UNIVERZITA PALACKÉHO V OLOMOUCI
KATEDRA INFORMATIKY

M. Rotter, T. Kukučka, J. Zehnula

KMI/FJAA – Formální jazyky a automaty

22. března 2012

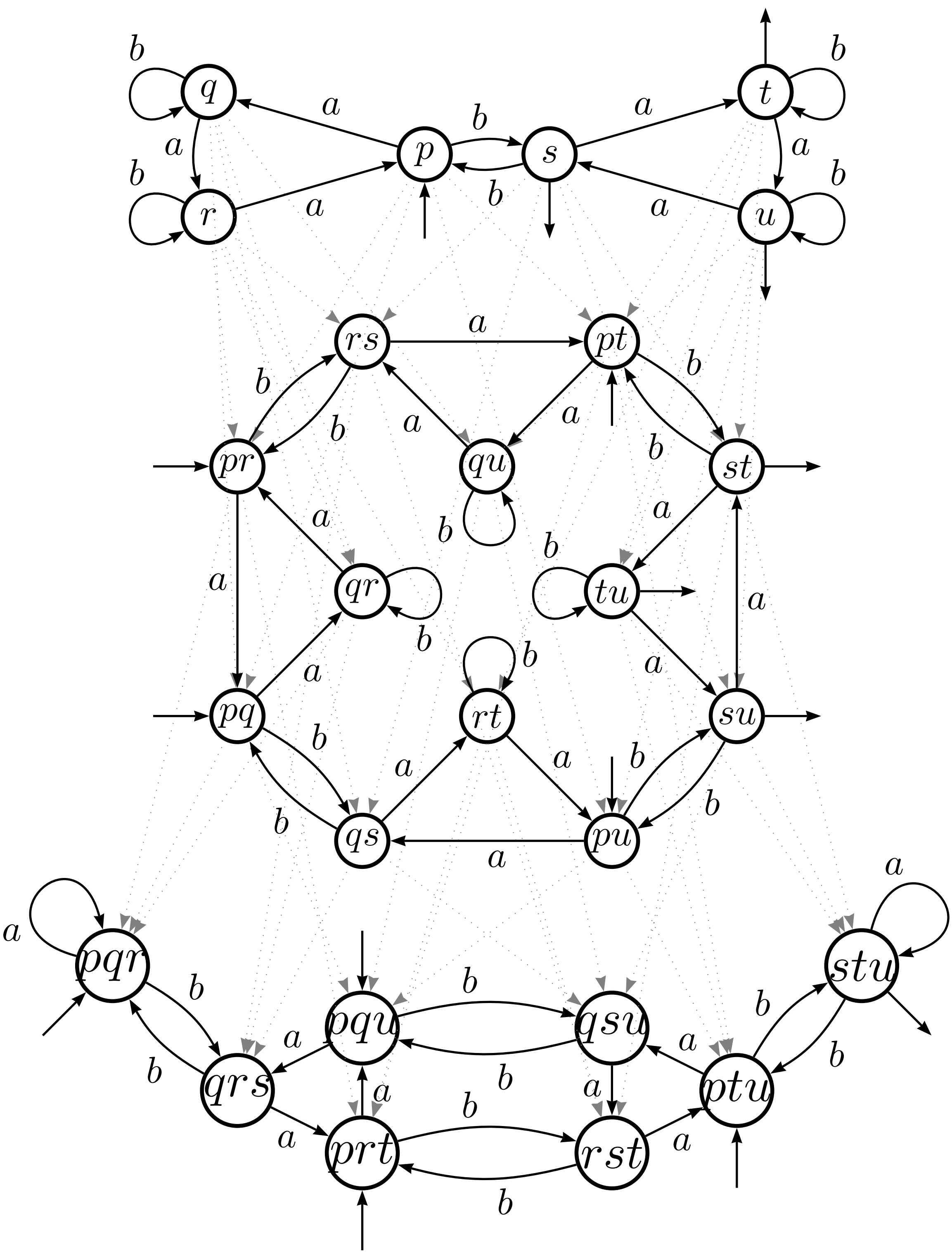
Abstrakt

Tento dokument je pouze přepisem zápisků a poznámek z přednášek předmětu KMI/FJAA. Přednášel doc. Vilém Vychodil PhD.

Obsah

Seznam obrázků

Seznam tabulek



q
r
p
s
t
u
rs
pt
pr
qu
st
qr
tu
pq
rt
su
qs
pu
pqr
stu
pqu
qsu
qrs
ptu
prt
rst
a
b

1. Historie

Počátek úvah, jež byly později základem seriozního zkoumání formálních jazyků potažmo automatů se datuje do 30. let. Průkopníkem této oblasti byl Noam Chomsky [1].

Jako příklad selhání autora programovacího jazyka si uveďme jazyk **Fortran**, jehož konstrukce byla po syntaktické stránce špatná, což vedlo ke **gramatické nejednoznačnosti** tohoto jazyka.

2. Kódová analýza

2.1. Lexikální analýza

Dělení kódu na tokeny[2], jež se zapisují například ve stylu ⟨ znak, identifikátor ⟩.me Příkladem je tedy i token $\langle =, assignment \rangle$ a jiné.

2.2. Syntaktická analýza

Syntaktická analýza vytváři stromovou závislost jednotlivých tokenů, jejíž reprezentace se nazývá *syntaktický-derivační strom*. V rámci této analýzy rozlišme:

1. Teorii jazyků, jenž se zabýva stavbou jazyka (respektive jeho syntaxí) a poskytuje tzv. **generativní aparát**. Dodejme, že gramatika říká, v jakém tvaru může být zapsán validní program.
2. Teorii automatů, jež poskytuje tzv. **analytický aparát**. Dodejme, že automatem se rozumí de-facto jednoduchý algoritmus.

3. Základní pojmy

- **Symbol** (případně znak). Jedná se o syntaktický pojem (význam tedy nehraje roli), který představuje *jméno* (analogicky k *písmenu* z přirozeného jazyka). Mezi symboly počítejme například **0, +, Š, while**.
- **Abeceda**. Abecedou rozumíme množinu (například množinu X) všech přípustných *symbolů* (znaků), přičemž taková množina je neprázdná (tedy $|x| > 0$) a konečná. Konečnost množiny je omezení dané reprezentovatelností množiny v rámci počitačové techniky. Abecedy značíme řeckými písmeny. Například $\Sigma, \Sigma^{'}, \Gamma, \ldots, \Omega$. Například $\Sigma = \{a, b, c\}$.
- **Řetězec** (případně slovo). Jedná se o konečnou posloupnost symbolů (znaků) vybraných z nějaké dané abecedy. Například $\langle a_1, a_2, \ldots, a_n \rangle \in \Sigma, n$ nazvěme *délkou řetězce*. Formálně definujme řetězec jakožto *zobrazení*

  $$x : \{a, b, c, d, \ldots, i, j, \ldots\} \to \Sigma$$

  , kde

  $$1 \to a, 2 \to b, 3 \to c$$

  a tak podobně. Délku řetězce označme $|x|$.

[1] Jméno této osoby čti [*čomski*] a zapamatuj si ke státnícím, že Chomski byl *nebezpečný levicový intelektuál.*
[2] Překládej jako *část, díl nebo také fráze.*

- **Prázdný řetězec**. Jedná se o řetězec, pro který platí, že $|x| = 0$ a značíme jej ε, přičemž platí následující zápis:
$$\varepsilon \subseteq \emptyset \to \Sigma$$
Prázdný řetězec **není** symbolem, tedy $\varepsilon \notin \Sigma$.

Věta 1: Nad k-prvkovou abecedou je právě k^n řetězců délky n.

Σ^* označuje množinu všech řetězců nad abecedouΣ.
Σ^+ označuje množinu všech řetězců nad abecedouΣ vyjma ε.

4. Operace s řetězci

- **Zřetězení** (konkatenace). Jde v podstatě o spojení[3] dvou řetězců v daném pořadí do jednoho řetězce.

  Příklad 1: Mějme dva řetězce a, b:
$$a_1 \dots a_n \text{ a } b_1 \dots b_m$$
  Pak jejich zřetězení má tvar
$$a_1 \dots a_n b_1 \dots b_m$$
  Identifikátorem[4] operace zřetězení je $\circ$, například $x \circ y$ je zřetězením řetězců x a y. Formálně takto:
$$x : \{1, \dots, n\} \to \Sigma$$
$$y : \{1, \dots, m\} \to \Sigma$$
$$x \circ y : \{1, \dots, nm\} \to \Sigma$$

  Poznámka 1: Algebraicky je tatáž operace zapsána jako $\langle \Sigma^*, \circ, \varepsilon \rangle$.

- **Rovnost řetězců** Pro prohlášení dvou řetězců za sobě rovné v žádaném smyslu je třeba splnit obecně dvě následující podmínky:
  1. Oba řetězce mají stejnou délku, tedy $|x| = |y|$.
  2. Bude-li délka označena jako n, pak musí platit, že $\forall i | i \in \{1, \dots, n\}, x(i) = y(i)$. Tedy každé dva k sobě náležící symboly z daných řetězců jsou si rovny.

  Uvažujeme-li rovnost řetězců, pak je záhodno uvažovat následující pojmy:
  - **Prefix** řetězce. Označme jej $Pfx(x) = \{y | \exists z \text{ tak, že } yz = x\}$.
  - **Infix** řetězce. Označme jej $Ifx(x) = \{y | \exists z_1, z_2 \text{ tak, že } z_1 y z_2 = x\}$.
  - **Sufix** řetězce. Označme jej $Sfx(x) = \{y | \exists z \text{ tak, že } zy = x\}$.

  Věta 2:
$$xy = xz \implies y = z$$
$$yx = zx \implies y = z$$
  Algebraicky je operace zapsána jako $\langle \Sigma^*, \cdot, \varepsilon \rangle$.

[3] Pro milovníky jazyka Scheme můžeme tuto operaci přirovnat k proceduře *append*
[4] Identifikátor zřetězení se velmi čast v zápisech zřetězení vynechává.

Věta 3: Vyslovme předpoklad, že platí $xy = uv$. Pak platí právě jedno z těchto tvrzení:

$$x = u, y = v$$
$$|x| > |u| \text{ a } \exists w | w \neq \varepsilon, \text{ tak že } x = uw \text{ a } v = wy$$
$$|x| < |u| \text{ a } \exists w | w \neq \varepsilon, \text{ tak že } u = xw \text{ a } y = wv$$

- **N-tá mocnina** řetězce.

$$x^n = \left\{ \begin{array}{ll} x & \text{pro } n = 1 \\ xx^{n-1} & \text{v ostatních případech} \end{array} \right\}$$

respektive

$$x^n = \left\{ \begin{array}{ll} \varepsilon & \text{pro } n = 0 \\ xx^{n-1} & \text{v ostatních případech} \end{array} \right\}$$

Poznámka 2: Mějme na paměti, že operace mocnění má vyšší prioritu než-li operace konkatenace (zřetězení).

Věta 4: Mějme u a $v \in \Sigma^*$, pak platí $uv = vu$ (komutativita), právě tehdy, když $\exists z | z \in \Sigma^*$ a nezáporná celá čísla p, q tak, že $u = z^p$ a $v = z^a$.

Předpokládejme, že po p, z, q máme $u = z^p, v = z^q$. Pak obecně platí následující zápis:

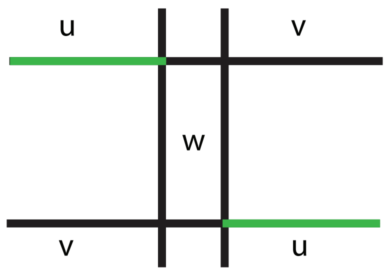
$$uv = z^p z^q = z^{p+q} = z^p z^q = vu$$

Předpokládejme, že $uv = vu$. Indukcí přes $|uv|$ předpokládáme, že tvrzení platí pro libovolné dva řetězce, jejichž délka zřetězení je menší než-li $|uv|$. Mohou nastat tyto případy:

1. $|u| = |v|$, pak $u = v$, pak $z = u, p = q = 1$
2. $|u| < |v|$

Berme v potaz také následující zápis doplněný obrázkem: 1.

$$uv = v$$
$$wu = v$$
$$uw = wu$$
$$|uw| < |uv|, \text{ tedy } \exists z, p, q \text{ tak, že } u = z^p, w = z^q, v = z^{p+q}$$



Obrázek 1. Grafická pomůcka ke komutativitě zřetězení řetězců.

5. Formální jazyk

Zaveďme si pojem *formální jazyk nad množinou (resp. abecedou)* Σ^*. Označme tento jazyk jako L. Pak platí tato tvrzení:

$$L \subseteq \Sigma^* \text{ (každá podmnožina abecedy je jazykem)}$$
$$L = \emptyset \text{ (prázdný jazyk)}$$
$$L = \{\varepsilon\} \text{ (jazyk s prázdným řetězcem)}$$
$$L = jazykC++ \text{ (jazyk C++)}$$
$$\vdots$$

Pozor, obecně platí že prázdný jazyk $\neq$ jazyk s prázdným řetězcem.

6. Lexikografické uspořádání

Předpokládejme uspořádání na množině Σ^*[5]. Nazvěme toto uspořádání *striktním totálním.* Pak toto uspořádání například pro $\Sigma = \{a_1, \ldots, a_n\}$ je $a_1 < a_2 < a_3 < \ldots < a_n$.

Totální striktní uspořádání označme $<_l$.

Položme $x <_l y$ pro $x, y \in \Sigma^*$. To ale platí pokud platí alespoň jedno z následujících dvou tvrzení:

1. $|x| < |y|$
2. $|x| < |y|$ a $\exists i$ tak, že $x(i) < y(i)$ a zároveň $x(j) < y(j)$ pro $\forall j | j < i$

Příklad 2: $\Sigma = \{0, 1\}$. Triviálně tedy $0 < 1$. Následně striktně $\varepsilon <_l 0 <_l 1 <_l 00 <_l 01 <_l 10 <_l 11$.

Věta 5: Striktní totální uspořádání je asymetrické a tranzitivní. A pro $x \neq y$ platí buď $x <_l y$ nebo $y <_l x$.

Důsledek 1: Důsledkem věty 5 je tvrzení, že množina Σ^* je spočetně nekonečná. Dodejme, že jazyk je (obvykle) spočetná množina.

7. Operace nad jazyky

7.1. Množinové

Množinové operace nad jazyky jsou prakticky totožné operacím na kterýchkoliv jiných množinách. Můžeme tedy použít množinový průnik, sjednocení, komplement (doplněk) nebo rozdíl.

7.2. Ostatní

- **Zřetězení** (produkt) množin. Vyjádřeme produkt takto:

  $$L_1 L_2 = \{xy | x \in L_1, y \in L_2\}$$

  Produkt množin není obecně komutativní, ale je asociativní, přičemž prázdná množina tuto operaci anihiluje. Uveďme si rovněž monoid $\langle 2^{\Sigma^*}, \circ, \{\varepsilon\} \rangle$.

[5] Zopakujme si, že Σ^* je množina všech řetězců nad Σ.

- **Mocnina** jazyka. Mocninu vyjádříme takto:

$$L^n = \left\{ \begin{array}{ll} \{\varepsilon\} & \text{pro } n = 0 \\ LL^{n-1} & \text{pro } n > 1 \end{array} \right\}$$

- **Kleeneho**[6] uzávěr neboli **iterace**. Tento uzávěr vyjádříme takto:

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

- **Pozitivní** uzávěr neboli pozitivní iterace. Tento uzávěr vyjádříme takto:

$$L^* = \bigcup_{i=1}^{\infty} L^i$$

  Všimněte si podobností mezi těmito dvěma uzávěry. Pozitivní uzávěr vynechává prázdný řetezec.

8. Gramatiky

Jak víme, tak jazyky mohou být *nekonečné* ve smyslu, že obsahují nekonečný počet slov. Nabízí se tedy otázka, jak tyto jazyky rozumně popsat, jak je reprezentovat resp. jak vytvořit *konečnou* sadu pravidel, jejichž aplikace by vedla k opětovné generaci původního jazyka.

8.1. Přepisovací generovací pravidla

Pravidlem rozumíme zpravidla každou takto definovanou dvojici.

$$\langle x, y \rangle \in \Sigma^* \times \Sigma^*$$

Pak neformálně tvrdíme, že x *se přepisuje* na y. Nutno dodat, že předchozí zápis lze zapsat i například takto.

$x \rightarrow y$, kde o symbol $\rightarrow \notin \Sigma$ můžeme prohlásit za tzv. metasymbol.

8.1.1. Vlastnosti pravidel

- Nezkracující pravidlo je pravidlo, o kterém platí, že $|x| \leq |y|$. Tedy aplikaci tohoto pravidla na řetězec určitě nevznikne řetězec kratší, než-li jeho předloha.
- ε - pravidlo je pravidlo tvaru $x \rightarrow \varepsilon$.

8.1.2. Příklady pravidel

Příklad 3: Mějme zadání abecedy $\Sigma = \{a, b, c\}$. Pravidla s využítím této abecedy by mohla být například tato.

$$aa \rightarrow bc$$
$$bb \rightarrow y$$
$$c \rightarrow \varepsilon$$

[6]Stephen Cole Kleene je známý matematik, jenž se významně podílel na položení základů teoretických počítačových věd.

Příklad 4: Mějme další zadání abecedy $\Sigma = \{expr, +, \times\}$. Pravidla s využitím této abecedy by mohla být například tato.

$$expr \to expr + expr$$
$$expr \to expr \times expr$$

8.1.3. Přímé odvozování řetězců pomocí pravidel

Uvažujme odvozovací pravidlo $x \to y$ nad abecedou Σ, pak řekneme, že řetězec v **je přímo odvozen** z řetězce u pomocí pravidla $x \to y$, pokud $\exists p, q \in \Sigma^*$ tak, že

$$u = pxq$$
$$v = pyq$$

Značení předchozí operace je následující:

$$u \Rightarrow_{x \to y} v$$

Slovně bychom tento zápis vystihli jako „přímý přepis dle pravidla $x \to y$."

Řetězec v vznikne přímým přepisem z u pomocí pravidel $P \subseteq \Sigma^* \times \Sigma^*$, pokud $\exists \pi \in P$ tak, že $u \Rightarrow_\pi v$.

Značme $u \Rightarrow_P v$. P je množinou užitých pravidel. P i $\Rightarrow_p$ jsou binární relace na Σ^* a $P \subseteq \Rightarrow_p$, tedy „P je podmnožinou šipky p." Platí, že $x \to y \in P$ a $x \Rightarrow_{x \to y} y$.

Příklad 5: Mějme abecedu $\Sigma = \{a, b, c\}$ a soubor pravidel $P = \{aa \to bc, a \to cab, bb \to \varepsilon\}$. Pak by odvození v jednom kroku mohla vypadat například takto:

$$baaa \to bbca$$
$$bac \to bcabc$$

Definice 1: Definujme pojem **derivace**. Jedná se o posloupnost řetězců v tomto tvaru.

$$x_0, \ldots, x_k, \text{ kde } k \geq 0 \text{ a kde } \{x_0, \ldots, x_k\} \in \Sigma^*$$

se nazývá **P-derivace délky k**, pokud $x_{i-1} \Rightarrow_p x_i, \forall 1 \leq i \leq k$. Symbolicky totéž $x_0 \Rightarrow_p x_1 \Rightarrow_p \ldots \Rightarrow_p x_k$. Počet odvození tedy značí *délku* derivace.

Pokud pro $u, v \in \Sigma^* \exists$ P-derivace $u = x_0 \ldots x_k = v$, pak říkáme, že v je odvozeno z u pomocí pravidel z P, což značíme například $u \Rightarrow_P^* v$, tímto je pochopitelně myšleno odvození ve více krocích. Platí, že $P \subseteq \Rightarrow_P \subseteq \Rightarrow_P^+$.

Příklad 6: Mějme abecedu $\Sigma = \{a, \ldots, z\}$ a pravidla stejná jako v příkladu 5. Nyní odvozujeme například takto.

$$b\underline{aa}a, \underline{bb}ca, c\underline{a}, c\underline{cab}$$

8.2. Formální gramatiky

Mějme následující entity.

- $\Sigma \ldots$ abeceda terminálních symbolů (tyto symboly tvoří řetězce daného jazyka).
- $N \ldots$ abeceda neterminálních symbolů (tyto symboly se užívají k řízení průběhu odvozování).

Dodejme, že obě množiny by měly být neprázdné a konečné.

Definice 2: Odvozovací pravidlo $x \to y$ se nazývá *generativní*, pokud x obsahuje alespoň jeden neterminální symbol.

Definice 3: Mějme strukturu $G = \langle N, \Sigma, R, S \rangle$, kde N je abecedou neterminálních symbolů, Σ je abecedou terminálních symbolů, P je množinou odvozovacích pravidel a $S \in N$ je tzv. počátečním resp. startovním terminálem. Pak tuto čtveřici nazveme **gramatikou**.

Poznámka 3: Pokud chceme vyjádřit, že z jednoho symbolu odvozuje několik možných alternativ, tak to zapíšeme místo klasického dlouhé zápisu $y \to x_1, y \to x_2, \ldots$ pomocí zkrácené notace např. $y \to x_1|x_2|\ldots$

Příklad 7: Gramatikou může být i ta následující.

$$\begin{aligned} N &= \{\varepsilon, S, D, I\} \\ \Sigma &= \{0, \ldots, 9, +, -\} \\ P &= \{S \to -I| + I|I, I \to DI|D, D \to 0|1|\ldots|9\} \\ G &= \langle N, \Sigma, P, S \rangle \end{aligned}$$

Příklad 8: Nebo tato.

$$\begin{aligned} N &= \{S, X, Y\} \\ \Sigma &= \{a, b, c\} \\ P &= \{S \to XcYcX, X \to aX, X \to bX, X \to cX, X \to \varepsilon, y \to abY, Y \to ab\} \\ G &= \langle N, \Sigma, P, S \rangle \end{aligned}$$

Definice 4: Každý řetězec $x \in (N \cup \Sigma)^*$, pro který platí $S \to^* x$, je **větná forma** gramatiky $G = \langle N, \Sigma, P, S \rangle$. Větná forma se nazývá **větou**, pokud $x \in \Sigma^*$.

Definice 5: Jazyk generovaný gramatikou definujme jako:

$$L(G) = \{x \in \Sigma^* | S \Rightarrow_G^* x\}$$

Příklad 9: Tento příklad čerpá gramatiku z příkladu 1.

$$\begin{aligned} S &\Rightarrow_G^* abbccYcX \\ S &\Rightarrow_G^* Xcabababc \\ S &\Rightarrow_G^* cYcbaX \\ S &\Rightarrow_G^* abbccabca \\ S &\Rightarrow_G^* cababc \end{aligned}$$

Definice 6: Gramatiky G_1 a G_2 jsou **ekvivalentní**, pokud generují stejný jazyk.

8.3. Hierarchie gramatik

- **Gramatiky typu 0** – jedná se o gramatiky bez omezení.
- **Gramatiky typu 1** – jedná se o tzv. *kontextové* nebo *kontextově závislé* gramatiky. Ty splňují následující omezení na tvar pravidel. Pro každé pravidlo gramatik tohoto typu platí, že:

1. Buď je (pravidlo) ve tvaru $pAq \rightarrow p \times q$, kde $p, q \in (\Sigma \cup N)^*, A \in N, x \in (\Sigma \cup N)^*$, kde p a q se nazývají levým resp. pravým **kontextem**.
2. Nebo je (pravidlo) ve tvaru $S \rightarrow \varepsilon$, kde S je startovní terminál gramatiky, ale pouze za předpokladu, že S se nevyskytuje na pravé straně žádného pravidla.

- **Gramatiky typu 2** – jedná se o tzv. *bezkontextové* gramatiky, jenž obsahují pravidla ve tvaru:

$$A \rightarrow x, \text{ kde } A \in N, x \in (\Sigma \cup N)^*$$

Příklad 10: Mějme tuto gramatiku:

$$\begin{aligned} G &= \langle N, \Sigma, P, S \rangle \\ N &= \{A, S\} \\ \Sigma &= \{0, 1\} \\ P &= \{S \rightarrow 0A, A \rightarrow \varepsilon\} \end{aligned}$$

- **Gramatiky typu 3** – jedná se o tzv. *regulární* resp. *pravolineární* gramatiky, které obsahují pravidla ve třech následujících tvarech:

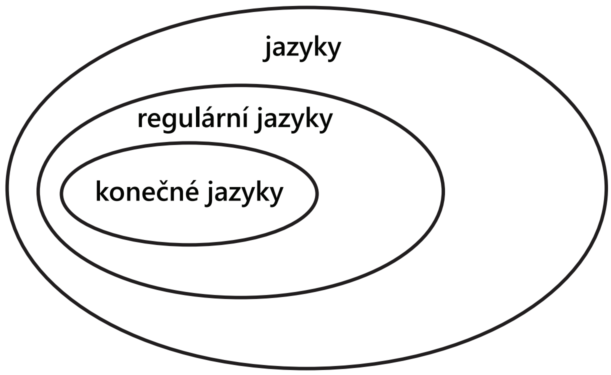
1. $A \rightarrow bB$, kde $A, B \in N, b \in \Sigma$
2. $A \rightarrow a$
3. $S \rightarrow \varepsilon$

Poznámka 4: Každý konečný jazyk je regulární.

Důkaz 1: Mějme jazyk $L = \{x_1, \ldots, x_n\}$. Abychom tento jazyk prohlásili za regulární, tak je třeba najít regulární gramatiku, která tento jazyk generuje.

Mějme tedy nějaké dané Σ a S a zvolme N. Následně platí $\forall x_i \in L$ je dvojího typu:

1. $x_i = \varepsilon$ a následně $S \rightarrow \varepsilon$
2. $x_i = a_1 \ldots a_k$ a následně $S \rightarrow a_{i1}A^{'}, A^{'} \rightarrow a_{i2}A^{''}, \ldots\ldots, A^{k-1} \rightarrow a_{ik}A^k$



Obrázek 2. Vychodilovo „vajíčko."

Příklad 11:

$$\begin{aligned} N &= \{S\} \\ \Sigma &= \{a, b\} \\ P &= \{S \rightarrow aSb|\varepsilon\} \end{aligned}$$

$$L(G) = \{a^n b^n | n \leq 0\}$$

Máme tedy *bezkontextový* jazyk.

Příklad 12:

$$\begin{aligned} N &= \{S\} \\ \Sigma &= \{a, b\} \\ P &= \{S \rightarrow SS|aSb|bSa|\varepsilon\} \end{aligned}$$

$L(G)$ je *bezkontextový* jazyk.

Příklad 13:

$$\begin{aligned} N &= \{S, V\} \\ \Sigma &= \{p,), (, \Rightarrow, !\} \\ P &= \{S \rightarrow V|(S \Rightarrow S)|!S, V \Rightarrow pV|p\} \end{aligned}$$

$L(G)$ je jazyk všech výrokových formulí.

8.4. Gramatika nezkracující

Gramatika G se nazývá nezkracující, pokud má pouze nezkracující pravidla a může mít pravidlo ve tvaru $S \rightarrow \varepsilon$, ale S se nenachází na žádné z pravých stran.

Příklad 14:

$$\begin{aligned} N &= \{S, A, B, C\} \\ \Sigma &= \{a, b, c\} \\ P &= \{S \rightarrow \varepsilon|abc|Ac, A \rightarrow aBcb, Bcb \rightarrow bBc, Bcc \rightarrow Ccc, bc \rightarrow Cb, aC \rightarrow aab|aA\} \end{aligned}$$

Věta 6: Gramatiky typu 1(8.3.) a 3(8.3.) jsou nezkracující.

Věta 7: Ke každé gramatice G, existuje ekvivalentní gramatika $G^{'}$, ve které jsou všechna pravidla obsahující terminální symboly ve tvaru $A \rightarrow a$, kde $A \in N, a \in \Sigma$.

Důkaz 2: Pro každý terminál $a \in \Sigma$, zavedeme terminál N_a a pravidlo $N_a \rightarrow a$. Všechny výskyty terminálů ve výchozích pravidlech nahradíme příslušnými pomocnými neterminály.

$$Bcb \rightarrow bBc \text{ se změní na } BN_cN_b \rightarrow N_bBN_c, N_c \rightarrow c, N_b \rightarrow b$$

Věta 8: Ke každé nezkracující gramatice existuje ekvivalentní gramatika, která je kontextově *závislá*.

Důkaz 3: Předpokládejme, že $G = \langle N, \Sigma, P, S \rangle$ je nezkracující gramatika. Dle věty 7 můžeme předpokládat, že všechna pravidla jsou buď ve tvaru $A \rightarrow a$ (nevadí) nebo ve tvaru obecně.

$A_1A_2 \cdots A_m \to B_1B_2 \cdots B_n$, kde $A_1, \ldots, A_m, B_1, \ldots, B_n \in N$ a navíc $m \leq n$. Tj. taková pravidla lze psát ve tvaru $A_1A_2 \cdots A_m \to B_1B_2 \cdots B_{my}$, kde $y = B_{m+1} \cdots B_n$ Budeme uvažovat nové pomocné neterminály $X_1, ..., X_m$ [7]. A zavedeme následující pravidla:

$$A_1A_2 \cdots A_m \to X_1A_2 \cdots A_m$$
$$X_1A_2 \cdots A_m \to X_1X_2A_3 \cdots A_m$$
$$\vdots$$
$$X_1X_2 \cdots X_{m-1}A_m \to X_1 \cdots X_{m-1}X_{my}$$
$$X_1X_2 \cdots X_{my} \to B_1X_2X_3 \cdots X_{my}$$
$$\vdots$$
$$B_1B_2 \cdots B_{m-1}X_{my} \to B_1B_2 \cdots B_{m-1}B_{my}$$

Tento postup se aplikuje pro všechna pravidla. Hledaná gramatika $G^{'}$ se skládá z Σ, N + všechny pomocné terminály + všechna odvozená pravidla.

8.5. Základní vlastnosti bezkontextových gramatik

- Levé strany pravidel obsahují jedeiný neterminál.
- Odvozování nezávisí na kontextu.

Věta 9: Mějme bezkontextovou gramatiku $G = \langle N, \Sigma, P, S \rangle$ a nechť $X_1 \cdots X_k, \ldots, z$ je P-derivace délky n, kde $X_1, \ldots, X_k \in (N \cup \Sigma)$ a $z \in (N \cup \Sigma)^*$ a potom pro každé $i = 1, \ldots, k$ existuje řetězec $z_i \in (N \cup \Sigma)^*$ a P-derivace $X_i, \ldots, z_i$ délky n_i tak, že $z = z_1, z_2, \ldots, z_k$ a $n = n_1 + n_2 + \cdots + n_k$

Důkaz 4: Tvrzení prokážeme indukcí přes délku výchozí derivace $X_1 \cdots X_k, \ldots, z$. Pro $n = 0$: Triviální $z = X_1 \cdots X_k, z_i = X_i, n_i = 0$. Každé X_i je derivace délky 0. Nechť tvrzení platí pro libovolnou derivaci délky n a dokážeme, že $X_1 \cdots X_k$ je P-derivace délky $n{+}1$. Jelikož má uvažovaná P-derivace délku n + 1, lze ji psát ve tvaru:

$$X_1 \cdots X_k, \ldots, y^8, z$$

Máme $y \Rightarrow_G z$. Můžeme aplikovat indukční předpoklad: Existují řetězce $y_1, \ldots, y_k$ a P-derivace $X_1, \ldots, y_1$ až $X_k, \ldots, y_k$ délek $n_1 \cdots n_k$ tak, že $y = y_1y_2 \cdots y_k$ a $n = n_1 + n_2 + \cdots + nk$. Z faktu, že $y \Rightarrow_G z$ a z toho, že gramatika je bezkontextová plyne, že y je ve tvaru $y = y^{''}y^{'}Aw^{'}w^{''}$ pro $i = 1, \ldots, k$. Pak z je ve tvaru $z = y^{''}y^{'}uw^{'}w^{''}$ a $A \to n \in P$, to jest $X_i, \ldots, y_i, y^{'}uw^{'}$ je P-derivace délky n_{i+1}. Hledané derivace jsou:

$$X_1, \ldots, y_1$$
$$\vdots$$
$$X_{i-1}, \ldots, y_{i-1}$$
$$X_i, \ldots, y_iy^{'}uw^{'}$$
$$X_{1+1}, \ldots, y_{i+1}$$
$$X_k, \ldots, j_k$$

[7] pro každé pravidlo se uvažují zvlášť

[8] $X_1 \cdots X_k, \ldots$, y má délku n

Příklad 15:

$$\begin{aligned} N &= \{S\} \\ \Sigma &= \{a, b\} \\ P &= \{S \to SS|aS|bSa|\varepsilon\} \end{aligned}$$

Posloupnost: $SbSaS, SbSa, SbaSba, aSbbaSba, abbaSba$ je P-derivace délky 4.
Hledáme P-derivace:

1. S, aSb, ab (délka 2)
2. b (délka 0)
3. S, aSb (délka 1)
4. a (délka 0)
5. S, ε (délka 1)

Příklad 16: Gramatika s jediným pravidlem $aBc \to abc$

Poznámka 5: U regulárních a kontextových gramatik lze hned vidět, jestli $\varepsilon \in L(G)$.

Pro bezkontextovou gramatiku $G = \langle N, \Sigma, P, S\rangle$ zavedeme následující podmnožiny

$$E_0 = \{A \in N | A \to \varepsilon \in P\}$$
$$E_{i+1} = E_i \cup \{A \in N | A \to x, \text{ kde } x \in E_i^*\}$$

Příklad 17:

$$\begin{aligned} A &\to \varepsilon \\ B &\to \varepsilon \\ E_0 &= \{A, B\} \\ E_1 &= \{A, B, F\} \\ E_2 &= \{A, B, F, G\} \end{aligned}$$

$$E_i \subseteq N, E_N = \bigcup_{i=0}^{\infty} E_i$$

Jelikož je N konečná, musí platit:

$$E_0 \subseteq E_1 \subseteq E_2 \subseteq \cdots \subseteq E_i = E_{i+1} = E_{i+2}$$
$$E_N = E_i$$

Věta 10: Pro každou bezkontextovou gramatiku $G = \langle N, \Sigma, P, S\rangle$ a pro příslušné E_N platí následující $A \Rightarrow_G^* \varepsilon$, pak $A \in E_N$. Speciálně $\varepsilon \in L(G)$, pak $S \in E_N$.

Důkaz 5: Prokážeme obě implikace:
Pokud $A \Rightarrow_G^* \varepsilon$, pak prokážeme indukci přes délku P-derivace, tj. triviální případ je $A \Rightarrow_G \varepsilon$, tj. existuje pravidlo $A \to \varepsilon \in P$ tj. $A \in E_0$. Předpokládejme, že tvrzení platí pro všechny P-derivace délky n. Mějme $A, \ldots, \varepsilon$ P-derivace délky n + 1. Použitím předchozí věty $(A, X_1 \cdots X_k, \ldots, \varepsilon)$ $A, X_i \cdots X_n, \ldots, \varepsilon$. Tzn. existují derivace $X_i, \ldots, \varepsilon$ délek nejvýše n. Z předpokladu $X_i \in E_n$, pro každé i tj. i $A \in E_N$. $\Leftarrow$ Dokáže, že pro každé E_i platí, pokud $E \in E_i$ pak $A \Rightarrow_G^* \varepsilon$. Pro E_0 zřejmé. $A \to X_0 \cdots X_k, A \in E_j$.

Věta 11: Pro každou bezkontextovou gramatiku G, existuje bezkontextová gramatika $G^{'}$ neobsahující ε pravidla tak, že $L(G) \setminus \{\varepsilon\} = L(G^{'})$.

Důkaz 6: $G = \langle N, \Sigma, P, S\rangle$ - výchozí gramatika.
Stanovíme množinu E_n dle předchozího postupu $G^{'} = \langle N, \Sigma, P^{'}, S\rangle$. $P^{'} = \{A \to y | A \to x \in P$ a $y \in D_{(x)}\}$, kde $D_{(x)}$ značí množinu řetězeců, které jsou neprázné a vznikly z řetězce x vynecháním libovolného množství neterminálů z E_N.

Příklad 18:

$$\begin{array}{rcl} E_n & = & \{A, B\} \\ X & \to & aAbAB \\ & \dots & \\ X & \to & aAbAB \\ X & \to & abAB \\ X & \to & aAbB \\ X & \to & aAbB \\ X & \to & aAbA \\ X & \to & abB \\ X & \to & abA \\ X & \to & aAb \\ X & \to & ab \end{array}$$

Věta 12: Pro každou bezkontextovou gramatiku existuje ekvivalentní bezkontextová gramatika, která je navíc kontextová (a tudíž nezkracující)

Důkaz 7: Vstupní gramatika G. Dle předchozí věty existuje $G^{'}$ tak, že $L(G) \setminus \{\varepsilon\} = L(G^{'})$. $G^{'}$ je nezkracující a kontextová, protože nemá ε pravidla. Pokud ε nepatří do $L(G)$, pak jsme hotovi. Pokud $\varepsilon \in L(G)$. Pak $G^{'}$ rozšíříme tak, že přidáme startovní symbol $S^{'}$ a pravidlo $S^{'} \to \varepsilon$ a $S^{'} \to S$.

dopsat jednu stránku

9. Automaty

Gramatiky x automaty

generativní formalismus

Automaty - analytické formalismy

Konečné automaty: neformální výpočetní formalismus „jednoduchý počítač“ omezená paměť vstup: řetězec nad vstupní abecedou Σ. Řídící jednotka. Skládá se z konečně mnoha stavů. **Počátek činnosti:** Vstup = celý vstupní řetězec. Řídící jednotka je v počátečním (iniciálním) stavu.
Činnost automatu: Na základě prvního symbolu na vstupu a na základě aktuálního stavu se řídící jednotka přepne do jiného stavu a odebere vstupní symbol.
Konec činnosti: Byl přečten celý vstupní řetězec. Podle toho v jaké končí automat stavu říkáme, že buď přijímá nebo zamítá vstupní řetězec. Některé stavy jsou označené jako přijímací.

Příklad 19: sešit - automat (obr. 4.1)

Formalizace: Konečný deterministický automat (s úplnou přechodovou funkcí) (nad vstupní abecedou Σ) je struktura:
$\langle \Sigma, Q, d, q_0 \rangle$
$\Sigma \ldots$ vstupní abeceda
$Q \ldots$ konečná množina stavu, která je neprázdná
$q_0 \in Q \ldots$ počáteční stav
$F \subseteq Q \ldots$ množina koncových stavů (přijímacích)
δ je zobrazení $\delta : Qx\sigma \to Q$
$\delta(r, a) = q$ čteme: automat A při vstupním symbolu $A \in \Sigma$ a aktuálním stavu $r \in Q$ přejde do stavu $q \in Q$

Pozn.: Q je konečná $\delta \ldots$ zobrazení

Definice 7: Za *determinismus* považujme takovou konfiguraci, pro kterou platí, že je v každém jejím kroku jasné, co bude následovat. Naopak u *nedeterministických* konfigurací není v určitých případech možné další krok přesně vyjádřit na základě znalostí aktuálního kroku.

Příklad 20: – obrázek –

Vstupní řetězce: *abba* (nepřijat), *baba* (nepřijat), *baab* (přijat), *bbaa* (přijat).

V případě řetězce *baab* máme dokonce 3 možnosti výpočtu:

1. $\langle q_0, baab \rangle, \langle q_0, aab \rangle, \langle q_0, ab \rangle, \langle q_0, b \rangle, \langle q_0, \varepsilon \rangle$ – končí neúspěchem.
2. $\langle q_0, baab \rangle, \langle q_0, aab \rangle, \langle q_1, ab \rangle, \langle q_2, b \rangle$ – končí neúspěchem.
3. $\langle q_0, baab \rangle, \langle q_0, aab \rangle, \langle q_0, ab \rangle, \langle q_1, b \rangle, \langle q_2, \varepsilon \rangle$ – končí úspěchem.

Předchozí zápisy můžeme pojmenovat také jako „nedeterministický výpočet."

Jiným zápisem téhož může být také ten následující.

$$\langle \{q_0\}, baab \rangle, \langle \{q_0\}, aab \rangle, \langle \{q_0, q_1\}, ab \rangle, \langle \{q_0, q_1, q_2\}, b \rangle, \langle \{q_0, q_2\}, \varepsilon + \rangle$$

Definice 8: Strukturu $A = \langle \Sigma, Q, \delta, I, F \rangle$ nazvěme **konečným nedeterministickým automatem** nad abecedou Σ. Pro tuto strukturu následně platí tato tvrzení:

- Σ, Q a F jsou stejné jako u konečného deterministického automatu.
- I označuje množinu počátečních stavů, která by měla být obecně neprázdná.
- δ označuje přechodovou funkci ve tvaru $\delta : Q \times \Sigma \to 2^Q$, tedy $\delta(q, a) = \{r_1, \ldots, r_k\}$. Totéž slovně: „Automat může při stavu q při symbolu a přejít do kteréhokoliv stavu z $\{r_1, \ldots, r_k\}$."

Příklad 21:

$$\begin{aligned} \Sigma &= \{a, b\} \\ P &= \{q_0, q_1, q_2, q_3\} \\ I &= \{q_0, q_3\} \\ F &= \{q_2\} \end{aligned}$$

Následně přechodová funkce:

$$\delta = \quad \{\langle q_0, a, \{q_0, q_1\} \rangle, \langle q_0, b, \{q_0\} \rangle, \langle q_1, a, \{q_2\} \rangle, \langle q_1, b, \{q_2\} \rangle, \\ \langle q_2, a, \emptyset \rangle, \langle q_2, b, \emptyset \rangle, \langle q_3, a, \emptyset \rangle, \langle q_3, b, \emptyset \rangle\}$$

9.1. Reprezentace KNA

Předchozí příklad číslo 21 lze reprezentovat několika způsoby:

1. Přechodová tabulka, která ve svém těle obsahuje množiny stavů.

	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2*	$\emptyset$	$\emptyset$
$\rightarrow q_3$	$\emptyset$	$\{q_1\}$

2. Diagram, který automat demonstruje v grafičtější podobě.
   – obrazek –

9.2. Nedeterministický výpočet

Nyní si popišme **nedeterministický výpočet**, který je definován následujícími věcmi:

- Konfigurace, což je dvojice ve tvaru $\langle \mathit{stav}, \mathit{řetězec} \rangle$.
- Počáteční konfigurace ve tvaru $\langle q, w \rangle$ kde $q \in I$.
- Koncová konfigurace ve tvaru $\langle q, \varepsilon \rangle$.
- Koncová přijímací konfigurace $\langle q, \varepsilon \rangle$ kde $q \in F$.

Definice 9: Mějme $A = \langle \Sigma, Q, \delta, I, F \rangle$ a $w \in \Sigma^*$. Pak posloupnost konfigurací $\langle r_i, w_i \rangle$ pro $i = \{0, \ldots, n\}$ splňující podmínky

$$R_0 \in I \tag{1}$$
$$w_0 = w \tag{2}$$
$$w_n = \varepsilon \tag{3}$$
$$w_i = a_i w_{i+1} \text{ a } r_{i+1} \in \delta(r_i, a_i) \text{ pro } i = \{0, \ldots, n-1\} \tag{4}$$

nazveme **nedeterministický výpočet**.

9.3. Rozšířená přechodová funkce

Definice 10: Rozšířená přechodová funkce ...