkými písmeny,
- $\Sigma = \{0, \ldots, 9, +, -\}$, terminály zde reprezentují číslice a znaménka,
- $P = \{\langle S, -I\rangle, \langle S, +I\rangle, \ldots, \langle D, 9\rangle\}$, kde předchozí zápis P čteme jako:
  - pravidla se stejnou levou stranou sloučíme na jeden řádek, pak namísto $A \to \alpha_1, \ldots, A \to \alpha_n$ píšeme jen $A \to \alpha_1 \mid \ldots \mid \alpha_n$ (pravidla oddělujeme pomocí „|“),
  - první řádek je vyhrazen pro pravidla startovního neterminálu,
- S obvykle označuje startovní neterminál.

Relace přímého odvození

Intuice: na řetězec obsahující alepsoň jeden neterminál aplikujeme pravidlo gramatiky, čímž provedeme jeden krok odvození.

Definice

Binární relaci **přímého odvození** $\gamma \Rightarrow_{\mathcal{G}} \delta$ v CFG $\mathcal{G}$ definujeme jako aplikaci konkrétního pravidla na řetězec $\gamma = \eta\alpha\rho$, jehož výsledkem je řetězec $\delta = \eta\beta\rho$, neboli:

$$\eta\alpha\rho \Rightarrow_{\mathcal{G}} \eta\beta\rho$$

kde $\eta, \rho \in (\Sigma \cup N)^*$ a $\alpha \to \beta$ je právě aplikované pravidlo gramatiky $\mathcal{G}$.

- pokud je gramatika zřejmá z kontextu, pak můžeme psát jen $\Rightarrow$ namísto $\Rightarrow_{\mathcal{G}}$
- **větná forma** je libovolný řetězec terminálů a neterminálů
- **derivace** označuje proces, kdy na řetězec aplikujeme konečný počet kroků odvození
  - běžně máme na výběr z několika pravidel, které lze v daném kroku odvození aplikovat
  - jeden řetězec může mít více různých způsobů odvození

Příklad: přímé odvození

Příklad

Mějme gramatiku $\mathcal{G} = (N, \Sigma, P, S)$ pro zápis celých čísel s pravidly:

$$S \to -I \mid +I \mid I$$
$$I \to DI \mid D$$
$$D \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

V $\mathcal{G}$ lze například přímo odvodit:

- $S \Rightarrow -I$, kde z počátečního neterminálu S odvodíme $-I$ pravidlem $S \to -I$,
- $-I \Rightarrow -DI$, kde na neterminál I aplikujeme pravidlo $I \to DI$,
- $-DD \Rightarrow -4D$, kde na první výskyt D aplikujeme pravidlo $D \to 4$,
- $DDD \Rightarrow D6D$, kde na druhý výskyt D aplikujeme pravidlo $D \to 6$,
- a tak dále...

Obecná relace odvození

Definice

Odvození v k krocích v $\mathcal{G}$, kde $k \in \mathbb{N}_0$, definujeme induktivně jako relaci $\Rightarrow_{\mathcal{G}}^{k}$:

- $\Rightarrow_{\mathcal{G}}^{0}$ je identita,
- $\Rightarrow_{\mathcal{G}}^{k+1} = \Rightarrow_{\mathcal{G}}^{k} \circ \Rightarrow_{\mathcal{G}}$.

Pomocí relace $\Rightarrow_{\mathcal{G}}^{k}$ můžeme definovat další užitečné relace:

- relace **odvození v nejvýše k krocích**:

$$\Rightarrow_{\mathcal{G}}^{\leq k} = \bigcup_{i=0}^{k} \Rightarrow_{\mathcal{G}}^{i}$$

- relace **odvození** $\Rightarrow_{\mathcal{G}}^{*}$ (v konečném počtu kroků) a **netriviálního odvození** $\Rightarrow_{\mathcal{G}}^{+}$:

$$\Rightarrow_{\mathcal{G}}^{*} = \bigcup_{i=0}^{\infty} \Rightarrow_{\mathcal{G}}^{i} \qquad\qquad \Rightarrow_{\mathcal{G}}^{+} = \bigcup_{i=1}^{\infty} \Rightarrow_{\mathcal{G}}^{i}$$

Příklad: derivace řetězce

Příklad

Mějme gramatiku $\mathcal{G} = (N, \Sigma, P, S)$ pro zápis celých čísel s pravidly:

$$S \to -I \mid +I \mid I$$
$$I \to DI \mid D$$
$$D \to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

V $\mathcal{G}$ lze například odvodit:

1. $S \Rightarrow -I \Rightarrow -DI \Rightarrow -DD \Rightarrow -4D \Rightarrow -42$
   - platí tedy $S \Rightarrow^5 -42$ (a tedy platí i $S \Rightarrow^* -42$) nebo $S \Rightarrow^3 -DD$
   - naopak -42 nelze odvodit v méně jak 5 korcích (tj. například $S \Rightarrow^4 -42$ už neplatí)
2. $S \Rightarrow -I \Rightarrow -DI \Rightarrow -4I \Rightarrow -4D \Rightarrow -42$
   - odvozujeme stejný řetězec jako v 1, ale změníme pořadí aplikování pravidel
3. $S \Rightarrow I \Rightarrow DI \Rightarrow DDI \Rightarrow DDD \Rightarrow 6DD \Rightarrow 66D \Rightarrow 666$
4. $S \Rightarrow^* w$, kde $w \in \Sigma^*$ (obsahuje jen terminály), pak w je zápis celého čísla

Jazyk generovaný gramatikou

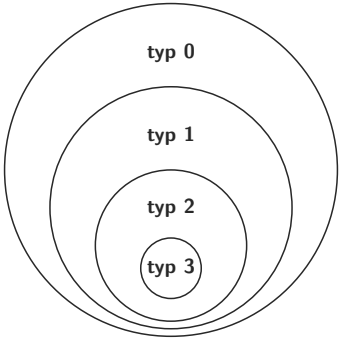
Intuice: gramatika popisuje všechny řetězce terminálů, které jsme schopni odvodit pomocí jejich pravidel v konečném počtu kroků.

Definice

Jazyk generovaný gramatikou $\mathcal{G}$ definujeme jako $L(\mathcal{G}) = \{w \in \Sigma^* \mid S \Rightarrow_{\mathcal{G}}^* w\}$.

- řetězce z jazyka $L(\mathcal{G})$ už neobsahují neterminály
  - přítomnost neterminálu ve větné formě značí, že proces generování ještě neskončil
  - řetězce odvozené z gramatiky $\mathcal{G}$, které obsahují pouze terminály, se označují jako **věty** $\mathcal{G}$
- gramatika sama o sobě nedává návod, jak rozhodnout problém náležení $w \in L(\mathcal{G})$
  - v některých případech je tento problém dokonce neřešitelný
  - existují i jazyky/problémy (rozumně definované), které nelze popsat gramatikou

Chomského hierarchie



- vytvořil ji lingvista Noam Chomsky v roce 1956
- gramatiky rozděluje do čtyř skupin podle tvaru jejich pravidel (a struktury jazyka):
  - **regulární gramatiky** (typ 3) jsou ekvivalentní s DFA
  - **bezkontextové gramatiky** (typ 2) generují jazyky zásobníkových automatů
  - **kontextové gramatiky** (typ 1) generují jazyky lineárně ohraničených automatů
  - **gramatiky bez omezení** (typ 0) generují jazyky *rozponávané* Turingovými stroji
- kromě toho rozděluje gramatiky i podle jejich popisné síly

Regulární gramatika (RG)

Regulární gramatika (typ 3)

Intuice: vhodným omezením tvaru pravidel gramatiky získáme další model, který popisuje/generuje regulární jazyky.

Definice

Regulární gramatika (RG z *regular grammar*) $\mathcal{G} = (N, \Sigma, P, S)$ má všechna pravidla v jednom z těchto tvarů (kde $A, B, S \in N$ a $a \in \Sigma$):

1. $A \to aB$ (neterminál generuje jeden terminál následovaný jedním neterminálem),
2. $A \to a$ (neterminál generuje pouze jeden terminál),
3. $S \to \varepsilon$, pokud se S nevyskytuje na pravé straně jakéhokoliv pravidla (pouze počáteční neterminál může generovat generovat prázdný řetězec).

- takto definovaná regulární gramatika je **pravolineární** (podle tvaru pravidel v 1)
  - analogicky lze definovat **levolineární** RG s pravidly 1 ve tvaru $A \to Ba$
  - levolineární a pravolineární gramatiky jsou ekvivalentní
  - v RG nesmíme míchat levolineární a pravolineární pravidla, tím zvýšíme sílu gramatiky

Příklady: regulární gramatika a protipříklad

Příklad

RG $\mathcal{G}_1 = (\{S, D\}, \{0, \ldots, 9, +, -\}, P, S)$ pro popis celých čísel s pravidly:

$$S \to +D \mid -D \mid 0D \mid \ldots \mid 9D \mid 0 \mid \ldots \mid 9$$
$$D \to 0D \mid \ldots \mid 9D \mid 0 \mid \ldots \mid 9$$

Příklad

Gramatika $\mathcal{G}_2 = (\{S, A, B\}, \{a, b\}, P, S)$, která **není regulární**:

$$S \to aA \mid aB$$
$$A \to Sb$$
$$B \to b$$

$\mathcal{G}_2$ není regulární, protože používá zároveň levolineární i pravolineární pravidla. Kombinací těchto pravidel můžeme vygenerovat neregulární jazyk $L(\mathcal{G}_2) = \{a^n b^n \mid n \in \mathbb{N}\}$.

Ekvivalence RG a DFA

Věta

Ke každé RG $\mathcal{G}$ existuje DFA A takový, že $L(A) = L(\mathcal{G})$.

Idea důkazu

Mějme RG $\mathcal{G} = (N, \Sigma, P, S)$ a sestrojíme NFA $A = (N \cup \{q_f\}, \Sigma, \delta, S, F)$, kde:

- A má jeden stav pro každý neterminál $B \in N$ a navíc jeden stav q_f,
- počáteční stav S odpovídá počátečnímu neterminálu,
- F obsahuje stav q_f, a pokud $\mathcal{G}$ obsahuje pravidlo $S \to \varepsilon$, pak i $S \in F$,
- pro každé pravidlo $A \to aB$ přidáme přechod $\delta(A, a) = B$ a pro pravidla ve tvaru $A \to a$ přidáme přechod $\delta(A, a) = q_f$.

Lze ukázat, že platí $S \Rightarrow_{\mathcal{G}}^{*} w$ právě tehdy, když $\delta(S, w) \in F$, a tedy $L(\mathcal{G}) = L(A)$. □

Věta

Ke každému DFA A existuje RG $\mathcal{G}$ taková, že $L(A) = L(\mathcal{G})$. □

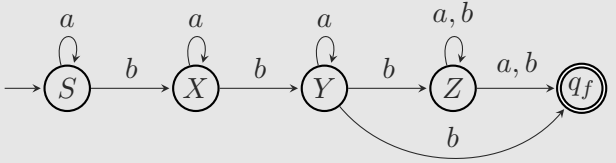
Příklad: převod RG na NFA

Příklad

Mějme gramatiku $\mathcal{G} = (N, \Sigma, P, S)$ s jazykem $L(\mathcal{G}) = \{w \in \Sigma^* \mid \#_b(w) \geq 3\}$ a pravidly:

$$S \to aS \mid bX$$
$$X \to aX \mid bY$$
$$Y \to aY \mid bZ \mid b$$
$$Z \to aZ \mid bZ \mid a \mid b$$

Zkonstruujeme NFA $A = (\{S, X, Y, Z, q_f\}, \{a, b\}, \delta, S, \{q_f\})$ takový, že $L(A) = L(\mathcal{G})$:



Bezkontextová gramatika (CFG)

Motivační příklad: bezkontextová gramatika

Příklad

Formálně popíšeme lispové výrazy, které obsahují `if` a aritmetiku na celých číslech:

$$\begin{aligned}
\langle exprs \rangle &\to \langle expr \rangle \mid \langle expr \rangle\ \langle exprs \rangle \mid \varepsilon \\
\langle expr \rangle &\to \langle num \rangle \mid (\langle func \rangle\ \langle exprs \rangle) \mid (\langle if \rangle) \\
\langle if \rangle &\to \texttt{if}\ \langle expr \rangle\ \langle expr \rangle \mid \texttt{if}\ \langle expr \rangle\ \langle expr \rangle\ \langle expr \rangle \\
\langle func \rangle &\to \texttt{=} \mid \texttt{<} \mid \texttt{>} \mid \texttt{<=} \mid \texttt{>=} \mid \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/} \\
\langle num \rangle &\to 0\langle num \rangle \mid \ldots \mid 9\langle num \rangle \mid 0 \mid \ldots \mid 9
\end{aligned}$$

Příklad odvození:

- $\langle exprs \rangle \Rightarrow^* (\texttt{/}\ 5\ 7)$
- $\langle exprs \rangle \Rightarrow^* (\texttt{+}\ 12\ 34\ 56\ 78)$
- $\langle exprs \rangle \Rightarrow^* (\texttt{if}\ (\texttt{=}\ 1\ 1)\ 42)$
- $\langle exprs \rangle \Rightarrow^* (\texttt{if}\ (\texttt{if}\ (\texttt{<}\ 0\ (\texttt{*}\ 6\ 6\ 6))\ 100\ (\texttt{-}\ 100))\ 1\ 0)$

Příklad – pokračování

$\langle exprs \rangle \Rightarrow \langle expr \rangle$	$\langle expr \rangle \to (\langle if \rangle)$
$\Rightarrow (\langle if \rangle)$	$\langle if \rangle \to \texttt{if}\ \langle expr \rangle\ \langle expr \rangle$
$\Rightarrow (\texttt{if}\ \langle expr \rangle\ \langle expr \rangle)$	$\langle expr \rangle \to (\langle func \rangle\ \langle exprs \rangle)$
$\Rightarrow (\texttt{if}\ (\langle func \rangle\ \langle exprs \rangle)\ \langle expr \rangle)$	$\langle func \rangle \to \texttt{=}$
$\Rightarrow (\texttt{if}\ (\texttt{=}\ \langle exprs \rangle)\ \langle expr \rangle)$	$\langle exprs \rangle \to \langle expr \rangle\ \langle exprs \rangle$
$\Rightarrow (\texttt{if}\ (\texttt{=}\ \langle expr \rangle\ \langle exprs \rangle)\ \langle expr \rangle)$	$\langle exprs \rangle \to \langle expr \rangle$
$\Rightarrow (\texttt{if}\ (\texttt{=}\ \langle expr \rangle\ \langle expr \rangle)\ \langle expr \rangle)$	$\langle expr \rangle \to \langle num \rangle$
$\Rightarrow (\texttt{if}\ (\texttt{=}\ \langle num \rangle\ \langle expr \rangle)\ \langle expr \rangle)$	$\langle num \rangle \to 1$
$\Rightarrow (\texttt{if}\ (\texttt{=}\ 1\ \langle expr \rangle)\ \langle expr \rangle)$	$\langle expr \rangle \to \langle num \rangle$
$\Rightarrow (\texttt{if}\ (\texttt{=}\ 1\ \langle num \rangle)\ \langle expr \rangle)$	$\langle num \rangle \to 1$
$\Rightarrow (\texttt{if}\ (\texttt{=}\ 1\ 1)\ \langle expr \rangle)$	$\langle expr \rangle \to \langle num \rangle$
$\Rightarrow (\texttt{if}\ (\texttt{=}\ 1\ 1)\ \langle num \rangle)$	$\langle num \rangle \to 4\langle num \rangle$
$\Rightarrow (\texttt{if}\ (\texttt{=}\ 1\ 1)\ 4\langle num \rangle)$	$\langle num \rangle \to 2$
$\Rightarrow (\texttt{if}\ (\texttt{=}\ 1\ 1)\ 42)$	

Bezkontextová gramatika (typ 2)

Intuice: pravidla bezkontextových gramatik umožňují z jednoho neterminálu odvodit libovolný řetězec terminálů a neterminálů.

Definice

Bezkontextová gramatika (CFG z *context-free grammar*) $\mathcal{G} = (N, \Sigma, P, S)$ má všechna pravidla ve tvaru (kde $A \in N$):

- $A \to (\Sigma \cup N)^*$

- název *bezkontextová* vychází z přepisování neterminálu bez ohledu na kontext, tedy symboly/řetězce v okolí přepisovaného neterminálu (to umí až kontextové gramatiky)
- L je **bezkontextový jazyk** pokud existuje CFG $\mathcal{G}$ taková, že $L(\mathcal{G}) = L$
  - bezkontextové jazyky budeme značit CFL (z *context-free language*)
- gramatika je bezkontextová jen na základě tvaru svých pravidel
  - musíme rozlišovat typ gramatiky a typ jejího jazyka
  - jazyk CFG totiž může být regulární (pak existuje regulární gramatika se stejným jazykem)

Příklady: CFG a jejich jazyky

Příklad

CFG $\mathcal{G}_1 = (\{S, A\}, \{a, b\}, P, S)$ s pravidly:

$$S \to \varepsilon \mid AbAbAS$$
$$A \to AA \mid a \mid \varepsilon$$

generuje jazyk $L(\mathcal{G}_1) = \{w \in \Sigma^* \mid w$ obsahuje sudý počet znaků $b\}$, který je regulární.

Příklad

CFG $\mathcal{G}_2 = (\{S\}, \{a, b\}, P, S)$ s pravidly $S \to \varepsilon \mid aSb$ a jazykem $L(\mathcal{G}_2) = \{a^n b^n \mid n \in \mathbb{N}_0\}$.

Příklad

CFG $\mathcal{G}_3 = (\{S\}, \{a, \ldots, z\}, P, S)$ s pravidly:

$$S \to \varepsilon \mid aSa \mid \ldots \mid zSz \mid a \mid \ldots \mid z$$

generuje jazyk palindromů, tj. $L(\mathcal{G}_3) = \{w \in \Sigma^* \mid w = w^R\}$.

Nejlevější derivace

Definice

Derivaci nazveme **nejlevější**, pokud ve větné formě přepisujeme vždy nejlevější neterminál.

- nejlevější derivaci značíme $\Rightarrow_{lm}$ (jeden krok) a $\Rightarrow_{lm}^{k}$ (více kroků)
- analogicky můžeme definovat i nejpravější derivaci $\Rightarrow_{rm}$

Příklad

Mějme CFG $\mathcal{G}_1 = (\{S, A\}, \{a, b\}, P, S)$ s pravidly

$$S \to \varepsilon \mid AbAbAS$$
$$A \to AA \mid a \mid \varepsilon$$

- $S \Rightarrow_{lm} AbAbAS \Rightarrow_{lm} abAbAS \Rightarrow_{lm} abbAS \Rightarrow_{lm} abbaS \Rightarrow_{lm} abba$
- $S \Rightarrow_{rm} AbAbAS \Rightarrow_{rm} AbAbA \Rightarrow_{rm} AbAba \Rightarrow_{rm} Abba \Rightarrow_{rm} abba$
- $S \Rightarrow AbAbAS \Rightarrow AbbAS \Rightarrow AbbA \Rightarrow Abb \Rightarrow bb$ (není ani nejlevější ani nejpravější derivace řetězce bb)

Derivační strom

Intuice: derivační strom je datová struktura reprezentující proces odvození nějakého řetězce v CFG.

Definice

Strom T nazveme **derivačním stromem** řetězce $\alpha \in (\Sigma \cup N)^*$ v CFG $\mathcal{G} = (N, \Sigma, P, S)$ právě tehdy, když platí:

1. každý uzel je označen symbolem z $N \cup \Sigma \cup \{\varepsilon\}$, přičemž kořen je označen S,
2. nelistové (vnitřní) uzly jsou označeny pouze neterminály,
3. pokud má nelistový uzel označení $A \in N$ a jeho potomci zleva doprava jsou označení $\beta_1, \ldots, \beta_k$, kde $\beta_1, \ldots, \beta_k \in \Sigma \cup N$, pak v $\mathcal{G}$ existuje pravidlo $A \to \beta_1 \ldots \beta_k$,
4. pokud je uzel označen ε, pak je list a nemá žádné sourozence,
5. zřetězením listových uzlů dostaneme α, neboli α je **derivace** stromu T.

- pro $\alpha \in (N \cup \Sigma)^*$ platí, že $S \Rightarrow_{\mathcal{G}}^* \alpha$ právě tehdy, když existuje strom v $\mathcal{G}$ s derivací α
- každému derivačnímu stromu odpovídá jediná levá (případně pravá) derivace

Příklad: derivační strom

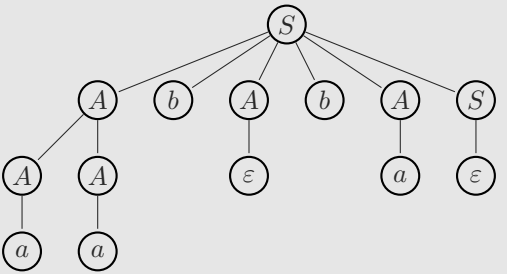
Příklad

Mějme CFG $\mathcal{G}_1 = (\{S, A\}, \{a, b\}, P, S)$ s pravidly

$$S \to \varepsilon \mid AbAbAS$$
$$A \to AA \mid a \mid \varepsilon$$

a derivaci řetězce $aabba \in L(\mathcal{G}_1)$ a k ní sestavený derivační strom:

$S \Rightarrow AbAbAS \Rightarrow AAbAbAS \Rightarrow aAbAbAS \Rightarrow aabAbAS \Rightarrow aabbAS \Rightarrow aabbaS \Rightarrow aabba$



Víceznačnost CFG a CFL

Motivace: pro určení významu nebo korektnosti některých vět je potřeba analyzovat jejich derivační strom, proto dělíme CFL a CFG na základě unikátnosti jejich derivačních stromů.

Definice

Bezkontextová gramatika $\mathcal{G}$ je **víceznačná** (nejednoznačná) pokud existuje $w \in L(\mathcal{G})$, které má v $\mathcal{G}$ dva různé derivační stromy. V opačném případě je $\mathcal{G}$ **jednoznačná**.

Definice

Bezkontextový jazyk L je **dědičně víceznačný** pokud je víceznačná každá gramatika, která generuje L.

- příkladem dědičně víceznačného jazyka je jazyk $\{a^i b^j c^k \mid i = j \vee j = k\}$
- je potřeba rozlišovat mezi víceznačností jazyka a gramatiky
- neexistuje způsob jak algoritmicky ověřit, že CFG je víceznačná

Příklad: víceznačnost

Příklad

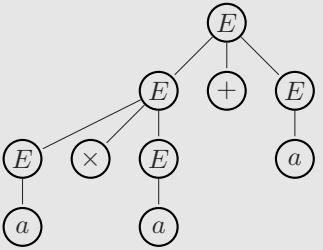
Mějme víceznačnou gramatiku $\mathcal{G} = (\{E\}, \{+, -, a, (,)\}, P, S)$ s pravidly:

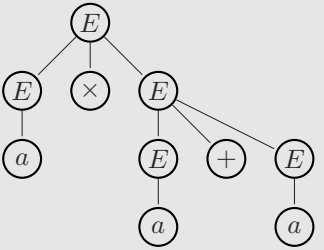
$$E \to E + E \mid E \times E \mid (E) \mid a$$

Řetězec $a \times a + a \in L(\mathcal{G})$ má dvě různé derivace, a tedy i dva derivační stromy:

1. $E \Rightarrow E + E \Rightarrow E \times E + E \Rightarrow a \times E + E \Rightarrow a \times a + E \Rightarrow a \times a + a$
2. $E \Rightarrow E \times E \Rightarrow E \times E + E \Rightarrow a \times E + E \Rightarrow a \times a + E \Rightarrow a \times a + a$

Jazyk $L(\mathcal{G})$ není dědičně víceznačný, tj. existuje jednoznačná gramatika, která jej generuje.





Backus-Naurova forma (BNF)

- BNF a její varianty umožňují formální zápis gramatik programovacích jazyků
- oproti formálním gramatikám jsou pravidla obvykle psána za operátor `::=` namísto $\rightarrow$
- neterminální symboly jsou většinou psány do zkosených závorek, například `<expr>`
- terminální symboly/řetězce jsou uzavřeny do uvozovek, například `"0"`
- tzv. rozšířená BNF (zkráceně EBNF) umožnuje na pravou stranu pravidel psát RE:
  - výraz ve složených závorkách lze libovolně opakovat
  - výraz v hranatých závorkách je volitelný, tj. jeden nebo nula výskytů
  - kulaté závorky používáme standardně pro změnu precedence operací

Příklad

Příklad gramatiky v EBNF pro zápis aritmetických operací na celých číslech:

```
<expr> ::= <term> { ("+" | "-") <term> }
<term> ::= <factor> { ("*" | "/") <factor> }
<factor> ::= <number> | "(" <expr> ")"
<number> ::= [ "-" ] <digit> { <digit> }
<digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

Příklad: použití CFG v překladačích

BNF a ENBF se používají v generátorech parserů při psaní překladačů:

- pomocí CFG specifikujeme jazyk, ke kterému generátor vytvoří syntaktický analyzátor
- příklady generátorů: Yacc, GNU Bison, ANTLR atd.

Překladače (například jazyka C do assembleru) postupují v několika krocích:

- **lexikální analýza** – převod z textu na tzv. *lexémy* (samostatné tokeny pro klíčová slova, identifikátory proměnných, čísla atd.) pomocí konečných automatů
- **syntaktická analýza** – na základě CFG se z lexémů sestavý derivační strom daného zdrojového kódu (v tomto kroku jsou odhaleny syntaktické chyby programu)
- **sémantická analýza** – na derivačním stromu se poté provádí kontrola a analýza významu kódu (například určení rozsahu platnosti proměnných, typová kontrola atd.)
- **optimalizace** – derivační strom se často převádí do lineárního mezikódu, na kterém se provádějí optimalizace (například rozvinutí cyklů, výpočet konstantních výrazů atd.)
- **generování kódu** – z výsledného mezikódu se vygeneruje finální kód

Příklad: gramatika jazyka C

- na webu jsou dostupné celé gramatiky jazyka C (zapsané v různých notacích)
- gramatika v EBNF: https://cs.wmich.edu/~gupta/teaching/cs4850/sumII06/The%20syntax%20of%20C%20in%20Backus-Naur%20form.htm
- varianta BNF pro Yacc: https://www.quut.com/c/ANSI-C-grammar-y.html

```
primary_expression
        : IDENTIFIER
        | constant
        | string
        | '(' expression ')'
        | generic_selection
        ;

constant
        : I_CONSTANT  /* includes character_constant */
        | F_CONSTANT
        | ENUMERATION_CONSTANT /* after it has been defined as such */
        ;

enumeration_constant  /* before it has been defined as such */
        : IDENTIFIER
        ;
...
```