UNIVERZITA PALACKÉHO V OLOMOUCI
KATEDRA INFORMATIKY

M. Rotter, T. Kukučka, J. Zehnula

# KMI/FJAA – Formální jazyky a automaty

26. 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

# 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 ⟩. 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 Chomsky 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 $\varepsilon$, přičemž platí následující zápis:
$$\varepsilon \subseteq \emptyset \rightarrow \Sigma$$
Prázdný řetězec **není** symbolem, tedy $\varepsilon \notin \Sigma$.

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

**Poznámka** 1: Uveďme si rovněž značení pro dva důležíté pojmy:

- $\Sigma^*$ označuje množinu všech řetězců nad abecedou$\Sigma$.
- $\Sigma^+$ označuje množinu všech řetězců nad abecedou$\Sigma$ vyjma $\varepsilon$.

## 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:
$$\begin{aligned} x &: \{1, \dots, n\} \rightarrow \Sigma \\ y &: \{1, \dots, m\} \rightarrow \Sigma \\ x \circ y &: \{1, \dots, nm\} \rightarrow \Sigma \end{aligned}$$
  **Poznámka** 2: 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 $Sfx(x) = \{y | \exists z \text{ tak, že } zy = x\}$.

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

**Věta** 2:

$$xy = xz \implies y = z$$
$$yx = zx \implies y = z$$

Algebraicky je operace zapsána jako $\langle \Sigma^*, \cdot, \varepsilon \rangle$.

**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** 3: 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:

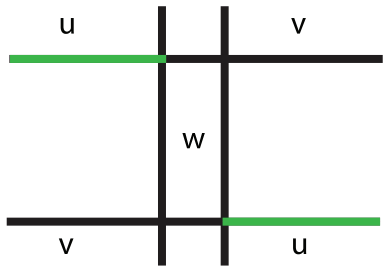
$$uv = z^p z^q = z^{p+q} = z^p z^q = vu$$

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 všech řetězců* $\Sigma^*$. Označme tento jazyk jako $L$. Pak platí tato tvrzení:

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

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

## 6. Lexikografické uspořádání

Předpokládejme uspořádání na množině $\Sigma^*$. 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 $\Sigma^*$ 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:

$$L_1L_2 = \{xy|x \in L_1, y \in L_2\}$$

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

- **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**[5] 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.

$x \to y$, kde symbol $\to \notin \Sigma$ 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 vstupní řetězec určitě nevznikne řetězec kratší, než-li jeho předloha.

- $\varepsilon$ - pravidlo je pravidlo tvaru $x \to \varepsilon$.

[5] Stephen Cole Kleene je známý matematik, jenž se významně podílel na položení základů teoretických počíta-čových věd.

### 8.1.2. Příklady pravidel

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

$$\begin{aligned} aa &\to bc \\ bb &\to y \\ c &\to \varepsilon \end{aligned}$$

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

$$\begin{aligned} expr &\to expr + expr \\ expr &\to expr \times expr \end{aligned}$$

### 8.1.3. Přímé odvozování řetězců pomocí pravidel

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

$$\begin{aligned} u &= pxq \\ v &= pyq \end{aligned}$$

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 $\Sigma^*$ 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:

$$\begin{aligned} baaa &\to bbca \\ bac &\to bcabc \end{aligned}$$

**Definice** 1: Definujme pojem **derivace**. Jedná se o posloupnost řetězců ve 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:

- $\Sigma$ - abeceda terminálních symbolů (tyto symboly tvoří řetězce daného jazyka).
- $N$ - 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, $\Sigma$ je abecedou terminálních symbolů, $P$ je množinou odvozovacích pravidel a $S \in N$ je tzv. *počátečním* resp. *startovním* neterminálem. Pak tuto čtveřici nazveme **gramatikou**.

**Poznámka** 4: Pokud chceme vyjádřit, že z jednoho symbolu odvozujeme 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 vypadat třeba takto:

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

**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:

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

Vidíme tedy, že takový jazyk obsahuje *věty*, které lze odvodit ze startovacího neterminálu pomocí pravidel této gramatiky.

**Příklad** 9: Tento příklad čerpá gramatiku z příkladu 1.

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

**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 $pAq \to p \times q,$ 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 $S \to \varepsilon$, 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:
  $$A \to x, \text{ kde } A \in N, x \in (\Sigma \cup N)^*$$
  Na levých stranách pravidel tedy očekáváme pouze neterminální symbol.

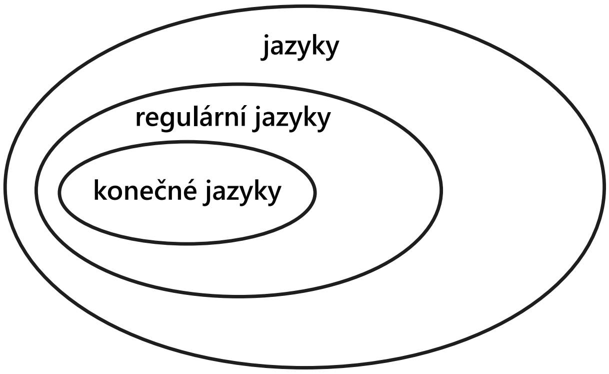
  **Příklad** 10: Mějme tuto gramatiku:
  $$\begin{aligned} G &= \langle N, \Sigma, P, S \rangle \\ N &= \{A, S\} \\ \Sigma &= \{0, 1\} \\ P &= \{S \to 0A, A \to \varepsilon\} \end{aligned}$$
- **Gramatiky typu 3** – jedná se o tzv. *regulární* resp. *pravolineární* gramatiky, které obsahují pravidla ve třech následujících tvarech:
  1. $A \to bB,$ kde $A, B \in N, b \in \Sigma$
  2. $A \to a$
  3. $S \to \varepsilon$

**Poznámka** 5: 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 $\Sigma$ a $S$ a zvolme $N$. Následně platí $\forall x_i \in L$ je dvojího typu:

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




Obrázek 2. Vychodilovo „vajíčko.“

**Příklad** 11:

$$\begin{aligned} N &= \{S\} \\ \Sigma &= \{a,b\} \\ P &= \{S \to aSb|\varepsilon\} \\ L(G) &= \{a^n b^n | n \leq 0\} \end{aligned}$$

Máme tedy *bezkontextový* jazyk.

**Příklad** 12:

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

$L(G)$ je *bezkontextový* jazyk.

**Příklad** 13:

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

$L(G)$ 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 $S \to \varepsilon$, přičemž $S$ se nenachází na žádné z pravých stran.

**Příklad** 14:

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

**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. $A_1A_2 \cdots A_m \rightarrow B_1B_2 \cdots B_n$, 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 $A_1A_2 \cdots A_m \rightarrow B_1B_2 \cdots B_{my}$, kde $y = B_{m+1} \cdots B_n$ Budeme uvažovat nové pomocné neterminály $X_1, ..., X_m$ [6]. A zavedeme následující pravidla:

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

Tento postup se aplikuje pro všechna pravidla. Hledaná gramatika $G^{'}$ se skládá z $\Sigma, N$ + 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:

$$X_1 \cdots X_k, \ldots, y^7, z$$

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

[6] pro každé pravidlo se uvažují zvlášť

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

$i = 1, \dots, k$. Pak $z$ je ve tvaru $z = y'' y' uw' w''$ a $A \to n \in P$, to jest $X_i, \dots, y_i, y' uw'$ je P-derivace délky $n_{i+1}$. Hledané derivace jsou:

$$\begin{gathered} X_1, \dots, y_1 \\ \vdots \\ X_{i-1}, \dots, y_{i-1} \\ X_i, \dots, y_i y' uw' \\ X_{1+1}, \dots, y_{i+1} \\ X_k, \dots, j_k \end{gathered}$$

**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. $S, \varepsilon$ (délka 1)

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

**Poznámka** 6: 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

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

**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 nasledující $A \Rightarrow_G^* \varepsilon$, 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 $\varepsilon$ 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:

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

**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 $\varepsilon$ pravidla. Pokud $\varepsilon$ 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.

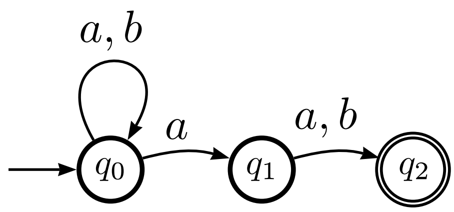
**Příklad** 19: sešit - automat (obr. 4.1)

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)
$\delta$ 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:




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 $\Sigma$. Pro tuto strukturu následně platí tato tvrzení:

- $\Sigma, Q$ a $F$ jsou stejné jako u konečného deterministického automatu.
- $I$ označuje množinu počátečních stavů, která by měla být obecně neprázdná.
- $\delta$ 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:

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

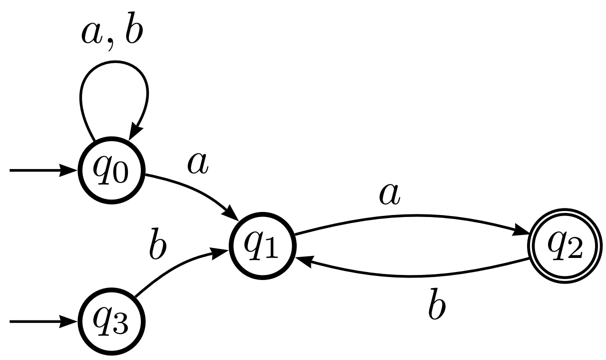
## 9.1. Reprezentace KNA

Předchozí příklad číslo 21 lze reprezentovat několika způsoby:

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

| | a | b |
|---|---|---|
| $\to q_0$ | $\{q_0, q_1\}$ | $\{q_0\}$ |
| $q_1$ | $\{q_2\}$ | $\{q_2\}$ |
| $q_2*$ | $\emptyset$ | $\emptyset$ |
| $\to q_3$ | $\emptyset$ | $\{q_1\}$ |

2. **Diagram**, který automat demonstruje v grafičtější podobě.



## 9.2. Nedeterministický výpočet

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

- Konfigurace, což je dvojice ve tvaru $\langle$*stav, 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 má tvar

$$A = \langle \Sigma, Q, \delta, I, F\rangle$$

navíc zavádíme:

$$\delta^* : \Sigma^Q \times \Sigma^* \to \Sigma^Q$$
$$\delta^*(R, w) = \left\{ \begin{array}{ll} R & \text{pokud } w = \varepsilon \\ \delta^*(\bigcup\limits_{q \in R} \delta(q, w), u) & \text{pokud } w = au \text{, kde } a \in \Sigma, u \in \Sigma^q \end{array} \right\}$$

**Věta** 13: Platí $\delta^*(R, w) = \delta^*(\delta^*(R, u), v), \forall R \subseteq Q, uv \in \Sigma^*$.

**Důkaz** 8: Předchozí tvrzení dokazujeme indukcí přes délku řetězce.

1. Pro $u = \varepsilon$ je situace triviální.
2. Pokud $u = ay, |y| < |u|,$ pak $\delta^*(R, w) = \delta^*(R, ayv) = \delta^*(R, a(yv))$.
3. Nyní aplikujme definici.

$$\delta^*(\bigcup_{q \in R} \delta(q, a), yv) = \text{ indukční předpoklad}$$
$$\delta^*(\delta^*(\bigcup_{q \in R} \delta(q, a), y), v) = \text{ definicie } \delta^*$$
$$\delta^*(\delta^*(R, ay), v) = \delta^*(\delta^*(R, u), v)$$

**Věta** 14: Platí následující tvrzení:

$$\delta^*(\bigcup_{i=1}^{k} R_i, w) = \bigcup_{i=1}^{k} \delta^*(R_i, w) \text{ pro každé} R_i \subseteq Q, w \in \Sigma^*$$

**Důkaz** 9: Předchozí tvrzení dokazujeme indukcí přes délku řetězce w.

$$\delta^*(\bigcup_{i=1}^{k} R_i, w) = \delta^*(\bigcup_{i=1}^{k} R_i, au) = \delta^*(q \in \bigcup_{\bigcup_{i=1}^{k}} \delta(q,a), u)$$

$$\delta^*(\bigcup_{i=1}^{k} \bigcup_{q\in R_i} \delta(q,a), u) \ldots \text{indukční předpoklad}$$

$$\bigcup_{i=1}^{k} \delta^*( \bigcup_{q\in R_i} \delta(q,a), u)$$

$$\bigcup_{i=1}^{k} \delta^*(R, a_n) = \bigcup_{q\in R_i} \delta(R, w)$$

## 9.4. Řetězce přijímané KNA

KNA $A$ přijímá řetězec $w$, pokud $\delta^*(I,w) \cap F \neq \emptyset$. Navíc jazyk, přijímaný KNA $A$ si definujme jako $L(A) = \{w \in \Sigma^* | \delta^*(I,w) \cap F \neq \emptyset\}$.

**Věta** 15: Platí, že $w \in L(a)$ právě tehdy, když KNA $A$ má přijímací výpočet pro $w$.

**Důkaz** 10: Předchozí tvrzení lze dokázat indukcí přes délku řetězce w.

$$q \in \delta^*(I,w) \text{ právě tehdy, když existuje výpočet pro w, končící ve stavu } q$$
$$\text{dekompozice navíc } w = ua$$

## 9.5. Determinizace KNA

**Věta** 16: Pro každý KDA $A = \langle \Sigma, Q, \delta, q_0, F\rangle \exists$ KNA $A'$ tak, že $L(A) = L(A')$.

**Důkaz** 11: Pro výchozí $A$ uvažujme $A' = \langle\langle \Sigma, Q, \delta', q_0, F\rangle$, pak $\delta'(q,a) = \{\delta(q,a)\}$. Zbytek důkazu je zřejmý.

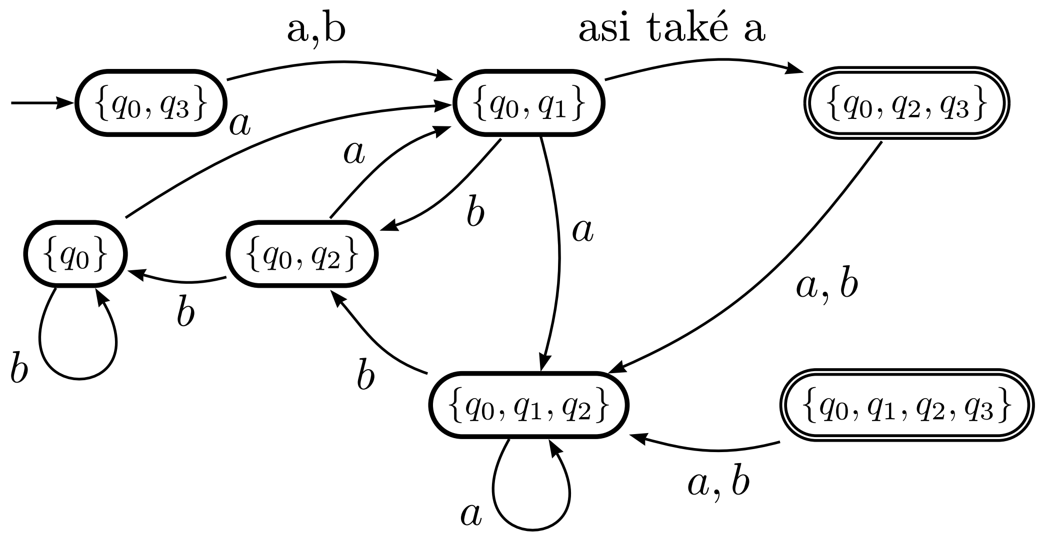
**Věta** 17: Pro každý KNA $A$ existuje KDA $A^D$ tak, že $L(A) = L(A^D)$.

**Důkaz** 12: Předchozí větu lze dokázat následujícím způsobem:

1. Uvažujme $A^D = \langle \Sigma, 2^Q, \delta^D, I, F^D\rangle$, kde $F^D = \{R \subseteq Q | R \cap F \neq \emptyset\}, \delta^D(R,a) = \delta^*(R,a)$. Nyní zbývá ukázat, že $\delta^*(I,w) \cap F \neq \emptyset$ právě tehdy, když $(\delta D)^*(I,w) \in F^D$, což prokážeme indukcí přes délku řetězce $w$.
2. Pro $w = \varepsilon$ je situace zřejmá. Jinak $(\delta D)^*(R,w) = (\delta D)^*(R,\varepsilon) = R = \delta^*(R,\varepsilon) = \delta^*(R,w)$.
3. Předpokládejme, že tvrzení platí pro řetězce délky $n$ a nechť $w$ má délku $n+1$ a $w = au$ pro $a \in \Sigma, |u| < |v|$. Pak:

$$(\delta^D)^*(R,w) = (\delta^D)^*(R,au) = (\delta^D)^*(\delta^D(R,a),u) =$$
$$= \delta^*(\delta^D(R,a)u) = \delta^*(\delta^*(R,a),u) = \delta^*(R,au) = \delta^*(R,w)$$

**Příklad** 22: Vemme KNA z příkladu 21.




Ještě jeden automat, nepřečtu to dobře ze sešitu. :)

## 9.6. Algoritmus pro převod KNA na KDA

Nyní si ukažme pseudokód algoritmu pro převod konečných nedeterministických automatů na konečné deterministické automaty, pro které platí, že akceptují řetězece stejného jazyka.

```
δˆD ← ∅; QˆD ← ∅; FˆD ← ∅; w ← 1
while w ≠ Q do
     select R ∈ w
      w ← w ∖ R; QˆD ← QˆD ∪ R
     if R ∩ F ≠ ∅ then
          FˆD ← FˆD ∪ R
     endif
     foreach a ∈ Σ do
          v ← δˆ∗(R, a)
          if N ≠ ∅ then
               if N ∉ w ∪ QˆD then
                    w ← w ∪ N
               endif
               δˆD ← δD ∪ < R, u, N >
          endif
     end
end
return < Σ, QˆD, δˆD, I, FˆD >
```

Obrázek 3. Pseudokód pro převod KNA na KDA.

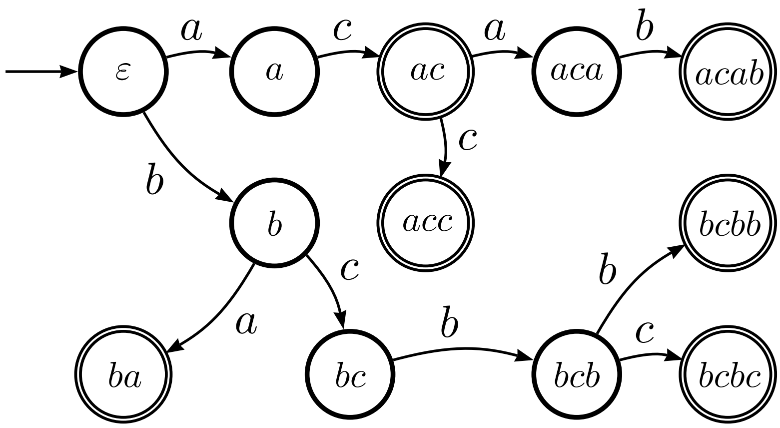
**Definice** 11: *Trie* je prefixový strom, který umožňuje „rychlé hledání ve slovníku.“

**Definice** 12: Jako *slovník* označujeme konečný neprázdný jazyk který neobsahuje $\varepsilon$.

**Definice** 13: Trie slovníku $L$ je KDA $T_L = \langle \Sigma, Q, \delta, \varepsilon, F \rangle$, přičemž:

$$\begin{aligned} Q &\ - \ \text{množina prefixů všech řetězců} \\ Q &\ - \ \bigcup_{w \in L} Pfx(w) \\ F &= L^{w \in L} \\ \delta(w, a) &= wa, \ \text{pokud } wa \in Q, \ \text{jinak } \delta(w, a) \text{ není definováno} \end{aligned}$$

**Příklad** 23: Uveďme si příklad konečného slovníkového automatu $D_L$, který je pochopitelně deterministický:




# 10. Vztah regulárních jazyků a konečných automatů

## 10.1. Regulární jazky jsou rozpoznatelné KDA (implikace zleva)

**Věta** 18: Pro každou regulární gramatiku $G = \langle N, \Sigma, P, S \rangle$ existuje konečný deterministický automat $A$ tak, že jazyk generovaný gramatikou je totéž, jako jazyk rozpoznatelný automatem, tj. $L(G) = L(A)$

**Důkaz** 13: Nejprve uvažujeme situaci, že $\varepsilon \notin L(G)$. Uvažujme konečný nedeterministický automat $A = \langle \Sigma, N \cup \{\#\}, \delta, \{S\}, \{\#\} \rangle$.

$$\delta(A, b) = \begin{cases} \{B \in N | A \to bB \in P\} & \text{pokud} \quad A \in N \land A \to b \notin P \\ \{B \in N | A \to bB \in P\} \cup \{\#\} & \text{pokud} \quad A \in N \land A \to b \in P \\ \emptyset & \text{jinak} \end{cases}$$

Pro důkaz $L(G) = L(A)$ stačí prokázat, že pro každé $A \in N$ a $x \in \Sigma^*$ platí, že $A \Rightarrow_G^* x$ právě když $\# \in \delta^*(\{A\}, x)$.

Důkaz provedeme indukcí přes délku řetězce x.

1. Pro $|x| = 1$ zřejmé. $A \Rightarrow_G^* x$ právě když $A \Rightarrow_G x$ , tj. existuje pravidlo $A \to x \in P$ tj. z definice $\delta^*$ platí, že $\# \in \delta^*(\{A\}, x)$. Nechť $|x| = n$ a nechť tvrzení platí pro všechny řetězce

kratší délky. Jelikož gramatika G je regulární, má P-derivace $A, \dots, x$ právě n kroků. Pokud $|x| > 1$ pak $A \Rightarrow_G bB \Rightarrow_G^* by = x$ pro nějaké $A \to bB \in P$.

2. Pro $|y| < n$ z indukčního předpokladu platí, že $\# \in \delta^*(\{B\}, y)$. Tím spíš $\delta^*(\{A\}, x) = \delta^*(\{A\}, by) = \delta^*(\delta(A, b), y) = \delta^*(\{B, \dots\}, y) \supseteq \delta^*(\{B\}, y)$ tj. $\# \in \delta^*(\{A\}, \#)$ protože $A \to bB \in P$tj.$B \in \delta(A, b)$

3. Tím jsme prokázali, že pokud $A \Rightarrow_G^* x$ pak $\# \in \delta^*(\{A\}, x)$.

4. Obráceně, pokud $\# \in \delta^*(\{A\}, x)$ pak pro $x = by, b \in \Sigma$ máme: $\# \in \delta^*(\{A\}, by) = \delta^*(\delta(A, b), y) = \delta^*(\bigcup_{B \in \delta(A,b)} \{B\}, y) = \bigcup_{B \in \delta(A,b)} \delta^*(\{B\}, y)$ Tj. existuje $B \in \delta(A, b)$ tak, že $\# \in \delta^*(\{B\}, y)$. Ze zavedení $\delta$ plyne, že $A \to bB \in P$

5. Aplikací indukčního předpokladu, existuje P-derivace $B, \dots, y$. Hledaná P-derivace je ve tvaru: $A, bB, \dots, by = x$,tj. $A \Rightarrow_G^* x$.

V případě, že $\varepsilon \in L(G)$, rozšíříme automat následovně, jednou ze tří možností:

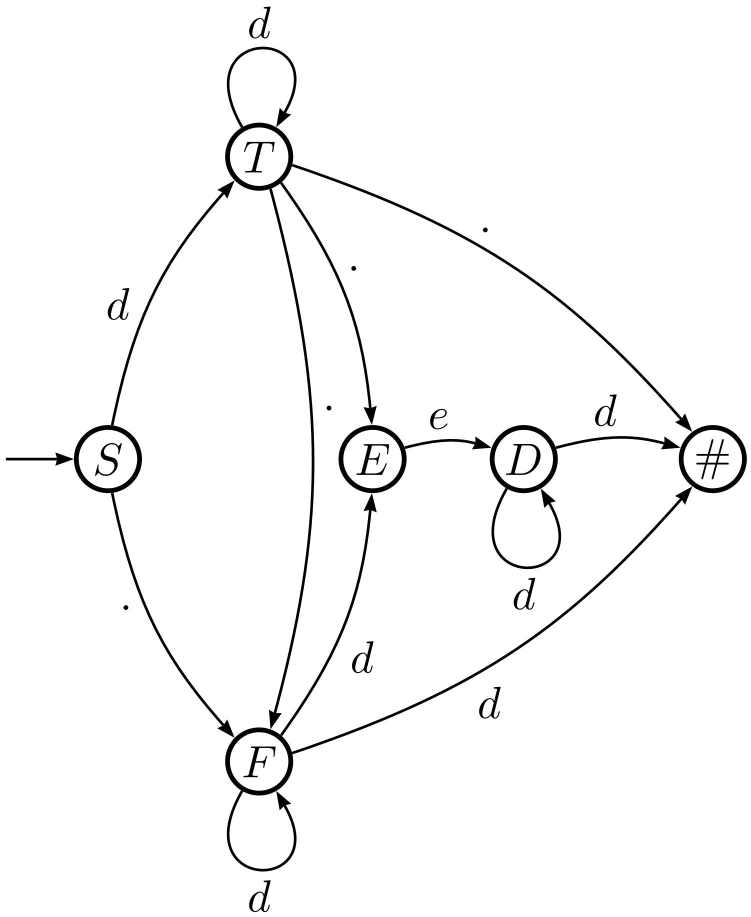
1. Přidáme $S$ do množiny koncových stavů.
2. Přidáme # mezi počáteční stavy
3. Zavedeme nový stav, který bude počáteční a zároveň koncový a nevedou z něj žádné přechody jinam.

**Poznámka** 7: Nyní zbývá automat pouze determinizovat.

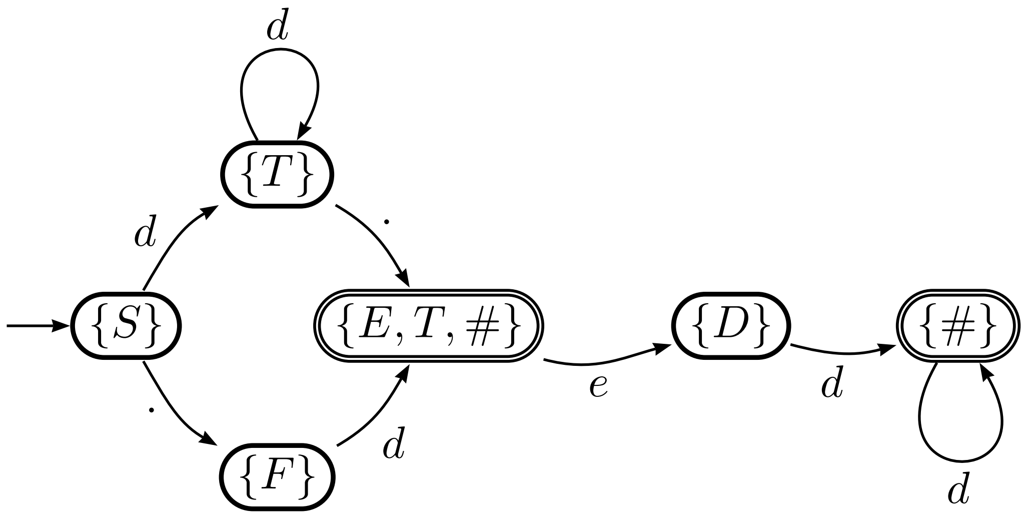
**Příklad** 24: Máme gramatiku $G$.

$$\begin{aligned} G &= \langle N, \Sigma, P, S \rangle \\ \Sigma &= \{e, d, .\} \\ P &= \{S \to .F|dT, T \to .E|.F|dT|., D \to dD|d, E \to eD, F \to dE|dF|d\} \end{aligned}$$

Automat rozpoznávající jazyk, generovaný gramatikou G, bude vypadat následovně:



Když tento automat zdeterminizujeme, dostaneme následující automat:




## 10.2. Jazyky rozpoznatelné KDA jsou regulární (implikace zprava)

**Věta** 19: Pro každý konečný deterministický automat $A = \langle \Sigma, Q, \delta, q_0, F \rangle$ existuje regulární gramatika $G$ tak, že $L(A) = L(G)$.

**Důkaz** 14: Za neterminální symboly $G$ vezmeme stavy automatu. Startovní neterminál bude $q_0$. Uvažujeme gramatiku: $G = \langle Q, \Sigma, P, q_0 \rangle$

$P = \{q \to ar |$ pokud $\delta(q, a) = r,$ pro $q, r \in Q$ a $a \in \Sigma\}$
$\cup \{q \to a |$ pokud $\delta(q, a) \in F\}$

Prokážeme že: $q \Rightarrow_G^* x$ právě když $\delta^*(q, x) \in F$

Pro $|x| = 1$ platí: $q \Rightarrow_G^* x$ právě když existuje pravidlo $q \to x \in P$ , tj. z definice $P$ platí $\delta(q, x) \in F$

Pro $x = by$, kde $b \in \Sigma^*$ předpokládejme, že tvrzení platí pro $y$. Platí, že $q \Rightarrow_G br \Rightarrow_G^* by = x$ právě když $\delta(q, b) = r$ a $\delta^*(r, y) \in F$

To znamená $\delta^*(q, by) = \delta^*(\delta(q, b), y) \in F$

Předchozí dokazuje, že $x \in L(G)$ právě když $x \in L(A)$ pro každý neprázdný $x$.

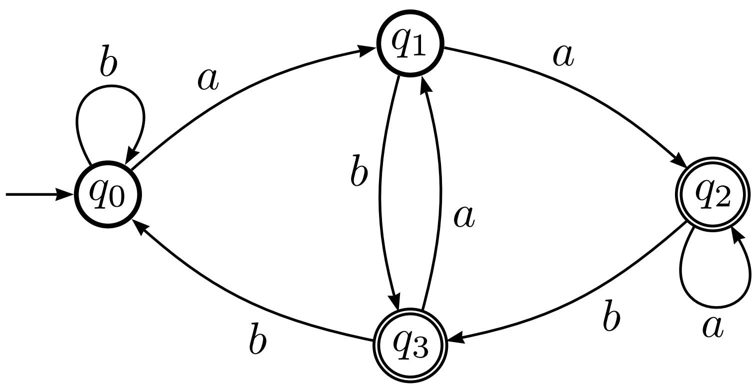
Pokud $A$ nepřijímá $\varepsilon$ , pak jsme hotovi.

Uvažujeme nový neterminál $S$, který bude nový startovní symbol, tj. místo $G$ uvažujeme $G' = \langle Q \cup \{S\}, \Sigma, P', S \rangle$

$P' = \{S \to \varepsilon\} \cup \{S \to x | q_0 \to x \in P\} \cup P$

Pak $L(A) = L(G)$.

**Příklad** 25: Mějme abecedu $\Sigma = \{a, b\}$ a automat zadaný diagramem:




Odvozovací pravidla gramatiky, generujíci tento jazyk budou:

$$\begin{array}{lll} q_0 & \rightarrow & aq_1 \mid bq_0 \\ q_1 & \rightarrow & aq_2 \mid a \mid bq_3 \mid b \\ q_2 & \rightarrow & aq_2 \mid a \mid bq_3 \mid b \\ q_3 & \rightarrow & aq_1 \mid bq_0 \end{array}$$

## 10.3. Regulární gramatiky

Co jsou to regulární gramatiky a jaké podmínky jejich odvozovací pravidla splňují již víme, ale můžeme si je ještě rozdělit na dva druhy, právě podle tvaru odvozovacích pravidel.

1. **Zprava regulární gramatiky:** Obsahují pravidla ve tvaru $A \rightarrow bB$ tj. neterminál na prvé straně je na pravo od terminálního symbolu.
2. **Zleva regulární gramatiky:** Obsahují pravidla ve tvaru $A \rightarrow Bb$. Analogicky se neterminál nachází vlevo od terminálního symbolu.

**Věta** 20: Pro každou zleva regulární gramatiku $G = \langle N, \Sigma, P, S \rangle$ existuje konečný deterministický automat $A$ tak, že *L(A)=L(G)*.

**Důkaz** 15: Budeme konstruovat automat, jehož stavy budou $N$, nový pomocný počáteční stav # a jediný koncový stav je $S$.

Hledaný KNA $A = \langle \Sigma, N \cup \{\#\}, \delta, \{\#\}, \{S\} \rangle$ s následovně definovanou přechodovou funkcí $\delta$

$$\delta(q,a) = \begin{cases} \{A \in N | A \rightarrow a \in P\} & \text{pokud} \quad q = \# \\ \{A \in N | A \rightarrow Ba \in P\} & \text{pokud} \quad q = B \end{cases}$$

Ekvivalence *L(A)=L(G)* se dokazuje vzájemně jednoznačnou korespondencí P-derivace a nedeterministického výpočtu.

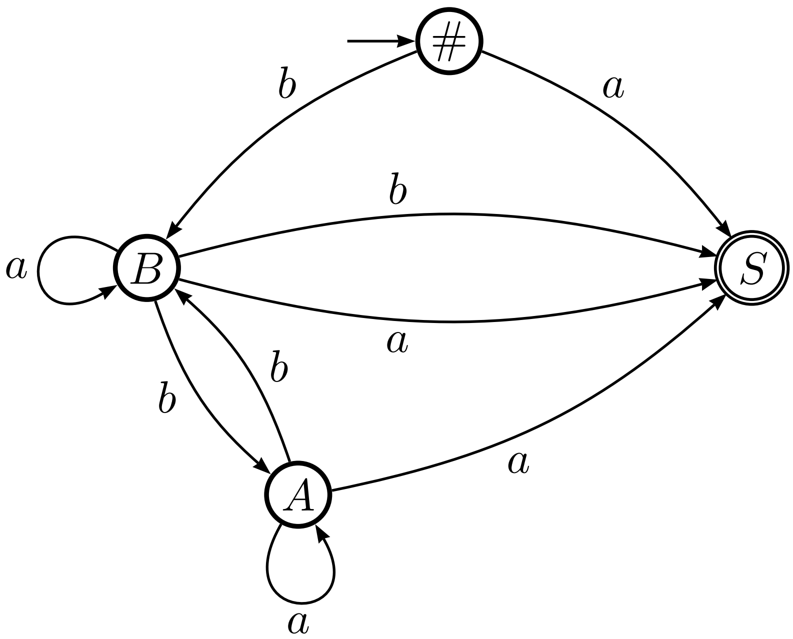
Pro derivaci:

$x_0 = S, x_1, x_2, \ldots, x_{n-1}, x_n = x$
jsme schopni sestavit posloupnost
$\langle \#, X_n \rangle, \langle A_{n-1}, y_{n-1} \rangle, \ldots, \langle A_1, y_1 \rangle, \langle S, \varepsilon \rangle$ kde $x_i = A_i y_i$

**Příklad** 26: Máme gramatiku $G$ s následovně definovanými pravidly.

$$\begin{aligned} S &\rightarrow Aa|Ba|Bb|a \\ A &\rightarrow Aa|Bb \\ B &\rightarrow Ab|Ba|b \end{aligned}$$

Automat rozpoznávající jazyk generovaný touto gramatikou bude vypadat následovně:




**Věta** 21: Pro každý konečný deterministický automat $A$ existuje zleva regulární gramatika taková, že *L(A)=L(G)*

**Důkaz** 16: Neterminály gramatiky jsou stavy automatu a budeme uvažovat dodatečný statovní neterminál S.

$$\begin{aligned} P = \quad & \{ \quad \delta(q, a) \rightarrow qa | q \in Q \wedge a \in \Sigma \} \ \cup \\ & \{ \quad \delta(q_0, a) \rightarrow a | q_0 \text{ je počáteční stav} \} \ \cup \\ & \{ \quad S \rightarrow w | w \text{ je pravá strana každého pravidla } q \rightarrow w, \text{ kde } q \in F \} \end{aligned}$$

**Příklad** 27: Vezmeme KDA z příkladu 25. Odvozovací pravidla budou vypadat takto:

$$\begin{array}{rcl} q_0 & \to & q_0 b \mid b \mid q_3 b \\ q_1 & \to & q_0 a \mid a \mid q_3 a \\ q_2 & \to & q_1 a \mid q_2 a \\ q_3 & \to & q_1 b \mid q_2 b \\ S & \to & q_1 a \mid q_1 b \mid q_2 a \mid q_2 b \end{array}$$

**Definice** 14: Regulární jazyky jsou jazyky, generované zprava (zleva) regulárními gramatikami, tj. jsou rozpoznatelné konečnými ne/deterministickými automaty.

**Poznámka** 8: Pravidla zprava a zleva nelze míchat.