UNIVERZITA PALACKÉHO V OLOMOUCI
KATEDRA INFORMATIKY

M. Rotter, T. Kukučka, J. Zehnula

KMI/FJAA – Formální jazyky a automaty

16. dubna 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 vytváří stromovou závislost jednotlivých tokenů, jejíž reprezentace se nazývá *syntaktický-derivační strom*. V rámci této analýzy rozlišme:

1. Teorii jazyků, jenž se zabývá 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 počítačové 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*

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

  kde

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

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

[1] Jméno této osoby čti [*čomski*] a zapamatuj si ke státnicím, 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 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.

Poznámka 1: Uveďme si rovněž značení pro dva důležité pojmy:

- 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:
$$\begin{aligned} x &: \{1, \dots, n\} \to \Sigma \\ y &: \{1, \dots, m\} \to \Sigma \\ x \circ y &: \{1, \dots, n+m\} \to \Sigma \end{aligned}$$
  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 $v = z^q$.

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

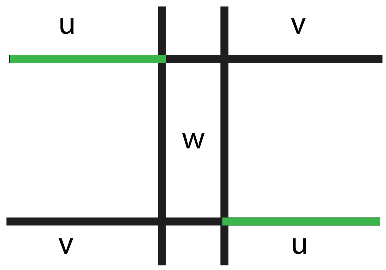
$$uv = z^p z^q = z^{p+q} = z^q z^p = vu$$

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.

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



Obrázek 1. Grafická pomůcka ke komutativitě zřetězení řetězců.

5. Formální jazyk

Zaveďme si pojem *formální jazyk nad množinou všech řetězců* Σ^*. 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 Σ^*. 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:

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

  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:

$$L^n = \begin{cases} \{\varepsilon\} & \text{pro } n = 0 \\ LL^{n-1} & \text{pro } n \geq 1 \end{cases}$$

- **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.

$$x \to y, \text{ kde symbol } \to \notin \Sigma \text{ můžeme prohlásit za tzv. metasymbol.}$$

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

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

$$\begin{aligned} aa &\rightarrow bc \\ bb &\rightarrow abba \\ c &\rightarrow \varepsilon \end{aligned}$$

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

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

8.1.3. Přímé odvozování řetězců pomocí pravidel

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

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

Značení předchozí operace je následující:

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

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

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 $x \rightarrow y \in P$ a $x \Rightarrow_{x \rightarrow y} y$.

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

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

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:

- Σ - 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 $G = \langle N, \Sigma, P, S\rangle$, 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* 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: Gramatika 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 takto:

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

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 8.

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

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 $pAq \to p \times q,$ 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.

  Zároveň platí pro každé pravidlo (s výjimkou pravidla $S \to \varepsilon$), že délka odvozeného řetězce je minimálně stejně velká jako délka vstupního řetězce. Gramatika tedy zároveň obsahuje pouze tzv. *nezkracující* 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 a na pravé straně očekáváme minimálně jeden symbol (s výjimkou pravidla $S \to \varepsilon$). S se navíc nesmí vyskytovat na pravých stranách pravidel.

  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:

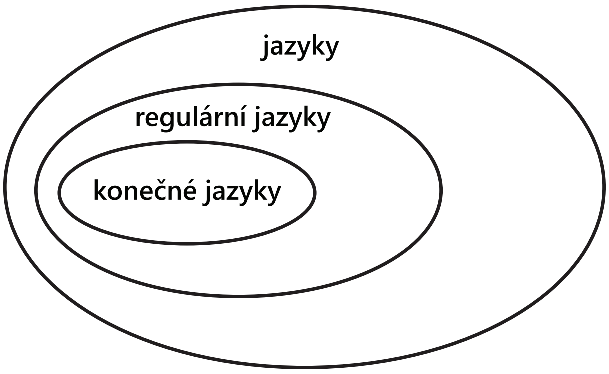
  1. $A \to bB,$ kde $A, B \in N, b \in \Sigma$
  2. $A \to a$
  3. $S \to \varepsilon$

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 Σ 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:

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

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 $A \to a$, kde $A \in N, a \in \Sigma$.

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

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

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 $A \to a$ (nevadí) nebo ve tvaru obecně. $A_1A_2 \cdots A_m \to B_1B_2 \cdots B_n$, kde $A_1, \dots, A_m, B_1, \dots, B_n \in N$ 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 $X_1, ..., X_m$ [6]. A zavedeme následující pravidla:

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

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 $X_1 \cdots X_k, \dots, z$ je P-derivace délky n, kde $X_1, \dots, X_k \in (N \cup \Sigma)$ a $z \in (N \cup \Sigma)^*$ a potom pro každé $i = 1, \dots, k$ existuje řetězec $z_i \in (N \cup \Sigma)^*$ a P-derivace $X_i, \dots, z_i$ délky n_i tak, že $z = z_1, z_2, \dots, z_k$ a $n = n_1 + n_2 + \cdots + n_k$

Důkaz 4: Tvrzení prokážeme indukcí přes délku výchozí derivace $X_1 \cdots X_k, \dots, z$. 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:

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

Máme $y \Rightarrow_G z$. Můžeme aplikovat indukční předpoklad: Existují řetězce $y_1, \dots, y_k$ a P-derivace $X_1, \dots, y_1$ až $X_k, \dots, y_k$ 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] $X_1 \cdots X_k, \dots$, y má délku n

$i = 1, \ldots, k$. 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:

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

Příklad 15:

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

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 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 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:

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

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 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)

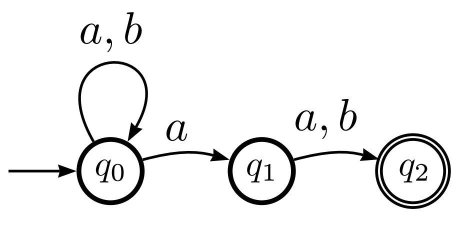
Formalizace: Konečný deterministický automat (s úplnou přechodovou funkcí) (nad vstupní abecedou Σ) je struktura:

$\langle \Sigma, Q, d, q_0 \rangle$
$\Sigma \ldots$ 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:



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:

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

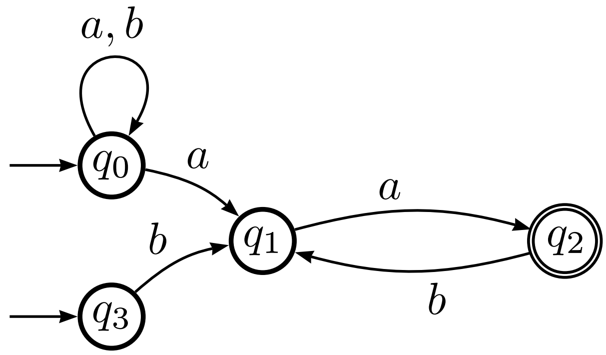
9.1. Reprezentace KNA

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 $\langle \mathit{stav}, \mathit{řetězec} \rangle$.
- 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 má tvar:

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

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.

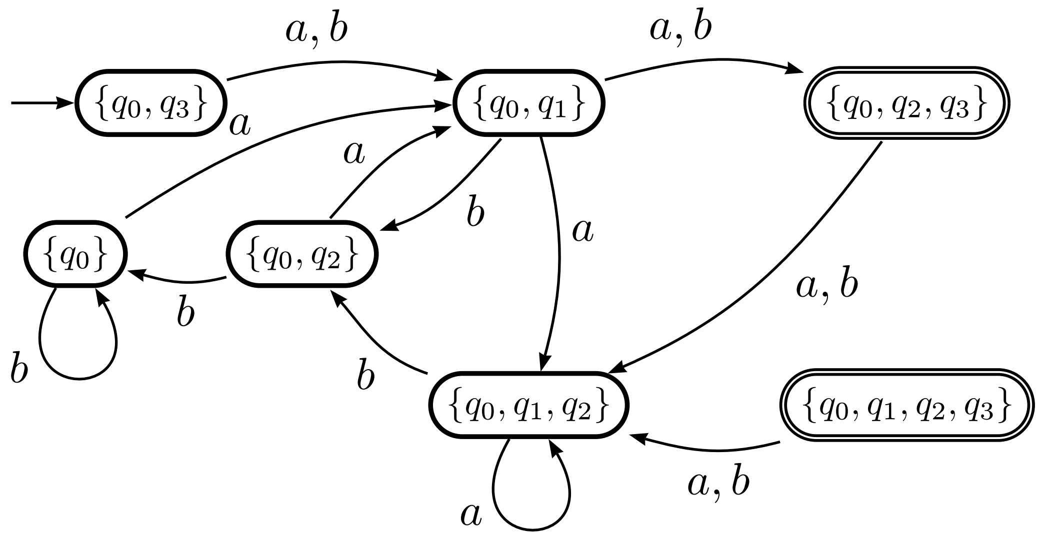
Věta 17: Pro každý KNA A existuje KDA A^D tak, že $L(A) = L(A^D)$.

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.

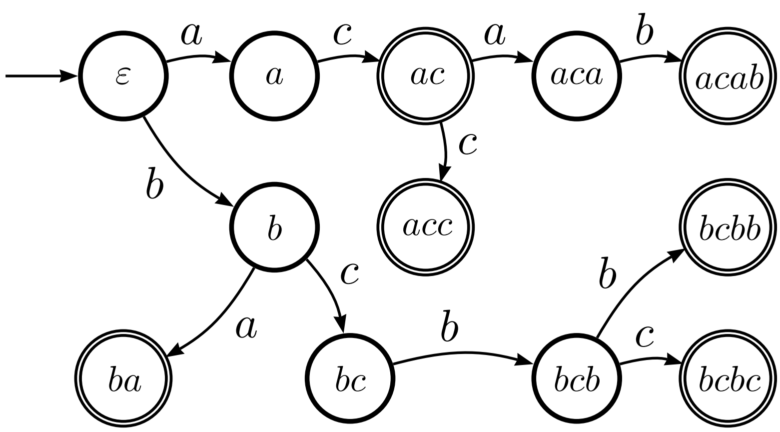
Definice 11: *Trie* je prefixový strom, který umožňuje „rychlé hledání ve slovníku."

Definice 12: Jako *slovník* označujeme konečný neprázdný jazyk který neobsahuje ε.

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 $A = \langle \Sigma, N \cup \{\#\}, \delta, \{S\}, \{\#\} \rangle$.

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

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 δ^* 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 $A, \ldots, x$ 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 $\delta^*(\{A\}, x) = \delta^*(\{A\}, by) = \delta^*(\delta(A, b), y) = \delta^*(\{B, \ldots\}, y) \supseteq \delta^*(\{B\}, y)$ 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 δ plyne, že $A \to bB \in P$
5. Aplikací indukčního předpokladu, existuje P-derivace $B, \ldots, y$. Hledaná P-derivace je ve tvaru: $A, bB, \ldots, by = x$,tj. $A \Rightarrow_G^* x$.

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

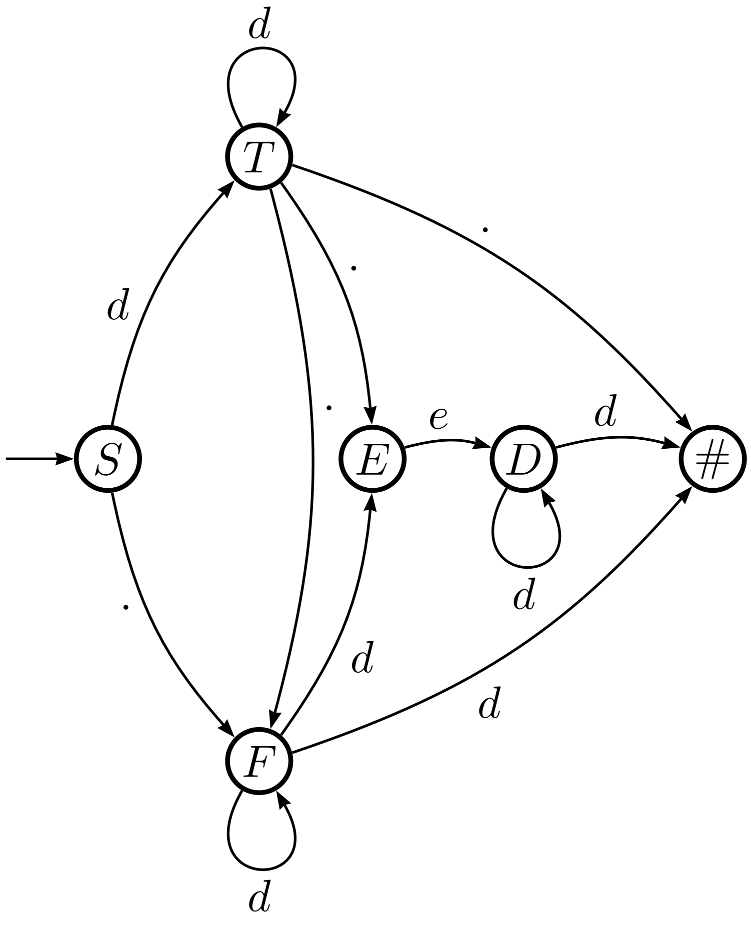
1. Přidáme S do množiny koncových stavů.
2. Přidáme # mezi počáteční stavy
3. Zavedeme nový stav, který bude počáteční a zároveň koncový a nevedou z něj žádné přechody jinam.

Poznámka 7: Nyní zbývá automat pouze determinizovat.

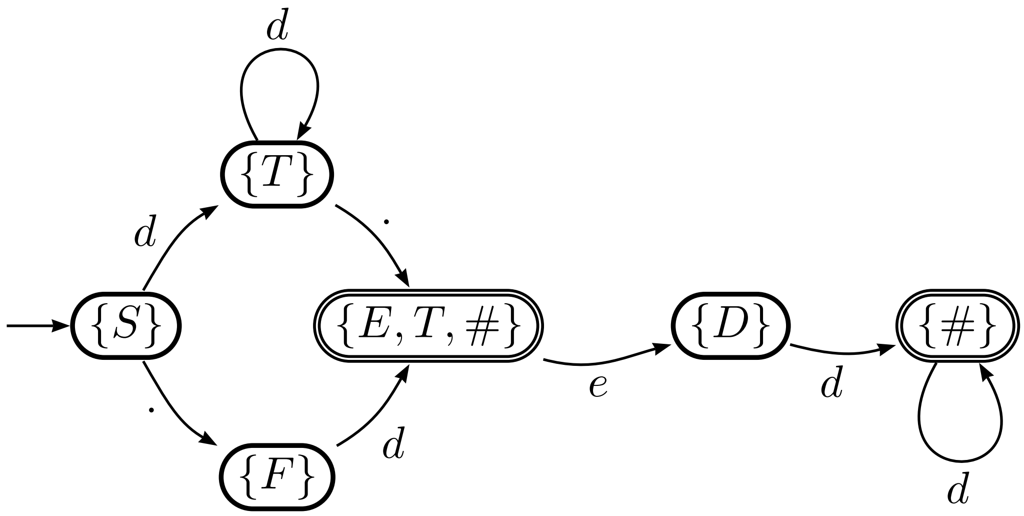
Příklad 24: Máme gramatiku G.

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

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.

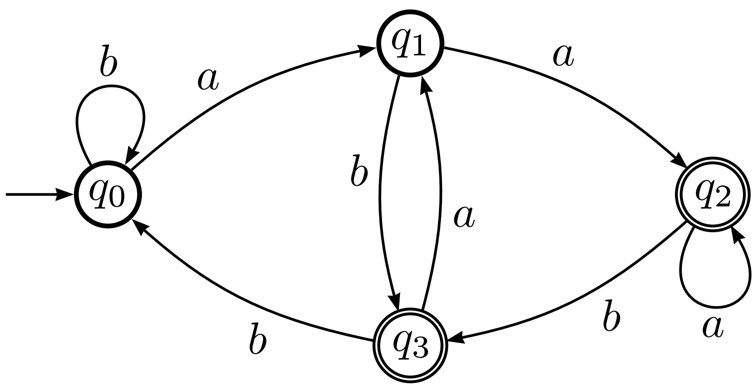
Pokud A nepřijímá ε , pak jsme hotovi.

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:

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

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 δ

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

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

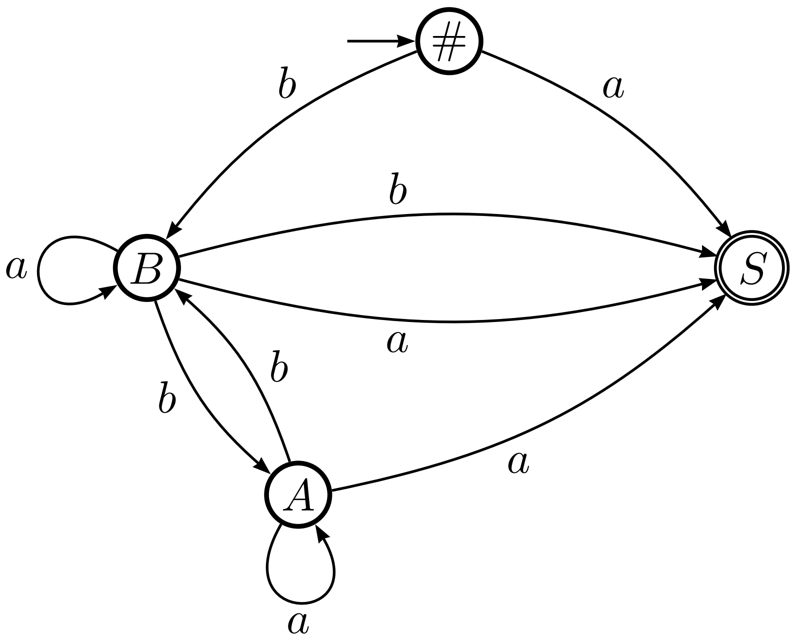
Pro derivaci:

$x_0 = S, x_1, x_2, \ldots, x_{n-1}, x_n = x$
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:

$$\begin{array}{rcl} q_0 & \to & q_0b \mid b \mid q_3b \\ q_1 & \to & q_0a \mid a \mid q_3a \\ q_2 & \to & q_1a \mid q_2a \\ q_3 & \to & q_1b \mid q_2b \\ S & \to & q_1a \mid q_1b \mid q_2a \mid q_2b \end{array}$$

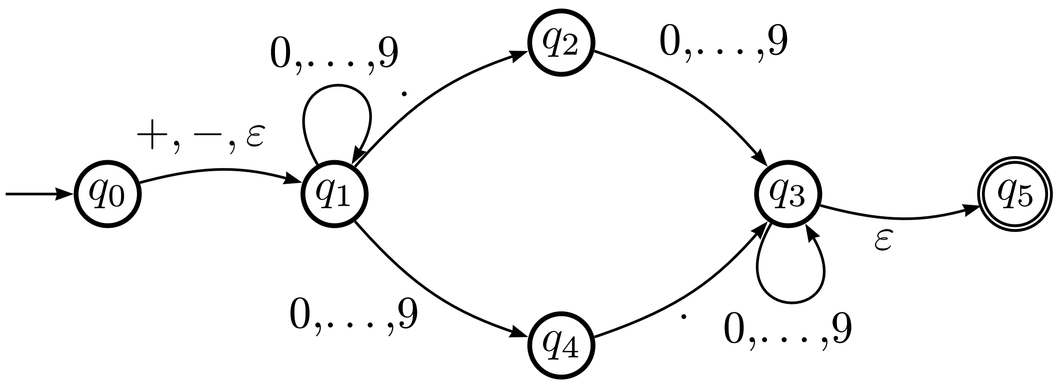
Definice 14: Regulární jazyky jsou jazyky, generované zprava (zleva) regulárními gramatikami, tj. jsou rozpoznatelné konečnými ne/deterministickými automaty.

Poznámka 8: Pravidla zprava a zleva nelze míchat.

11. Nedeterministický konečný automat s ε-přechody

Značíme εKNA.

Příklad 28: Zde je jeden motivační příklad na úvod.



Definice 15: Nedeterministický konečný automat s ε-přechody je struktura $\langle \Sigma, Q, \delta, I, F\rangle$, kde Σ, Q, δ, I, F mají stejná význam jako u KNA. δ je přechodová funcke $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$.

Fakt $\delta(q, a) = \{r_1, \ldots, r_k\}$ čteme: „automat A při čtení symbolu a přejde ze stavu q do některého ze stavů $r_1, \ldots, r_k$"

Fakt $\delta(q, \varepsilon) = \{r_1, \ldots, r_k\}$ čteme: „automat A přejde samovolně ze stavu q do některého ze stavů $r_1, \ldots, r_k$"

11.1. Reprezentace εKNA

1. **Přechodová tabulka**, vypadá stejně jako u KNA s tím, že přidáme jeden sloupec, ve kterém budeme zaznamenávat ε-přechody.

   Takto bude vypadat předchozí příklad 28, reprezentovaný pomocí tabulky.

	+,-	.	$0,\dots,9$	ε
$\rightarrow q_0$	$\{q_1\}$	$\emptyset$	$\emptyset$	$\{q_1\}$
q_1	$\{q_2\}$	$\emptyset$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\emptyset$	$\emptyset$	$\{q_3\}$	$\emptyset$
q_3	$\emptyset$	$\emptyset$	$\{q_3\}$	$\{q_5\}$
q_4	$\emptyset$	$\{q_3\}$	$\emptyset$	$\emptyset$
q_5	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

2. **Přechodový diagram**, u kterého mohou být některé hrany ohodnoceny ε. Viz příklad 28.

11.2. Nedeterministický výpočet

Pojem **konfigurace** pro nás zůstavává stejný, jedná se stále o dvojici $\langle$*stav,řetězec*$\rangle$.

Definice 16: Mějme automat $A = \langle \Sigma, Q, \delta, I, F \rangle$ a řetězec $w \in \Sigma^*$. Posloupnost konfigurací $\langle r_i, w_i \rangle$ pro $i = 0, \dots, n$ splňující podmínky:

1. $r_0 \in I, w_0 = w$
2. $w_n = \varepsilon$
3. pro každé $i = 0, \dots, n$
   (a) $w_i = aw_{i+1}, r_{i+1} \in \delta(r_i, a)$
   (b) $w_i = w_{i+1}, r_{i+i} \in \delta(r_i, \varepsilon)$

11.3. ε-uzávěry množin stavů

Je dána množina stavů $R \subseteq Q$

R se nazývá ε-uzavřená, pokud $\delta(q, \varepsilon) \subseteq R$ pro každý stav $q \in Q$.

Příklad 29: Lze ukázat na našem příkladě 28.

$\{q_0\}$ není ε-uzavřená protože $\delta(q_0, \varepsilon) = \{q_1\} \nsubseteq \{q_0\}$

$\{q_0, q_1\}$ je ε-uzavřená

ε-uzávěr R

vstup: $R \subseteq Q$

$E_0 = R$

a pro $i \geq 1$

$E_i = E_{i-1} \cup \{\delta(q, \varepsilon) | q \in E_{i-1}$

$E_A(R) = \bigcup_{i=0}^{\infty} E_i$

$E_0 \subseteq E_1 \subseteq E_2 \subseteq \dots \subseteq E_A(R)$

Poznámka 9: Vzhledem ke konečnosti množiny $E_A(R)$ musí existovat indek k, pro který platí $E_k = E_{k+1} = \dots = E_A(R)$

Můžeme pozorovat, že $E_A(R)$ není jen ε-uzavřená, ale je také **nejmenší** ε-uzavřená. Z toho můžeme vyvodit, že

$$E_A : 2^Q \to 2^Q$$

je **uzávěrový** operátor.

11.4. Rozšířená přechodová funkce

Pro automat $A = \langle \Sigma, Q, \delta, I, F \rangle$ je rozšířená přechodová funkce definovaná jako
$\delta^* : 2^Q \times \Sigma^* \to 2^Q$

$$\delta^*(R, w) = \begin{cases} E_A(R) & \text{pokud} \quad w = \varepsilon \\ \delta^*(E_A(\bigcup_{q \in E_A(R)} \delta(q, a), u)) & \text{pokud} \quad w = au, a \in \Sigma, u \in \Sigma^* \end{cases}$$

Věta 22: Pro libovolné množiny $R_i \subseteq Q \quad (i = 1, \ldots, k)$ platí:
$\bigcup_{i=1}^k E_A(R_i) = E_A(\bigcup_{i=1}^k R_i)$

Důkaz 17: Z monotonie E_A dostáváme
$E_A(R_i) \subseteq E_A(\bigcup_{i=1}^k R_i)$ pro všechna i
$\bigcup_{i=1}^k E_A(R_i) \subseteq E_A(\bigcup_{i=1}^k R_i)$
Opačná inkluze
$\bigcup_{i=1}^k E_A(R_i)$ je ε-uzavřená a obsahuje $\bigcup_{i=1}^k R_i$
z extenzivity E_A plyne, že $\bigcup_{i=1}^k R_i \subseteq \bigcup_{i=1}^k E_A(R_i)$
Stačí tedy ukázat, že $\bigcup_{i=1}^k E_A(R_i)$ je ε-uzavřená.
$q \in \bigcup_{i=1}^k E_A(R_i) \Rightarrow \exists k \; q \in E_A(R_i)$
Protože $E_A(R_i)$ je ε-uzavřená, $\delta(q, \varepsilon) \in E_A(R_i) \Rightarrow \delta(q, \varepsilon) \in \bigcup_{i=1}^k E_A(R_i) \Rightarrow \bigcup_{i=1}^k (E_A(R_i))$ je ε-uzavřená množina.

Nechť $A = \langle \Sigma, Q, \delta, I, F \rangle$ je εKNA. Řetězec $w \in \Sigma^*$ nazýváme řetězec **přijímaný** A, pokud
$\delta^*(I, w) \cap F \neq \emptyset$
jinak w nazýváme řetězec **zamítaný** A.

11.5. Ekvivalence s KDA

Věta 23: Ke každému εKNA $A = \langle \Sigma, Q, \delta, I, F \rangle$ existuje KNA $A^S = \langle \Sigma, Q, \delta^S, I^S, F \rangle$ takový, že $L(A) = L(A^S)$.

Důkaz 18: $I^S = E_A(I)$
$\delta^S(q, a) = \delta^*(\{q\}, a) = E_A(\bigcup_{u \in E_A(\{q\})} \delta(u, a))$
Indukcí přes délku řetězce $w \in \Sigma^*$ prokážeme, že $\delta^{S^*}(E_A(R), w) = \delta^*(R, w)$

1. $w = \varepsilon$ platí triviálně

2. Předpokládejme, že tvrzení platí pro w délky n a dokážeme pro slova w délky $n+1$.

$$w = av, \quad a \in \Sigma, \quad v \in \Sigma^*, \quad |v| = n, \quad R \subseteq Q$$

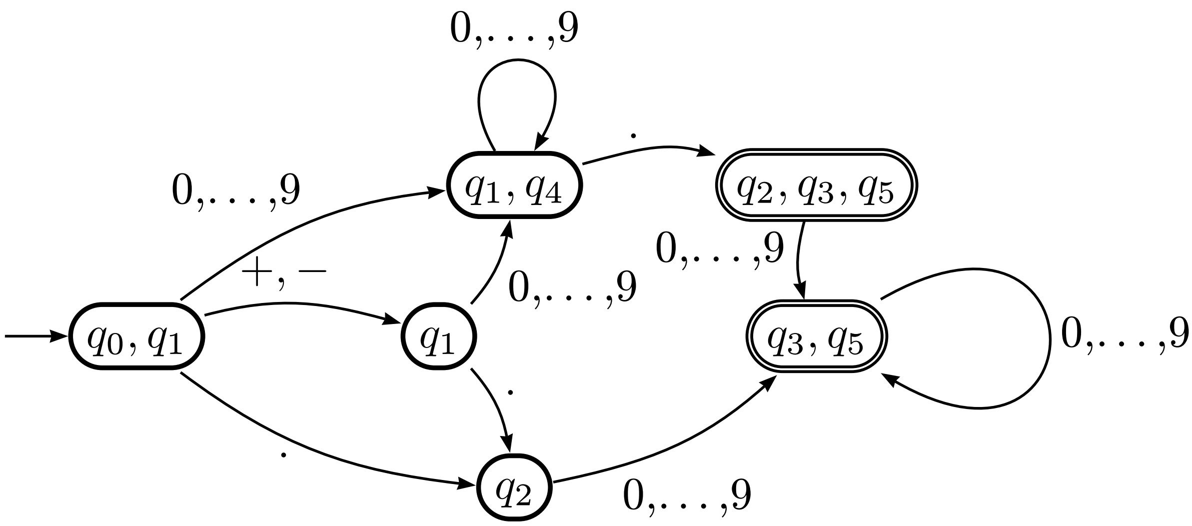
$$\begin{aligned}
\delta^{S^*}(E_A(R), w) &= \delta^{S^*}(E_A(R), av) \\
&= \delta^{S^*}(\bigcup_{q \in E_A(R)} \delta^S(q, a), u) \\
&= \delta^{S^*}(\bigcup_{q \in E_A(R)} E_A(\bigcup_{u \in E_A(\{q\})} \delta(u, a)), u) \\
&= \delta^{S^*}(E_A \bigcup_{q \in E_A(R)} \bigcup_{u \in E_A(\{q\})} \delta(u, a)), u) \\
&= \delta^{S^*}(E_A \bigcup_{q \in E_A(R)} \delta(q, a)), u) \\
&= \delta^{S^*}(E_A(E_A \bigcup_{q \in E_A(R)} \delta(q, a))), u) \\
&= \delta^*(E_A \bigcup_{q \in E_A(R)} \delta(q, a)), u) \\
&= \delta^*(R, w)
\end{aligned}$$

12. Algoritmus na převod εKNA na KDA

Máme $A = \langle \Sigma, Q, \delta, I, F \rangle$ a víme, že W je neprázdná množina dosud nezpracovaných stavů.

```
δ^s ← ∅; Q^s ← ∅; F^s ← ∅; W ← E_A(I)
while W ≠ Q do
     select R ∈ W
      W ← W ∖ R; Q^s ← Q^s ∪ R
     if R ∩ F ≠ ∅ then
          F^s ← F^s ∪ R
     endif
     foreach a ∈ Σ do
          N ← δ^*(R, a)
         if N ≠ ∅ then
             if N ∉ W ∪ Q^s then
                  W ← W ∪ N
             endif
             %%% je to R, W, N ???
              δ^s ← δs ∪ <R, W, N>
         endif
     end
end
return <Σ, Q^s, δ^s, I, F^s>
```

Obrázek 4. Pseudokód pro převod εKNA na KDA.



13. Regulární výrazy

Definice 17: Nechť je dána $\Sigma = \{a_1, \dots, a_k\}$. Pak regulární výraz na Σ je:

1. $\emptyset$
2. ε
3. symboly $a_1, \dots, a_k$
4. pokud R_1, R_2 jsou RV, pak $(R_1|R_2)$ je RV
5. pokud R_1, R_2 jsou RV, pak $(R_1 R_2)$ je RV
6. pokud R je RV, pak $(R^*$ je RV

Příklad 30: Podívejme se na následující výrazy a rozhodněme, zda-li jsou regulární:

- $((ab)c)^*$, $((a|b)c)^*$ – jsou RV
- $a^*b)$, $a||b$ – nejsou RV

14. Jazyky generované regulárními výrazy

Definice 18: Nechť R je RV nad Σ. Pak $L(R) \subseteq \Sigma^*$. Pak také platí:

1. $L(R) = \emptyset$, pokud $R = \emptyset$.
2. $L(R) = \{\varepsilon\}$, pokud $R = \varepsilon$.
3. $L(R) = \{a_i| \quad pokud R = a_i\}$.
4. $L(R) - L(R_1) \cup L(R_2)$, pokud $R = \{R_1|R_2\}$.
5. $L(R) - L(R_1) \circ L(R_2)$, pokud $R = \{R_1 \circ R_2\}$.

6. $L(R) = L(R_1)*$, pokud $R = R_1*$.

Věta 24: Každý regulární výraz lze převést na konečný automat.

Věta 25: Každý jazyk generovaný regulárním výrazem je regulární.

Důkaz 19: Předchozí body dokážeme indukcí dle složitosti regulárního výrazu.

- Pro body 1 až 3 je vše zřejmé.
- Pro bod 4 – $R = R_1|R_2$, kde R_1, R_2 jsou regulární výrazy. Předpokládáme, že existuje KDA, rozpoznávající jazyky $L(R_1)$ a $L(R_2), L(R_1) = L(A_1), L(R_2) = L(A_2)$.

  Pak vytvoříme KNA, který má tvar $\langle \Sigma, Q_1 \cup Q_2, \delta 1 \cup \delta_2, \{q_0, q_0^{`}\}, F_1 \cup F_2 \rangle$.
- Pro bod 5 – $R = R_1 R_2$. Z koncových stavů A_1 vytvoříme ε-přechody do počátečního stavu automatu A_2 a počátečním stavem bude stav q_0 z automatu A_1.
- Pro bod 6 – $R = R_1^*, L(A_1) = L(R_1)$. Následně sestavujeme εKNA tak, že před počátečním stavem vytvoříme nový koncový stav, který bude navíc novým počátečním stave a do kterého vedeme ε přechody z již existujících koncových stavů a z našeho nového koncového stavu navíc vedeme ε přechod do původního počátečního stavu.

Věta 26: Každý regulární jazyk lze generovat regulárním výrazem.

15. Uzávěrové vlastnosti regulárních jazyků

15.1. Základní uzávěrové vlastnosti

- **Komplement**

  Pokud je L regulární, pak je $\Sigma^* \backslash L$ regulární.

  Důkaz 20: Pro L existuje KDA s úplnou přechodovou funkcí tak, že $L(A) = L$. Na základě tohoto automatu zkonstruujeme $A' = \langle \Sigma, Q, \delta, q_0, Q \backslash F \rangle$, A i A' mají stejnou rozšířenou přechodovou funkci δ^*.

  Tj. $\Sigma^* \backslash L = L(A')$

- **Sjednocení**

  Jsou-li L_1 a L_2 regulární, pak je $L_1 \cup L_2$ regulární.

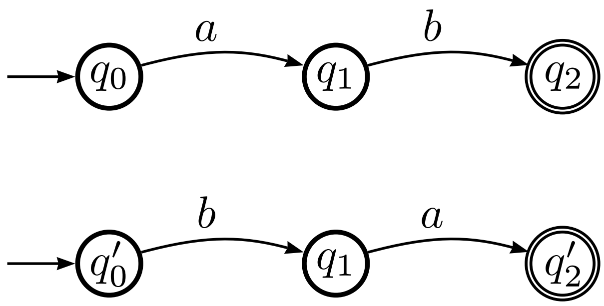
  Důkaz 21: Pro L_1 existuje $A_1 = \langle \Sigma, Q_1, \delta_1, q_{01}, F_1 \rangle$, tak že $L_1 = L(A_1)$

  Pro L_2 existuje $A_2 = \langle \Sigma, Q_2, \delta_2, q_{02}, F_2 \rangle$, tak že $L_2 = L(A_2)$

  Předpokládame-li, že A_1 a A_2 mají disjunktní množiny stavů, tj. $Q_1 \cap Q_2 = \emptyset$, sestavíme následující KNA:
  $A = \langle , Q_1 \cup Q_2, \delta, \{q_{01}, q_{02}\}, F_1 \cup F_2 \rangle$ s přechodovou funkcí δ definovanou jako

$$\delta(q, a) = \begin{cases} \{\delta_1(q, a)\} & \text{pokud} \quad q \in Q_1 \\ \{\delta_2(q, a)\} & \text{pokud} \quad q \in Q_2 \end{cases}$$

Příklad 31: Nyní si uvedeme protipříklad, co by se stalo, kdyby množiny stavů nebyly disjunktní.



První automat přijímá řetězec ab, druhý automat přijímá řetězec ba, čili od jejich sjednocení očekáváme, že bude přijímat ab i ba. Jelikož množiny stavů nejsou disjunktní (q_1 je společný pro oba), sjednocení těchto automatů může stejně dobře přijímat i řetězce aa nebo bb, což je nežádoucí.

- **Průnik**

  S použitím De Morganových zákonů, dostáváme:

  $$L_1 \cap L_2 = \Sigma^* \backslash (\Sigma^* \backslash L_1 \cup \Sigma^* \backslash L_2)$$

  tj. $L_1 \cap L_2$ je regulární.

  Důkaz 22: Uvažujeme konečné deterministické automaty s úplnou přechodovou funkcí $A_1 = \langle \Sigma, Q_1, \delta_1, q_{01}, F_1 \rangle$ a $A_2 = \langle \Sigma, Q_2, \delta_2, q_{02}, F_2 \rangle$
  Zkonstuujeme automat $A_1 \times A_2$ (direktní součin):

  $$A_1 \times A_2 = \langle \Sigma, Q_1 \times Q_2, \delta, \langle q_{01}, q_{02} \rangle, F_1 \times F_2 \rangle$$

  s přechodovou funkcí δ:

  $$\delta(\langle q, r \rangle, a) = \langle \delta_1(q, a), \delta_2(r, a) \rangle$$

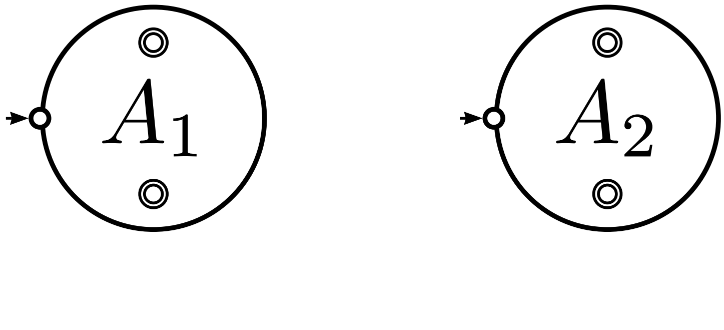
  platí:

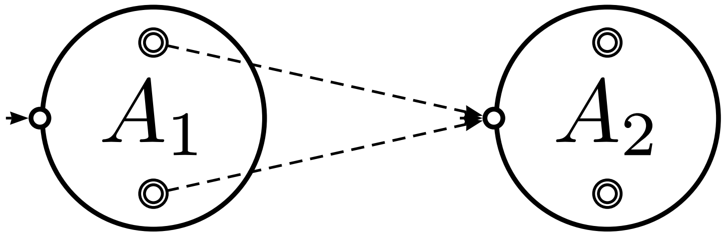
  $$L_1 \cap L_2 = L(A_1 \times A_2)$$

- **Produkt (zřetězení)**

  $L_1 \cdot L_2$

  Předpokládáme existenci automatů $A_1 = \langle \Sigma, Q_1, \delta_1, q_{01}, F_1 \rangle$ a $A_2 = \langle \Sigma, Q_2, \delta_2, q_{02}, F_2 \rangle$ takových, že $L_1 = L(A_1)$ a $L_2 = L(A_2)$.





Sestavíme εKNA

$$A = \langle \Sigma, Q_1 \cup Q_2, \delta, \{q_{01}\}, F_2 \rangle$$

s přechodovou funkcí $\delta : (Q_1 \cup Q_2) \times (\Sigma \cup \{\varepsilon\}) \to 2^{Q_1 \times Q_2}$

$$\delta(q,a) = \begin{cases} \{\delta_1(q,a)\} & \text{pokud} \quad q \in Q_1 \\ \{\delta_2(q,a)\} & \text{pokud} \quad q \in Q_2 \end{cases}$$

$$\delta(q,\varepsilon) = \begin{cases} \{q_{02}\} & \text{pokud} \quad q \in F_1 \\ \emptyset & \text{pokud} \quad q \notin F_1 \end{cases}$$

- **Kleeneho uzávěr**

  Pokud je L regulární, pak je L^* regulární.

  Předpokládáme, že existuje automat A tak, že $L = L(A)$

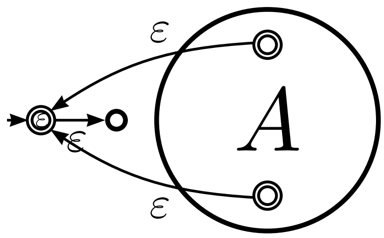
  Poznámka 10: Automat rozpoznavající L^* musí mít možnost dostat se z koncového stavu zpět na začátek.

  Pro automat $A = \langle \Sigma, Q, \delta, q_0, F \rangle$ je potřeba zavést nový počáteční stav $q_T \notin Q$. Konstruovaný εKNA bude vypadat následovně:

  $$A' = \langle \Sigma, Q \cup \{q_T\}, \delta', \{q_T\}, F \cup \{q_T\} \rangle$$

  s přechodovou funkcí δ':

  $$\begin{array}{ll} \delta'(q,a) = \{\delta(q,a)\} & \text{pokud} \quad q \in Q \\ \delta'(q,\varepsilon) = \emptyset & \text{pokud} \quad q \in Q \backslash F \\ \delta'(q,\varepsilon) = \{q_T\} & \text{pokud} \quad q \in F \\ \delta'(q_T,\varepsilon) = \{q_0\} & \\ \delta'(q_T,a) = \emptyset & a \in \Sigma \end{array}$$



15.2. Další uzávěrové vlastnosti

- **Množinový rozdíl**

  Jsou-li L_1 a L_2 regulární, pak $L_1 \backslash L_2 = L_1 \cap (\Sigma^* \backslash L_1)$ je regulární.

- **Kleeneho pozitivní uzávěr**

  Je-li L regulární, pak $L^+ = (L^* \backslash \{\varepsilon\}) \cup L$ je regulární.

- **N-tá mocnina jazyka**

  L^n ... plyne z uzavření na produkt

- **Jazyk reverzních řetězců**

  $L^R = \{w^R \mid w \in L\}$

  Zdůvodníme pomocí konstrukce automatu rozpoznávající L^R (viz cvičení 6)

- **Jazyk sufixů**

  $Sfx(L) = \bigcup_{w \in L} Sfx(w)$

  Důkaz 23: Pro L uvažujeme A tak, že $L = L(A)$. Navíc A je KDA s úplnou přechodovu funkcí takový, že všechny jeho stavy jsou dosažitelné.

  Námi hledaný automat je KNA definován jako:

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

  s přechodovou funkcí

  $$\delta(q, a) = \{\delta(q, a)\}$$

  Poznámka 11: Všechny stavy jsou označeny za počáteční, aby měl automat možnost skočit do libovolné fáze výpočtu a tím "uhádnout" vynechané znaky řetězce, jehož sufix zkoumáme.

- **Jazyk prefixů**

  $$\begin{aligned} Pfx(L) &= \bigcup_{w \in L} Pfx(w) \\ Pfx(L) &= (Sfx(L^R))^R \end{aligned}$$

  Důkaz plyne z uzavření na Sfx a reverzní řetězec.

- **Jazyk infixů**

  $$Ifx(L) = Pfx(Sfx(L))$$

  Důkaz je taktéž zřejmý.

15.3. Pumping lemma

Poznámka 12: Jen drobné upozornění na začátek: jedná se o tvrzení ve tvaru když $\rightarrow$ pak, tj. "Pokud je L regulární, pak ..."

Věta 27: Nechť L je regulární jazyk nad Σ. Pak existuje $n \in \mathbb{N}$ tak, že pro každý řetězec $w \in L$ délky alespoň n platí, že existují $x, y, z \in \Sigma^*$ tak, že jsou splněny podmínky:

1. $w = xyz$

2. $|xy| \leq n$
3. $y \neq \varepsilon$
4. pro každé $i \geq 0$ platí, že $xy^i z \in L$

Důkaz 24: Rozlišíme dva případy.

- L **je konečný**

  pak je tvrzení triviální. Hledané n je ve tvaru $l+1$, kde l je délka nejdelšího řetězce z L. Pak není v L žádný řetězec delší než n a tvrzení 1. – 4. platí triviálně.

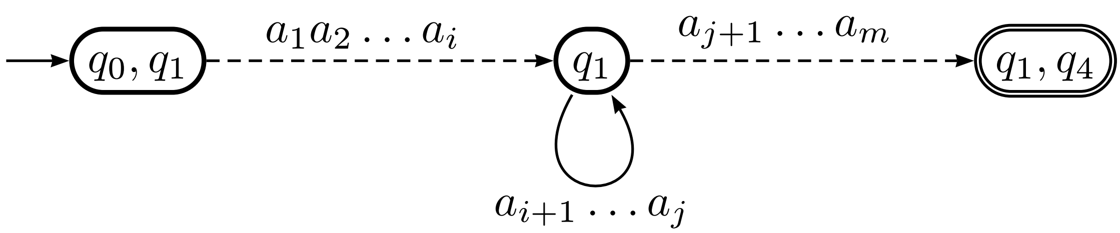
- L **je nekonečný**

  pak pro něj existuje KDA s množinou stavů Q tak, že $L(A) = L$. Položíme $n = |Q|$.

  Pro každý řetězec $w = a_1 a_2 \cdots a_m$ kde $m \geq n$ existuje přijímací výpočet délky m:

  $$\langle q_0, w\rangle = \langle r_0, a_1 \cdots a_m\rangle, \langle r_1, a_2 \cdots a_m\rangle \ldots \langle r_{m-1}, a_m\rangle, \langle r_m, \varepsilon\rangle$$

  Jelikož je $m + 1 > n$ musí existovat alespoň 1 stav, který je v tomto přijímacím výpočtu zopakován.



  Položme

  $$\begin{aligned} x &= a_1 a_2 \ldots a_i \\ y &= a_{i+1} a_{i+2} \ldots a_j \\ z &= a_{j+1} a_{j+2} \ldots a_m \end{aligned}$$

Příklad 32: Jazyk $L = \{a^n b^n | \quad \text{kde } n \in N\}$ není regulární.

Důkaz 25: Předchozí příklad 32 není regulární, což dokážeme sporem. Nechť je tedy L regulární a dle předchozí věty existuje číslo n tak, že vezmem řetězec $a^n b^n = xyz$ tak, že $x = a^k, y = a^l, z = a^{n-k-l} b^n$.

16. Minimalizace KDA

Poznámka 13: Pro regulární jazyk L více, než jeden automat A tak, že $L = L(A)$ a navíc můžeme mít A_1, A_2 tak, že $L(A_1) = L(A_2)$, ale $|Q_1| < |Q_2|$.

16.1. Zprava invariantní ekvivalence

Definice 19: Předpokládejme, že máme $A = \langle \Sigma, Q, \delta, q_0, F \rangle$ a relaci ekvivalence $\Theta \subseteq Q \times Q$ nazveme **zprava invariantní ekvilancí** vzhledem k δ, pokud platí, že $\langle q_r \rangle \in \Theta$ a $a \in \Sigma$, pak $\langle \delta(q,a)\delta(r,a) \rangle \in \Theta$.

Pravá invariance reprezentuje přirozenou vlastnost, kterou musí mít relace nerozlišitelnosti stavů.

Mezními případy invariantních relací zprava jsou:

1. identita $\Theta = \{< p, q > | \quad q \in Q\}$
2. $\Theta = Q \times Q$

16.2. Faktorizace automatu

Definice 20: Mějmse KDA $A = \langle \Sigma, Q, \delta, q_0, F \rangle$ a zprava invariantní ekvivalenci Θ vzhledem k δ, pak zavedeme $A/\theta = \langle \Sigma, Q/\Theta, \delta^{A/\Theta}, [q_0]_\Theta, F^{A/\Theta} \rangle$, kde

$$\delta^{A/\Theta} = ([q]_\Theta, a) = [\delta(q,a)]_\Theta$$

a

$$F^{A/\Theta} = \{[q]_\Theta | \quad q \in F\}$$

.

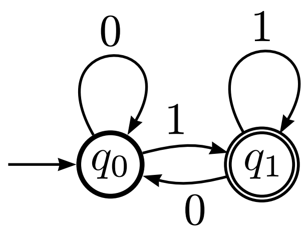
Věta 28: Automat A je dobře definovaný KDA.

Důkaz 26: Větu 28 si ověříme tak, že definice $\delta^{A/\Theta}$ nezávisí na výběru prvků z třídy rozkladu dle Θ.

Mějme $[q]_\Theta = [r]_\Theta$, to jest $\langle q, r \rangle \in \Theta$. Potom pro $a \in \Sigma$ platí $\langle \delta(q,a), \delta(r,a) \rangle \in \Theta$, to jest $[\delta(q,a)]_\Theta = [\delta(r,a)]_\Theta$.

Obecně $L(A/\Theta) \neq L(A)$. Snažíme se najít co největší Θ, tak aby tato rovnost platila.

Příklad 33: $\Theta = Q \times Q$



Věta 29: Platí, že $(\delta^{A/\Theta})^*([q]_\Theta, x) = [\delta^*(q,x)]_\Theta$ pro každý $x \in \Sigma^*$.

Důkaz 27: Dokážeme indukcí přes délku řetězce.

- Tedy $x = \varepsilon$, pak
$$(\delta^{A/\Theta})^*([q]_\Theta, \varepsilon) = [q]_\Theta = [\delta^*(q,\varepsilon)]_\Theta$$
- pak nechť toto tvrzení platí pro $u \in \Sigma^*$ a řetězec $x = au$, kde $a \in \Sigma$.
$$(\delta^{A/\Theta})^*([q]_\Theta, au) = (\delta^{A/\Theta})^*(\delta^{A/\Theta}([q]_\Theta, a), u) =$$
$$= (\delta^{A/\Theta})^*([\delta(q,a)]_\Theta, u) = [\delta^*(\delta(q,a), u]_\Theta = [\delta^*(q,au)]_\Theta$$

Definice 21: Mějme KDA $A = \langle \Sigma, Q, \delta, q_0, F\rangle$ s úplnou přechodovou funkcí. Pro stavy $q, r \in Q$ položme $q \equiv_A r$, pokud pro každý řetězec $x \in \Sigma*$ platí, že $\delta^*(q,x) \in F$, pokud $\delta^*(r,x) \in F$.

Věta 30: $\equiv_A$ je zprava invariantní operace.

Důkaz 28: Dokazujeme větu 30. Důkaz reflexivity je zřejmý, stejně tak symetrie i tranzitivita. Nechť $q \equiv_A r$ a máme $a \in \Sigma$. Máme dokázat, že $\delta(q,a) \equiv_A \delta(r,a)$. Pro každéD $x \in \Sigma^*$ platí:

$$\begin{aligned} \delta^*(\delta(q,a), x) &= \delta^*(q, ax) \\ \delta^*(\delta(r,a), x) &= \delta^*(r, ax) \end{aligned}$$

Užitím faktu, že $\delta^{*(q,ax} \in F$ dostaneme $\delta^*(r,ax) \in F$, tedy $\delta^*(\delta(q,a),x) \in F$ právě tehdy, když $\delta^*(\delta(r,a),x) \in F$, to jest $\delta(q,a) \equiv_A \delta r, a$ a tedy $A/\equiv_A$.

Věta 31: Pro $A/\equiv_A$ a stav $q \in Q$ platí, že $q \in F$ právě, když $[q]_{\equiv_A} \in F^{A/\equiv_A}$.

Důkaz 29: Předchozí větu dokážeme tak, že dokážme implikace z obou stran.

- Implikace zleva doprava („$\Rightarrow$“) plyne z definice.
- Implikace zprava doleva („$\Leftarrow$“): pokud $[q]_{\equiv_A} \in F^{A/\equiv_A}$, pak z definice $\exists r \in F$ tak, že $r \in [q]_{\equiv_A}$. To znamená, že $r \equiv_A q$ pro každý $x \in \Sigma^*$ platí, že $\delta^*(r,x) \in F$ právě tehdy, když $\delta^*(q,x) \in F$, speciálně pro $x = \varepsilon$, navíc $\delta^*(r,\varepsilon) = r \in F$, to jest $\delta^*(q,\varepsilon) = q \in F$.

Důsledek 2: KDA A nazveme redukovaný, pokud je $\equiv_A$ identita.

Věta 32: KDA A je identita.

Důkaz 30: Automat $A \equiv_A$ označme jako B, následně prokážeme, že $\equiv_B$ je identita, tozn., že pokud $[q]_{\equiv_A} \equiv_B [r]_{\equiv_A}$, tak pak jsou si rovny.

Předpokládejme, že platí $[q]_{\equiv_A} \equiv [r]_{\equiv_A}$.

- Pak podle definice $\equiv_B$ tozn, že pro $x \in \Sigma^*$ platí $\delta^{A/\equiv_A}([q]_{\equiv_A}, x) \in F$ právě, když $(\delta^{A/\equiv_A})^*([r]_{\equiv_A}, x) \in F^{A/\equiv_A}$.
- S využitím věty 29 pro každé $x \in \Sigma^*$:
$$[\delta^*(q,x)] \in F^{A/\equiv_A} \text{ právě tehdy, když } [\delta^*(r,x)]_{\equiv A} \in F^{A/\equiv_A}$$
Následně aplikujeme větu 31:
$$\delta^*(q,x) \in F \text{ právě tehdy, když } \delta^*(r,x) \in F \text{ , tozn., že } q \equiv_A r \text{ a tedy } [q]_{\equiv_A} = [r]_{\equiv_A}$$

Věta 33: $L(A) = L(A/\equiv_A)$, což plyne užitím vět 31 a 29.

Důkaz 31: Máme dokázat, že $\delta^* \in F$, právě, když $(\delta^{A/\equiv_A})([q_0]_{\equiv_A}, x) \in F^{A/\equiv_A}$.
Z věty 29: $\delta^*(q_0, x)$ právě, když $[\delta^*(q_0, x)]_{\equiv_A} \in F^{A/\equiv_A}$, to ale platí z věty 31.

Důsledek 3: Obecně platí, že $L(A) \subseteq L(A/\Theta)$.

Věta 34: Pokud je každý stav automatu A dosažitelný, pak má $A/\equiv_A$ také každý stav dosažitelný.

16.3. Algoritmus pro hledání redukovaného automatu

1. Označíme $A/\equiv_A\ jakoA^R$, konstruujeme posloupnost rozkladů na Q tak, že $\varphi_1, \varphi_2, \ldots, \varphi_i = \varphi_{i+1}$, kde $\varphi_1 = \{F, Q \setminus F\}$
2. Při odvození rozkladů φ_1 z φ_{i-1} postupujeme následovně:
   (a) Vezmeme libovolný stav $r \in R$.
   (b) Položíme $S = \{s \in R|$ pro každé $a \in \Sigma$ platí také, že $\delta(S, a) \in [\delta(R, a)]\}$.
   (c) Pokud $S = R$, pak vložíme R do φ_i, pokud $S \subsetneq R$, pak vložíme S a $R \setminus S$ do φ_i.

Věta 35: Korektnost algoritmu pro nalezení $\equiv_A$: pokud $\varphi_i = \varphi_{i+1}$, pak $\varphi_i = Q/\equiv_A$.