Šárka Vavrečková

Teorie jazyků a automatů

Sbírka příkladů pro cvičení

Slezská univerzita v Opavě
Filozoficko-přírodovědecká fakulta
Ústav informatiky

Opava, poslední aktualizace 20. května 2009

Anotace: Tento dokument obsahuje příklady ke cvičením z předmětu Teorie jazyků a automatů I. Studenti zde najdou řešené příklady s podrobně popsaným postupem a neřešené příklady, na kterých si mohou sami postupy procvičit.

Teorie jazyků a automatů – sbírka příkladů pro cvičení

RNDr. Šárka Vavrečková, Ph.D.

Dostupné na: `http://fpf.slu.cz/~vav10ui/formal.html`

Ústav informatiky
Filozoficko-přírodovědecká fakulta
Slezská univerzita v Opavě
Bezručovo nám. 13, 746 01 Opava

Sázeno v systému LaTeX

Obsah

Kapitola 1

Konečné automaty

1.1 Jazyky

U konečných automatů je nejmenší (atomickou, dále nedělitelnou) jednotkou, se kterou pracujeme, *signál* (*znak*). Množinu signálů, se kterými dokáže konkrétní automat pracovat, nazýváme *abeceda* a značíme symbolem Σ (velké řecké písmeno sigma).

Automaty rozpoznávají (tj. přijímají na svém vstupu) posloupnosti signálů (nazýváme je *slova*) – na vstupu postupně čtou signály a zároveň mění svůj vnitřní stav. Jeden automat obvykle dokáže rozpoznávat více takových slov. Množina všech slov, která dokáže automat rozpoznat, je *jazyk* rozpoznávaný automatem. Pokud je automat označen $\mathcal{A}$, pak jazyk jím rozpoznávaný značíme $L(\mathcal{A})$.

Protože jazyky mohou být i hodně rozsáhlé množiny, je třeba si zápis jazyka vhodně zkrátit. Můžeme použít toto značení:

- ε – takto značíme prázdné slovo, tedy řetězec o délce nula (řecké písmeno epsilon)
- a^2, a^3, a^4, ... – počet opakování objektu, který je takto „umocněn“, například a^3 znamená slovo aaa (symbol zřetězení „$\cdot$“ nemusíme psát), a^0 představuje ε (počet signálů a je nula)
- a^* – operátor * (Kleeneho operátor, iterace) znamená, že objekt, za kterým následuje (signál, prvky množiny, apod.) může být ve slově opakován, a to jakýkoliv počet krát (může být i nula opakování), za hvězdičku můžeme dosadit jakoukoliv nezápornou celočíselnou mocninu, tedy a^* představuje množinu slov $\{\varepsilon, a, aa, aaa, \ldots\}$
- $\{a, b, c\}^* = \{\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, \ldots\}$ (operace hvězdička se provádí přes všechny prvky množiny, kterýkoliv z nich může být použit na kterémkoliv místě)
- $(abc)^4 = abcabcabcabc$ (prvky v posloupnosti narozdíl od prvků množiny nejsou

odděleny čárkou, mezi nimi je vztah zřetězení

- $(ab)^* = \{\varepsilon, ab, (ab)^2, (ab)^3, \ldots\} = \{\varepsilon, ab, abab, ababab, \ldots\}$
- $\{a, dfg, bb\}^* = \{\varepsilon, a, dfg, bb, aa, adfg, abb, dfga, dfgdfg, dfgbb, aaa, \ldots\}$
- a^+ je pozitivní iterace, uvedený objekt může být v jakémkoliv počtu výskytů podobně jako u *, ale nejméně jednou, takže $a^+ = \{a, a^2, a^3, \ldots\}$
- operátor pozitivní iterace se z důvodů snadné zaměnitelnosti s jiným operátorem často přepisuje do tvaru a^*a nebo aa^*

Protože jazyk je vlastně množina slov, můžeme použít klasický matematický množinový zápis: $\{a^iba^j \mid i, j \geq 0\}$ je totéž jako a^*ba^*.

Příklad 1.1

Napíšeme všechna slova následujících jazyků kratší než 5.

Postup: nejdřív místo všech „proměnných" indexů * a $^+$ dosadíme nejnižší možnou hodnotu, tj. například ve výrazu a^*b^* vytvoříme slovo $a^0b^0 = \varepsilon$, pak postupně dosazujeme vyšší hodnoty v různých možných kombinacích.

- $\{ab^*, ba\} = \{a, ab, ba, abb, abbb, \ldots\}$
- $2 \cdot \{0, 1\}^* = \{\, 2, 20, 21, 200, 201, 210, 211, 2000, 2001, 2010, 2011, 2100, 2101, 2110, 2111, \ldots\}$
- $\{a^ibb^j \mid i, j \geq 1\} \cup \{(ab)^i \mid i \geq 0\} = \{abb, aabb, abbb, \ldots\} \cup \{\varepsilon, ab, abab, \ldots\} = \{abb, aabb, abbb, \varepsilon, ab, abab, \ldots\}$

  Zde si všimněme odlišných spodních mezí pro proměnné i, j v obou sjednocovaných jazycích – pokud $i \geq 1$, pak a^i znamená a^+ neboli aa^*.

Úkol 1.1

Napište všechna slova následujících jazyků kratší než 5.

- $(abc)^* =$
- $(a^*b)^* =$
- $(ab)^*c =$
- $(a^*b)^*c =$
- $\{(ab)^ib^j \mid i \geq 0,\ j \geq 1\} =$
- $0^*(10)^*1 =$

Konečný automat lze zapsat celkem třemi způsoby:

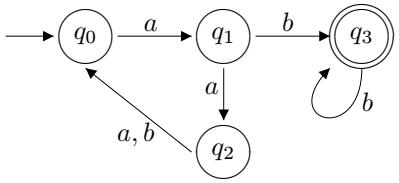
1. použitím plné specifikace s δ-funkcí
2. stavovým diagramem
3. tabulkou přechodů

Ukážeme si všechny tři způsoby.

Příklad 1.2

Vytvoříme konečný automat pro jazyk $a \cdot (a \cdot \{a, b\} \cdot a)^* \cdot b \cdot b^*$ ve všech třech reprezentacích.

U menších automatů je nejjednodušší vytvořit stavový diagram:



δ-funkce včetně plné specifikace:

$\mathcal{A} = (\{q_0, q_1, q_2, q_3\},\ \{a, b\},\ \delta,\ q_0,\ \{q_3\})$

$\delta(q_0, a) = q_1$ $\quad$ $\delta(q_2, a) = q_0$

$\delta(q_1, a) = q_2$ $\quad$ $\delta(q_2, b) = q_0$

$\delta(q_1, b) = q_3$ $\quad$ $\delta(q_3, b) = q_3$

Tabulka přechodů:

	a	b
$\rightarrow\ q_0$	q_1	
q_1	q_2	q_3
q_2	q_0	q_0
$\leftarrow\ q_3$		q_3

Zatímco u stavového diagramu a tabulky přechodů nemusíme uvádět plnou specifikaci – řetězec $\mathcal{A} = (\{q_0, q_1, q_2, q_3\},\ \{a, b\},\ \delta,\ q_0,\ \{q_3\})$, při použití δ-funkce je plná specifikace nutná. Bez ní bychom nepoznali, který stav je počáteční a které stavy jsou koncové.

Úkol 1.2

1. Pro následujích šest jazyků sestrojte konečný automat ve všech třech reprezentacích:

   (a) a^*b
   (b) $\{0, 1\}^*11$
   (c) $11\{0, 1\}^*$
   (d) $(ab)^*ba$
   (e) $\{ab^i \mid i \geq 1\} \cup \{ba^i \mid i \geq 0\}$
   (f) $a^* \cdot \{a, bc\}$

2. Podle zadání vytvořte zbylé dvě reprezentace daného konečného automatu.

   (a) Podle δ-funkce vytvořte stavový diagram a tabulku přechodů:
   $\mathcal{A}_1 = (\{q_0, q_1, q_2\},\ \{a, b, c\},\ \delta,\ q_0,\ \{q_1, q_2\})$
   $\delta(q_0, a) = q_1$ $\quad$ $\delta(q_1, c) = q_1$
   $\delta(q_0, b) = q_2$ $\quad$ $\delta(q_2, c) = q_1$

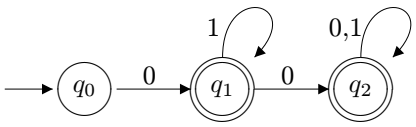
(b) Podle každé tabulky přechodů vytvořte stavový diagram a δ-funkci s plnou reprezentací automatu:

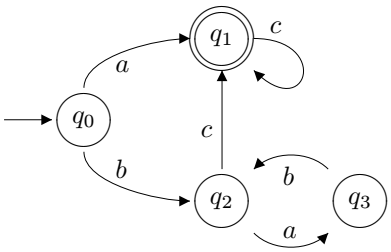
	0	1
→ A	B	C
B	B	C
← C	D	
← D		D

	a	b
↔ q_0	q_1	q_0
q_1	q_1	q_2
← q_2	q_3	
← q_3		

Všimněte si, že podle druhé tabulky je počáteční stav zároveň koncový (šipka vede oběma směry).

(c) Podle každého stavového diagramu vytvořte tabulku přechodů a δ-funkci s plnou reprezentací automatu:

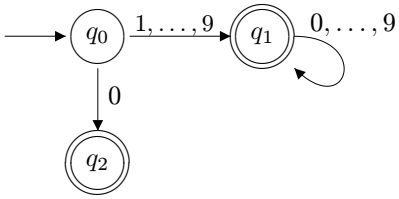




Příklad 1.3

Sestrojíme konečný automat rozpoznávající celá čísla. Používáme číslice 0 . . . 9, číslo se musí skládat z alespoň jedné číslice. Víceciferná čísla nesmí začínat nulou.

Vytvoříme automat se třemi stavy. Oddělíme zpracování jednociferného čísla „0“ od ostatních, která nulou nesmí začínat.



Úkol 1.3

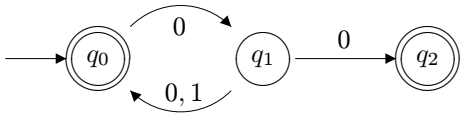
1. Sestrojte konečný automat (stavový diagram) rozpoznávající reálná čísla. Celá část je podle zadání v předchozím příkladu, následuje desetinná čárka a pak opět sekvence číslic (alespoň jedna, může to být i číslice 0).
2. Podle stavového diagramu pro reálná čísla vytvořte tabulku přechodů.

1.2 Nedeterministický konečný automat

Nedeterministický konečný automat je takový automat, kde v alespoň jednom stavu lze na některý signál reagovat více různými způsoby. Protože se tento typ automatu špatně programuje (je těžké vložit do instrukcí náhodnost – zvolit jednu z nabízených cest, a to pokud možno tak, aby vedla „správným směrem“, do koncového stavu), může se hodit postup, jak nedeterministický automat převést na deterministický se zachováním rozpoznávaného jazyka.

Příklad 1.4

Zadaný nedeterministický automat převedeme na deterministický (je dán stavový diagram a tabulka přechodů).



	0	1
↔ q_0	q_1	
q_1	q_0, q_2	q_0
← q_2		

Tento převod je nejjednodušší na tabulce přechodů. V některých buňkách je více než jedna položka, proto obsah buněk budeme chápat jako množiny. Vytváříme tedy nový konečný (deterministický) automat takový, že jeho stavy jsou množinami stavů původního automatu.

Novou tabulku přechodů tedy vytvoříme z původní tabulky tak, že nejdřív jak obsah buněk, tak i stavy v označení řádků uzavřeme do množinových závorek. Tím ale v některých buňkách (zde v jedné) dostaneme stav, který není v označení žádného řádku. To napravíme jednoduše tak, že přidáme nový řádek tabulky s tímto označením a obsah buněk zjistíme sjednocením řádků původní tabulky označených prvky množiny, se kterou právě pracujeme:

...			
$\{A\}$	...	$\{B,C\}$	...
...			
$\{B\}$	$\{D,E\}$	...	$\{G\}$
$\{C\}$	$\{F\}$	...	$\{H,I\}$
...			

⇒

...			
$\{A\}$	...	$\{B,C\}$	...
...			
$\{B\}$	$\{D,E\}$	...	$\{G,H\}$
$\{C\}$	$\{F\}$	...	$\{H,I\}$
...			
$\{B,C\}$	$\{D,E,F\}$	...	$\{G,H,I\}$

Může se stát, že sjednocením množin v buňkách opět vznikne množina, která se nenachází v označení žádného řádku. Pak opět vytvoříme nový řádek a pro buňky použijeme operaci sjednocení množin. Postup je tedy rekurzivní.

Takže k příkladu:

	0	1
$\leftrightarrow \{q_0\}$	$\{q_1\}$	
$\{q_1\}$	$\{q_0, q_2\}$	$\{q_0\}$
$\leftarrow \{q_2\}$		

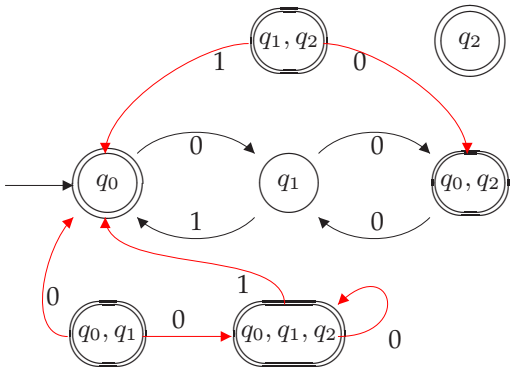
$\Rightarrow$

	0	1
$\leftrightarrow \{q_0\}$	$\{q_1\}$	$\emptyset$
$\{q_1\}$	$\{q_0, q_2\}$	$\{q_0\}$
$\leftarrow \{q_2\}$	$\emptyset$	$\emptyset$
$\leftarrow \{q_0, q_2\}$	$\{q_1\} \cup \emptyset = \{q_1\}$	$\emptyset \cup \emptyset = \emptyset$

Vlastnost „být koncovým stavem“ se také dědí v rámci sjednocení – pokud alespoň jeden z prvků množiny stavů je v původním automatu koncovým stavem, stává se koncovým stavem i celá tato množina.

Uvedený postup je vlastně zkrácený. Ve skutečnosti bychom měli postupovat tak, že bychom jako stavy použili všechny možné kombinace stavů a opět operaci sjednocení množin:

	0	1
$\leftrightarrow \{q_0\}$	$\{q_1\}$	$\emptyset$
$\{q_1\}$	$\{q_0, q_2\}$	$\{q_0\}$
$\leftarrow \{q_2\}$	$\emptyset$	$\emptyset$
$\leftarrow \{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0\}$
$\leftarrow \{q_0, q_2\}$	$\{q_1\}$	$\emptyset$
$\leftarrow \{q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0\}$
$\leftarrow \{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_0\}$



Ve stavovém diagramu můžeme vidět, že stavy, které jsou oproti předchozímu postupu navíc, jsou nedosažitelné z počátečního stavu a tedy se nenacházejí na žádné cestě při zpracování slov jazyka. Proto je vlastně můžeme bez újmy na obecnosti z automatu odstranit.

Úkol 1.4

Uvedené nedeterministické konečné automaty reprezentované tabulkami převeďte na ekvivalentní deterministické. Ke každému vytvořte stavový diagram původního nedeterministického automatu i vytvořeného deterministického a porovnejte.

	a	b
$\leftrightarrow X$	Y	
Y	Z	
Z	X	X, Z

	0	1
$\rightarrow q_0$	q_0, q_1	q_0, q_1
$\leftarrow q_1$		q_1, q_2
$\leftarrow q_2$		

	a	b	c
$\rightarrow q_0$	q_1, q_2	q_1	
q_1	q_2	q_1, q_2	
$\leftarrow q_2$	q_0		q_2

1.3 Totální automat

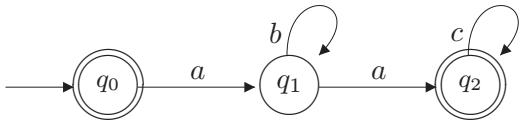
V totálním (úplném) automatu lze v každém stavu reagovat na jakýkoliv symbol. Při převodu (netotálního) automatu na totální vytvoříme nový stav („odpadkový koš“, chybový stav), do kterého přesměrujeme všechny chybějící přechody. Nesmíme zapomenout, že požadavek možnosti reakce na kterýkoliv symbol se vztahuje také na tento nově přidaný stav.

Postup převodu na totální automat lze použít pouze na deterministický konečný automat, proto u nedeterministického automatu je prvním krokem vždy převod na deterministický.

Příklad 1.5

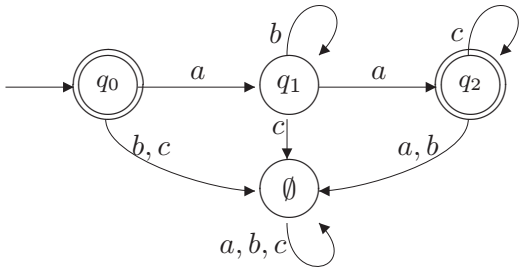
K zadanému deterministickému konečnému automatu vytvoříme ekvivalentní totální automat:

	a	b	c
$\leftrightarrow\ q_0$	q_1		
q_1	q_2	q_1	
$\leftarrow\ q_2$			q_2



Tento automat určitě není totální – například ve stavu q_0 nelze reagovat hned na dva signály. Vytvoříme nový stav, označíme jej symbolem $\emptyset$ a přesměrujeme do tohoto stavu všechny chybějící přechody:

	a	b	c
$\leftrightarrow\ q_0$	q_1	$\emptyset$	$\emptyset$
q_1	q_2	q_1	$\emptyset$
$\leftarrow\ q_2$	$\emptyset$	$\emptyset$	q_2
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$



Příklad 1.6

V příkladu 1.4 jsme k nedeterministickému automatu vytvořili ekvivalentní deterministický. Tento deterministický automat nyní zúplníme (převedeme na totální). Stav $\{q_2\}$ není dosažitelný z počátečního stavu, proto jej také vypustíme.

Abychom trochu zjednodušili značení, budeme místo množin $\{\ldots\}$ používat velká písmena, provedeme přeznačení:

Původní:

	0	1
$\leftrightarrow \{q_0\}$	$\{q_1\}$	$\emptyset$
$\{q_1\}$	$\{q_0, q_2\}$	$\{q_0\}$
$\leftarrow \{q_0, q_2\}$	$\{q_1\}$	$\emptyset$

$\Rightarrow$

Přeznačený:

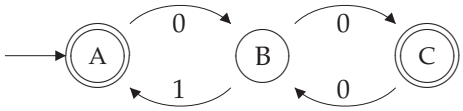
	0	1
$\leftrightarrow A$	B	$\emptyset$
B	C	A
$\leftarrow C$	B	$\emptyset$

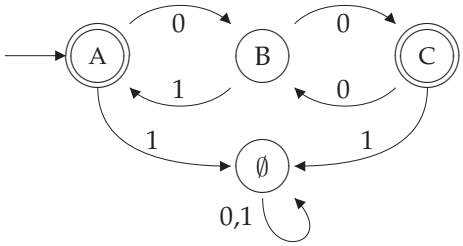
$\Rightarrow$

Totální:

	0	1
$\leftrightarrow A$	B	$\emptyset$
B	C	A
$\leftarrow C$	B	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$

Stavové diagramy přeznačeného a zúplněného automatu:

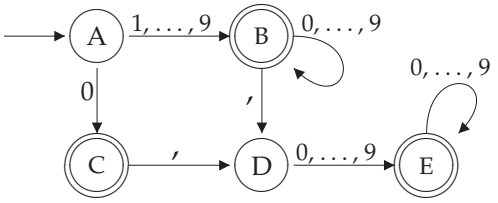




Úkol 1.5

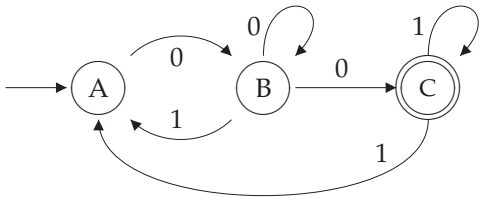
1. Všechny konečné automaty, které jste v úkolu č. 1.4 na straně 6 převedli na deterministické, zúplňte (převeďte na totální).
2. Následující konečný automat převeďte na totální a nakreslete jeho stavový diagram (signál v záhlaví posledního sloupce je desetinná čárka):

	0	1, . . . , 9	,
→ A	C	B	
← B	B	B	D
← C			D
D	E	E	
← E	E	E	



3. Následující (nedeterministický!) konečný automat zúplňte.

	0	1
→ A	B	
B	B, C	A
← C		A, C



1.4 Odstranění nepotřebných stavů

Nepotřebné stavy jsou stavy, které nejsou použity při žádném úspěšném výpočtu (tj. končícím v koncovém stavu). Jedná se o stavy

- *nedosažitelné* – neexistuje k nim cesta z počátečního stavu,
- *nadbytečné* – neexistuje cesta z tohoto stavu do jakéhokoliv koncového stavu.

S odstraňováním (lépe řečeno ignorováním, neuvedením či vypuštěním) prvního typu nepotřebných stavů – nedosažitelných – jsme se setkali už u zkráceného postupu pro převod nedeterministického automatu na deterministický. Pokud nové řádky tabulky přechodů tvoříme jen pro takové množiny původních stavů, které se již vyskytly v některé buňce, většinu nedosažitelných stavů (ale ne vždy všechny) automaticky odstraňujeme.

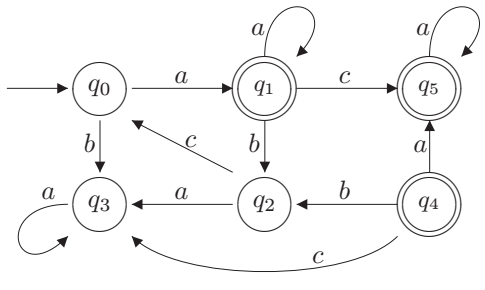
Když chceme odstranit nepotřebné stavy, *vždy* začínáme nedosažitelnými stavy a až potom odstraníme nadbytečné. V obou případech postupujeme rekurzívně, a to buď podle stavového diagramu nebo podle tabulky přechodů.

- *nedosažitelné stavy*: jako bázi (základní množinu) zvolíme $S_0 = \{q_0\}$ (obsahuje pouze počáteční stav), další prvky přidáváme „ve směru šipek“ v stavovém diagramu, resp. v tabulce přechodů ve směru označení řádku $\rightarrow$ obsah buněk na řádku, postupně vytváříme množinu všech stavů, do kterých vede cesta z počátečního stavu o délce max. 0, 1, . . . , n kroků (toto číslo je dolní index u označení vytvářené množiny S_i);
- *nadbytečné stavy*: jako bázi naopak zvolíme množinu koncových stavů – $E_0 = F$, další prvky přidáváme „proti směru šipek“ v stavovém diagramu, resp. ve směru obsah některé buňky tabulky přechodů $\rightarrow$ označení řádku, vytváříme množinu všech stavů, ze kterých vede cesta do některého koncového stavu o délce max. 0, 1, . . . , n kroků.

Příklad 1.7

V zadaném konečném automatu odstraníme nedosažitelné a nadbytečné stavy.

	a	b	c
$\rightarrow\ q_0$	q_1	q_3	
$\leftarrow\ q_1$	q_1	q_2	q_5
q_2	q_3		q_0
q_3	q_3		
$\leftarrow\ q_4$	q_5	q_2	q_3
$\leftarrow\ q_5$	q_5		



Odstraníme nedosažitelné stavy (do kterých neexistuje cesta z počátečního stavu):

$S_0 = \{q_0\}$

$S_1 = \{q_0\} \cup \{q_1, q_3\} = \{q_0, q_1, q_3\}$ (ze stavu q_0 vede přechod do q_1 a q_3)

$S_2 = \{q_0, q_1, q_3\} \cup \{q_5, q_2\} = \{q_0, q_1, q_3, q_5, q_2\}$
$S_3 = \{q_0, q_1, q_3, q_5, q_2\} = S_2$ (konec, v posledním kroku žádný stav do množiny nepřibyl)

V množině S_3 není stav q_4, je tedy nedosažitelný z počátečního stavu a můžeme ho z automatu odstranit. V každé z množin S_i, které jsme postupně vytvořili, najdeme všechny stavy, které jsou z počátečního stavu dosažitelné po nejvýše i krocích.

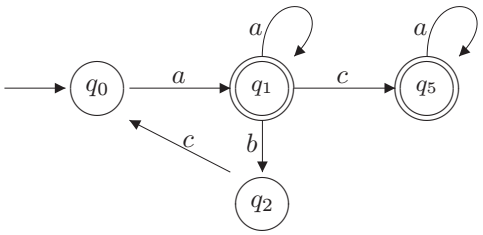
Dále zjistíme, ze kterých stavů nevede cesta do koncových stavů. Budeme pracovat již s automatem po odstranění stavu q_4.

$E_0 = F = \{q_1, q_5\}$
$E_1 = \{q_1, q_5\} \cup \{q_0\} = \{q_1, q_5, q_0\}$ (do stavů z E_0 vede přechod pouze ze stavu q_0)
$E_2 = \{q_1, q_5, q_0, q_2\}$
$E_3 = \{q_1, q_5, q_0, q_2\} = E_2$

V množině E_3 není stav q_3, to znamená, že z tohoto stavu neexistuje žádná cesta do koncového a tedy když tento stav odstraníme, neovlivníme výpočet žádného slova, které automat rozpoznává. Odstraníme tento stav a také všechny přechody s ním přímo související.

Po odstranění stavů, které jsme vyřadili s použitím množin S_i a E_i, dostáváme následující konečný automat:

	a	b	c
$\rightarrow q_0$	q_1		
$\leftarrow q_1$	q_1	q_2	q_5
q_2			q_0
$\leftarrow q_5$	q_5		



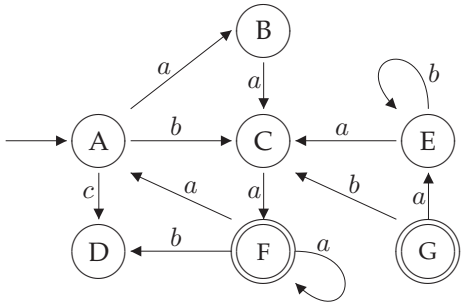
Úkol 1.6

1. Podle zadání následujícího automatu vytvořte tabulku přechodů a stavový diagram. Potom odstraňte nepotřebné stavy. Takto upravený automat pak zúplňte (převeďte na totální automat).

   $\mathcal{A} = (\{A, B, C, D, E, F\}, \{0, 1, 2\}, \delta, A, \{A, C\})$ s funkcí δ:

   $\delta(A, 0) = E$ $\quad$ $\delta(B, 1) = D$ $\quad$ $\delta(D, 2) = C$ $\quad$ $\delta(F, 1) = E$
   $\delta(A, 1) = B$ $\quad$ $\delta(B, 2) = E$ $\quad$ $\delta(E, 1) = E$ $\quad$ $\delta(F, 2) = F$
   $\delta(A, 2) = E$ $\quad$ $\delta(C, 0) = C$ $\quad$ $\delta(E, 2) = E$
   $\delta(B, 0) = C$ $\quad$ $\delta(C, 1) = C$ $\quad$ $\delta(F, 0) = C$

2. K následujícímu konečnému automatu vytvořte zbývající dvě reprezentace – δ-funkci a tabulku přechodů. Potom odstraňte všechny nepotřebné stavy.



3. Podle následujících konečných automatů určených tabulkami přechodů sestrojte stavové diagramy a pak odstraňte všechny nepotřebné stavy.

	a	b	c
$\leftrightarrow$ A	A	A	B
B	C		
C	C	B	
D	A	E	B
$\leftarrow$ E	B	C	C

	a	b	c
$\rightarrow$ q_0	q_4	q_1	
$\leftarrow$ q_1	q_4		q_0
$\leftarrow$ q_2	q_3	q_1	q_2
q_3		q_4	q_3
q_4		q_3	

4. Zamyslete se nad těmito případy: jak by vypadal jazyk upraveného automatu (bez nepotřebných stavů), kdyby do množin S_i nebyly zařazeny žádné koncové stavy? Jak by vypadal tento jazyk, kdyby do množin E_i nebyl zařazen počáteční stav?

Kapitola 2

Regulární jazyky

2.1 Konečné jazyky

Všechny konečné jazyky jsou regulární, proto pro ně dokážeme sestrojit konečný automat.

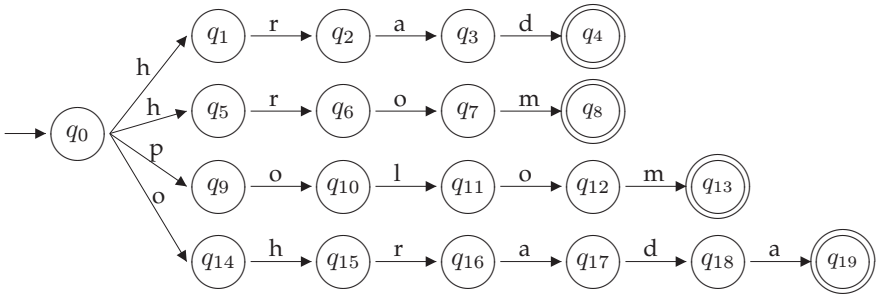
U konečného automatu pro konečný jazyk bývá obvyklý požadavek na rozlišitelnost načítaných slov podle koncového stavu, tedy pro každé slovo jazyka by měl existovat samostatný koncový stav. Pak bez nutnosti porovnávání vstupu s jednotlivými slovy jazyka snadno zjistíme, které slovo bylo načteno – podle koncového stavu, ve kterém skončil výpočet.

Postup je jednoduchý – pro každé slovo jazyka vytvoříme „větev“ ve stavovém diagramu. Jestliže chceme automat deterministický (opět jde o obvyklý požadavek, pokud tento postup používáme při programování), stačí sloučit počátky těch větví, které stejně začínají (konce větví nesmíme sloučit, nebylo by možné rozpoznat slova jazyka podle koncových stavů!).

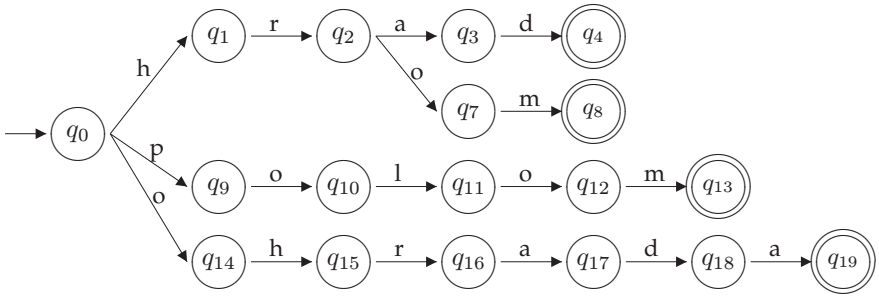
Příklad 2.1

Podle zadaného konečného jazyka vytvoříme konečný automat.

$L = \{\text{hrad, hrom, polom, ohrada}\}$



Dále chceme, aby automat byl deterministický se zachováním možnosti rozlišit rozpoznávaná slova podle koncového stavu. První dvě slova jazyka začínají stejným podřetězcem, tedy začátky prvních dvou větví sloučíme:



Kdybychom netrvali na podmínce odlišení slov koncovými stavy, bylo by možné sloučit všechny koncové stavy v jeden, a také stavy q_7 a q_{12}.

V konečném automatu pro konečný jazyk nesmí (a ani nemůže) být žádná smyčka (cyklus) – kdyby byla, pak by bylo možné tuto smyčku jakýkolivpočetkrát zopakovat a rozpoznávaný jazyk by byl nekonečný.

Úkol 2.1

1. Vytvořte konečný automat pro následující jazyky (deterministický, a to tak, aby pro každé slovo jazyka existoval jeden konečný stav):
   (a) $L_a = \{\text{strom, stroj, výstroj}\}$
   (b) $L_b = \{\text{if, else, elif}\}$
   (c) $L_c = \{\text{read, write, writeall, matrix}\}$
   (d) $L_d = \{\text{delfín, ryba, velryba}\}$

   Poznámka: V případě (c) bude koncový stav větve pro druhé slovo součástí větve pro třetí slovo (pokud vytvoříme deterministický automat). To je v pořádku, vlastnost rozpoznávání podle koncového stavu zůstává zachována.
2. Vytvořte konečný automat (stavový diagram), který bude rozpoznávat všechna slova nad abecedou $\Sigma = \{a, b, c\}$, jejichž délka je
   (a) právě 3 znaky,
   (b) nejvýše 3 znaky (tj. 0, 1, 2 nebo 3 znaky).

   Nápověda: Jde o konečné jazyky. Víme, že v automatu pro konečný jazyk nesmí být cyklus, a také víme, že přechody ve stavovém diagramu mohou být označeny více než jedním znakem.

2.2 Pumping lemma pro regulární jazyky

Pro každý regulární jazyk L platí tato vlastnost: vždy existuje přirozené číslo p takové, že pro každé slovo $w \in L$ existuje (alespoň jedno, případně více) rozdělení $w = x \cdot y \cdot z$ (tj. na tři části) takové, že

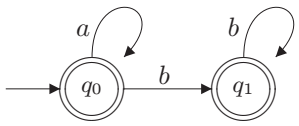
- $|y| > 0$ (v prostřední části musí být alespoň jeden znak – signál),
- $x \cdot (y)^k \cdot z \in L$ pro jakékoliv $k \geq 0$ (když prostřední část „pumpujeme“, výsledné slovo opět patří do jazyka – y buď vyřadíme pro $k = 0$ a nebo opakujeme $1\times$, $2\times$, $3\times$, atd.)

Pumpovací věta je implikace. Říká nám, že *když* je jazyk regulární, *tak* má danou vlastnost. Není nikde řečeno, že jazyk, který není regulární, danou vlastnost nemá.

Příklad 2.2

Nejdřív si ukážeme, jak ověřit, že regulární jazyk tuto vlastnost má. Postup si ukážeme na jazyku $L = \{a^i b^j \mid i, j \geq 0\}$

Pro tento jazyk dokážeme sestrojit konečný automat, tedy víme, že se jedná o regulární jazyk. Vlastnost popsanou v Pumping lemma si můžeme představit takto:



- použijeme p rovno počtu stavů konečného automatu, tedy $p = 2$
- vezmeme jakékoliv slovo jazyka delší než 2, například $w = a^3 b$ (má délku 4)
- w je delší než počet stavů automatu, je tedy zřejmé, že některá část slova bude vyhodnocována v cyklu
- při pohledu na stavový diagram automatu vidíme, že tento cyklus jde přes jediný stav, a to q_0
- vhodné rozdělení může být například $w = (a) \cdot (a^2) \cdot (b)$, po pumpování dostáváme slova $(a) \cdot (a^2)^k \cdot (b) = a^{1+2k} b$, pro jakékoliv $k \geq 0$ tato slova patří do jazyka L.

Platí pumping lemma pro všechny regulární jazyky včetně konečných? Samozřejmě ano. Pro každý konečný jazyk existuje konečný automat, který ho rozpoznává.

Jako p si zvolíme opět buď počet stavů tohoto automatu a nebo číslo ještě větší.

V Pumping lemma se píše „pro každé slovo w daného jazyka delší než $p \ldots$ “, ale nepíše se tam, že takové slovo opravdu musí existovat. Opravdu žádné takové neexistuje, protože kdyby existovalo slovo delší než počet stavů automatu, musel by být v grafu stavového diagramu cyklus, a cyklus znamená nekonečný jazyk. Takže Pumping lemma platí i pro konečné jazyky.

Úkol 2.2

U následujících regulárních jazyků vytvořte stavový diagram, vyberte „dostatečně dlouhé slovo“ – delší než počet stavů automatu, určete některé vhodné rozdělení slova na tři části podle podmínek Pumping lemma a ukažte, že pro jakékoliv číslo (mocninu) k jde opět o slovo z daného jazyka.

Pracujte s těmito jazyky:

(a) $L_a = \{ab^i a \mid i \geq 0\}$

(b) $L_b = \{(ab)^i \mid i \geq 0\}$

(c) $L_c = \{a(bb)^i \mid i \geq 1\}$

(d) $L_d = \{000(111)^i \mid i \geq 0\}$

(e) $L_c = \{\text{disk, data, plat}\}$

(f) $L_d = \mathcal{N}$ (množina přirozených čísel, zde včetně 0)

Obvyklejším způsobem použití Pumping lemma je její obrázená forma: místo tvrzení

$$\text{Regular}(L) \Rightarrow \text{Vlastnost}(L, p)$$

dokazujeme

$$\neg\text{Vlastnost}(L, p) \Rightarrow \neg\text{Regular}(L),$$

tedy „Pokud neexistuje žádné p takové, že pro každé slovo delší než p dokážeme najít dané rozdělení, pak jazyk není regulární.“.

Abychom mohli dokázat, že pracujeme s opravdu jakkoliv dlouhým slovem, použijeme v určení slova „proměnnou“ v exponentu, kterou v případě potřeby můžeme jakkoliv zvyšovat (do nekonečna), například proměnnou i ve slově $(ab)^i$.

Dále u zvoleného slova zkoušíme různé možnosti jeho rozdělení na tři části tak, aby prostřední část nebyla prázdné slovo. Nemusíme hledat absolutně všechny možnosti (to bychom hledali do nekonečna), ale je třeba vystihnout různé možné způsoby umístění hranic mezi potenciálními částmi. Pro každé možné rozdělení pak najdeme alespoň jednu hodnotu k, pro kterou $x \cdot y^k \cdot z$ nepatří do jazyka.

Končíme tehdy, když se podaří najít slovo takové, že pro jakékoliv jeho rozdělení vždy najdeme alespoň jednu hodnotu k takovou, že $x \cdot y^k \cdot z$ nepatří do daného jazyka.

Příklad 2.3

Ukážeme, že jazyk $L = \{a^n b^{2n} \mid n \geq 0\}$ není regulární.

Vybereme „dostatečně dlouhé“ slovo jazyka. Protože nemáme zkonstruován konečný automat pro tento jazyk (samozřejmě, vždyť ve skutečnosti neexistuje), musíme vybrat takové slovo, o kterém si můžeme být jisti, že je opravdu dlouhé. Například $w = a^i b^{2i}$ pro

„nějaké“ i, dostatečně velké. Použili jsme proměnnou i, o které víme jen jedinou konkrétní věc – je větší než 0 (protože musí být větší než nějaké číslo p, které je větší než 0).

Slovo máme, a to dostatečně reprezentativní, vlastně nám určuje všechna dlouhá slova daného jazyka. Zbývá projít všechna možná rozdělení $x \cdot y \cdot z$ tohoto slova a dokázat, že pro jakékoliv $k \geq 0$ slovo $x \cdot y^k \cdot z$ nepatří do jazyka L. U jednotlivých možností budeme volit $k = 0$ nebo $k = 2$.

Jsou tyto možnosti:

1. hranice mezi x a y je před začátkem slova (tj. $x = \varepsilon$):
   (a) hranice mezi y a z je někde mezi znaky a, v první části slova: $x \cdot y \cdot z = \varepsilon \cdot a^m \cdot a^{i-m}b^{2i},\ m > 0$, $x \cdot y^k \cdot z = a^{km+i-m}b^{2i}$, například pro $k = 0$ vychází $a^{i-m}b^{2i}$; protože $m > 0$, toto slovo nepatří do jazyka L
   (b) hranice mezi y a z je na hranici mezi znaky a a b: $x \cdot y \cdot z = \varepsilon \cdot a^i \cdot b^{2i}$, $x \cdot y^k \cdot z = a^{ki}b^{2i}$; opět stačí dosadit $k = 0$, dostáváme b^{2i}, protože i je velké číslo, určitě větší než 0, toto slovo není z jazyka L
   (c) hranice mezi y a z je někde mezi znaky b, v druhé části slova: $x \cdot y \cdot z = \varepsilon \cdot a^i b^m \cdot b^{2i-m},\ 0 < m < 2i$, $x \cdot y^k \cdot z = (a^i b^m)^k b^{2i-m}$, pro $k = 0$ je taktéž narušen vztah v exponentech – slovo $b^{2i-m} \notin L$
   (d) hranice mezi y a z je na konci slova, tj. celé slovo je v prostřední části rozdělení: $x \cdot y \cdot z = \varepsilon \cdot a^i b^{2i} \cdot \varepsilon$, $x \cdot y^2 \cdot z = a^i b^{2i} a^i b^{2i},\ i > 0 \notin L$

2. hranice mezi x a y je v první části slova, někde mezi znaky a:
   (a) hranice mezi y a z je ve stejné části slova: $x \cdot y \cdot z = a^m \cdot a^n \cdot a^{i-m-n}b^{2i},\ m, n > 0$, $x \cdot y^0 \cdot z = a^{i-n}b^{2i} \notin L$
   (b) hranice mezi y a z je na hranici znaků a a b: $x \cdot y \cdot y = a^m \cdot a^{i-m} \cdot b^{2i},\ 0 < m < i$, $x \cdot y^0 \cdot z = a^m b^{2i} \notin L$
   (c) hranice mezi y a z je někde mezi znaky b: $x \cdot y \cdot z = a^m \cdot a^{i-m}b^n \cdot b^{2i-n},\ m, n > 0,\ m < i$, $x \cdot y^2 \cdot z = a^i b^n a^{i-m} b^{2i} \notin L$
   (d) hranice mezi y a z je na konci slova: $x \cdot y \cdot z = a^m \cdot a^{i-m}b^{2i} \cdot \varepsilon,\ 0 < m < i$, $x \cdot y^0 \cdot z = a^m \notin L$

3. hranice mezi x a y je na hranici mezi znaky a a b:
   (a) hranice mezi y a z je někde mezi znaky b: $x \cdot y \cdot z = a^i \cdot b^m \cdot b^{2i-m},\ 0 < m < 2i$, $x \cdot y^0 \cdot z = a^i b^{2i-m} \notin L$
   (b) hranice mezi y a z je na konci slova: $x \cdot y \cdot z = a^i \cdot b^{2i} \cdot \varepsilon$, $x \cdot y^0 \cdot z = a^i \notin L$

4. hranice mezi x a y je v druhé části slova, někde mezi znaky b:
   (a) hranice mezi y a z je někde mezi znaky b: $x \cdot y \cdot z = a^i b^m \cdot b^n \cdot b^{2i-m-n},\ 0 < m, n < 2i$, $x \cdot y^0 \cdot z = a^i b^{2i-n} \notin L$

(b) hranice mezi y a z je na konci slova: $x \cdot y \cdot z = a^i b^m \cdot b^{2i-m} \cdot \varepsilon,\ 0 < m < 2i$, $x \cdot y^0 \cdot z = a^i b^m \notin L$

Jednotlivé varianty rozdělení jsou přehlednější v tabulce:

$x \cdot y \cdot z$	*Podmínky*	$x \cdot y^k \cdot z$	$k = 0$	$k = 2$
$\varepsilon \cdot a^m \cdot a^{i-m} b^{2i}$	$0 < m < i$	$a^{km+i-m} b^{2i}$	$a^{i-m} b^{2i} \notin L$	
$\varepsilon \cdot a^i \cdot b^{2i}$		$a^{ki} b^{2i}$	$b^{2i} \notin L$	
$\varepsilon \cdot a^i b^m \cdot b^{2i-m}$	$0 < m < 2i$	$(a^i b^m)^k b^{2i-m}$	$b^{2i-m} \notin L$	
$\varepsilon \cdot a^i b^{2i} \cdot \varepsilon$		$(a^i b^{2i})^k$		$a^i b^{2i} a^i b^{2i} \notin L$
$a^m \cdot a^n \cdot a^{i-m-n} b^{2i}$	$m, n > 0$	$a^{kn+i-n} b^{2i}$	$a^{i-n} b^{2i} \notin L$	
$a^m \cdot a^{i-m} \cdot b^{2i}$	$0 < m < i$	$a^{m+k(i-m)} b^{2i}$	$a^m b^{2i} \notin L$	
$a^m \cdot a^{i-m} b^n \cdot b^{2i-n}$	$m, n > 0,\ m < i$	$a^m (a^{i-m} b^n)^k b^{2i-n}$		$a^i b^n a^{i-m} b^{2i} \notin L$
$a^m \cdot a^{i-m} b^{2i} \cdot \varepsilon$	$0 < m < i$	$a^m (a^{i-m} b^{2i})^k$	$a^m \notin L$	
$a^i \cdot b^m \cdot b^{2i-m}$	$0 < m < 2i$	$a^i b^{km+2i-m}$	$a^i b^{2i-m} \notin L$	
$a^i \cdot b^{2i} \cdot \varepsilon$		$a^i b^{2ki}$	$a^i \notin L$	
$a^i b^m \cdot b^n \cdot b^{2i-m-n}$	$0 < m, n < 2i$	$a^i b^{kn+2i-n}$	$a^i b^{2i-n} \notin L$	
$a^i b^m \cdot b^{2i-m} \cdot \varepsilon$	$0 < m < 2i$	$a^i b^{m+k(2i-m)}$	$a^i b^m \notin L$	

Pro každé možné rozdělení jsme našli k takové, že $xy^k z$ nepatří do jazyka, proto L není regulární jazyk.

Pumping lemma není ekvivalence, ale pouze implikace. Proto mohou existovat jazyky, které sice nejsou regulární, ale přesto danou vlastnost splňují. Může se jednat například o jazyk vzniklý zřetězením regulárního a neregulárního jazyka.

Příklad 2.4

Jazyk $L = \{a^i b^j c^{2j} \mid i, j \geq 0\}$ není regulární, přesto splňuje Pumping lemma. Můžeme zvolit například slovo a^m. Toto slovo patří do jazyka L a existuje dokonce více různých rozdělení $x \cdot y \cdot z$ takových, že $x \cdot y^k \cdot z \in L$, například $\varepsilon \cdot (a^m)^k \cdot \varepsilon = a^{km} \in L$ pro jakékoliv $k \geq 0$.

Co z toho plyne? Když jazyk splňuje Pumping lemma, nemůžeme tvrdit, že je regulární. A naopak – když se nepodaří pomocí Pumping lemma dokázat, že jazyk není regulární, tak to ještě neznamená, že je regulární.

Úkol 2.3

1. Ukažte, že jazyk $L = \{ba(ab)^n \mid n \geq 0\}$ splňuje Pumping lemma.
2. Pomocí Pumping lemma dokažte, že jazyk $L = \{a^n b c^n \mid n \geq 1\}$ není regulární.

2.3 Uzávěrové vlastnosti – operace nad regulárními jazyky

Třída jazyků je množina všech jazyků s danou společnou vlastností. Také regulární jazyky tvoří třídu – třída regulárních jazyků je množina všech jazyků, pro které lze sestrojit konečný automat.

Víme, že na řetězce lze použít operaci zřetězení. Jazyk je množina řetězců, a tedy na jazyk můžeme použít různé množinové operace (sjednocení, průnik, doplněk – zrcadlení, substituci), a také operace odvozené z práce s řetězci a znaky (signály) – zřetězení, dále iteraci (operace „hvězdička“) a další.

Třída jazyků je uzavřena vzhledem k nějaké operaci, pokud po uplatnění operace na jazyky z této třídy opět vznikne jazyk patřící do stejné třídy.

Postupně si ukážeme všechny základní operace, a to na konečných automatech. Pokud jde o binární operaci (uplatňuje se na dva jazyky), je nutnou podmínkou disjunktnost průniku množin stavů těchto automatů (žádný stav nesmí existovat v obou automatech zároveň).

2.3.1 Sjednocení

V případě sjednocení využijeme celou definici původních automatů. Přidáme nový stav a z tohoto stavu povedeme přechody do všech stavů, do kterých vede přechod z počátečního stavu. Nový stav nám tedy simuluje použití počátečních stavů v původních automatech.

Postup si můžeme představit třeba tak, že vezmeme všechny přechody, které vedou z původních počátečních stavů, a *zkopírujeme* (ne přeneseme, ty původní musejí zůstat) „konce bez šipek“ k novému stavu.

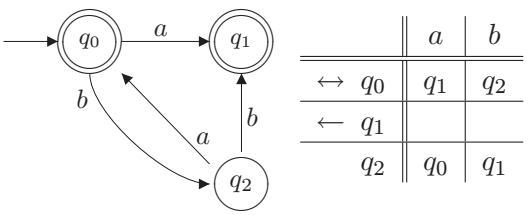
Pokud některý z původních počátečních stavů patří i do množiny koncových stavů, pak také tuto vlastnost přidáme novému počátečnímu stavu.

Příklad 2.5

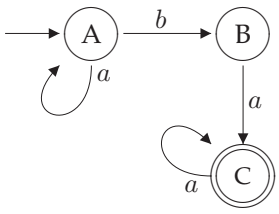
Ukážeme si operaci sjednocení na následujících jazycích a automatech:

$L_1 = \{(ba)^i(a \cup bb) \ : \ i \geq 0\}$
$L_2 = \{a^i baa^j \mid i, j \geq 0\}$

Konečné automaty pro tyto jazyky jsou následující:



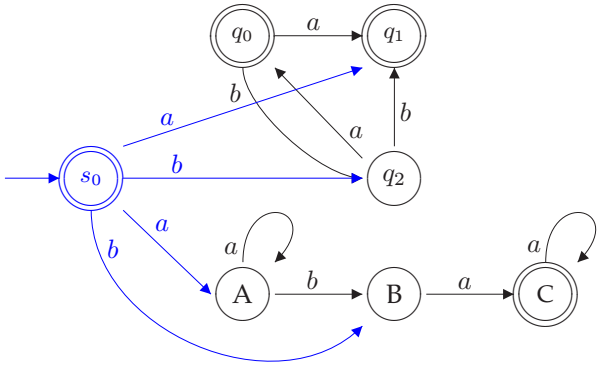
	a	b
$\leftrightarrow$ q_0	q_1	q_2
$\leftarrow$ q_1		
q_2	q_0	q_1



	a	b
$\rightarrow$ A	A	B
B	C	
$\leftarrow$ C	C	

Přidáme nový stav s_0 a všechny přechody z tohoto stavu nasměrujeme a označíme stejně jako přechody z počátečních stavů původních automatů.

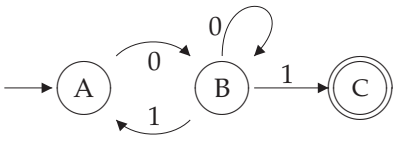
Postup je nejnázornější na stavovém diagramu, ale je jednoduchý také v tabulce přechodů – stačí obě tabulky shrnout do jedné a přidat nový řádek, jehož buňky budou sjednocením buněk na řádcích původních počátečních stavů a na příslušných sloupcích. Protože jeden z původních počátečních stavů (q_0) patří do množiny koncových stavů, také nový počáteční stav s_0 bude zároveň koncovým.

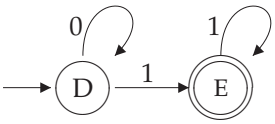


	a	b
$\leftrightarrow$ s_0	q_1, A	q_2, B
$\leftarrow$ q_0	q_1	q_2
$\leftarrow$ q_1		
q_2	q_0	q_1
A	A	B
B	C	
$\leftarrow$ C	C	

Úkol 2.4

1. Použijte operaci sjednocení na následující konečné automaty:





2. Použijte operaci sjednocení na následující konečné automaty (u druhého z nich je stav q_3 zároveň koncový):

	a	b
$\rightarrow$ q_0	q_0, q_2	q_1
$\leftarrow$ q_1		q_2
$\leftarrow$ q_2	q_2	

	a	b	c
$\leftrightarrow$ q_3	q_4	q_6	
q_4		q_5	
q_5			q_3
$\leftarrow$ q_6			

3. Pro oba konečné automaty ze zadání předchozího příkladu a také pro výsledný automat vytvořte zbylé dva typy reprezentací – stavový diagram a δ-funkci včetně plné specifikace automatu.
4. Zamyslete se nad tím, jak by vypadalo sjednocení více než dvou konečných automatů.

2.3.2 Zřetězení

Zřetězení dvou jazyků vytvoříme tak, že ve výsledném jazyce jsou všechny možné řetězce, jejichž první část je kterékoliv slovo z prvního jazyka a druhá část zase kterékoliv slovo z druhého jazyka. Záleží na pořadí, části slova nemůžeme přehodit, řídí se pořadím zřetězovaných jazyků.

Příklad 2.6

Princip zřetězení jazyků si ukážeme na těchto konečných jazycích:

$L_1 = \{\varepsilon, abc, ba\}$
$L_2 = \{ddb, dc\}$

Zřetězením těchto dvou jazyků získáme jazyk $L = L_1 \cdot L_2$:

$L = \{\varepsilon \cdot ddb, \varepsilon \cdot dc, abc \cdot ddb, abc \cdot dc, ba \cdot ddb, ba \cdot dc\} = \{ddb, dc, abcccb, abcdc, baddb, badc\}$

Při zřetězení není třeba ve výsledném automatu vytvářet nový stav. Opět využijeme všechny stavy a přechody z původních automatů a budeme přidávat nové přechody, které je propojí.

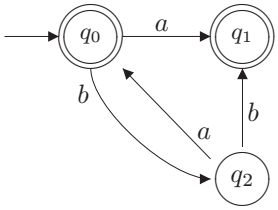
Při ukončení výpočtu první části slova (tj. v prvním původním automatu) je třeba „plynule“ navázat na výpočet druhého původního automatu. Proto nově přidávané přechody budou vést z koncových stavů prvního automatu, jejich cílový stav (stav, do kterého bude směřovat šipka přechodu) a označení přejmeme (zkopírujeme) od přechodů z počátečního stavu druhého původního automatu.

Počátečním stavem výsledného automatu bude samozřejmě počáteční stav prvního původního automatu. Do množiny koncových stavů zařadíme

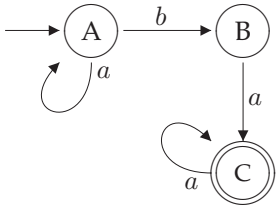
- koncové stavy druhého původního automatu,
- pokud počáteční stav druhého automatu byl zároveň koncovým (to znamená, že do druhého jazyka patřilo slovo ε), pak do množiny koncových stavů výsledného automatu zařadíme také koncové stavy prvního původního automatu.

Příklad 2.7

Zřetězíme jazyky těchto konečných automatů:

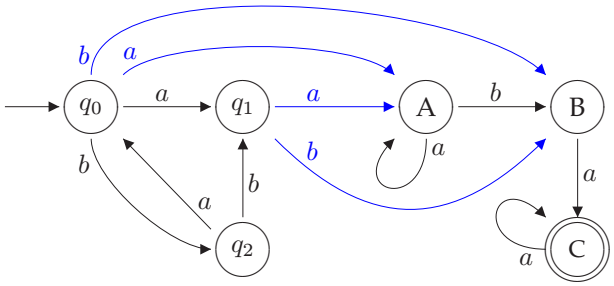


$\mathcal{A}_1$	a	b
$\leftrightarrow\ q_0$	q_1	q_2
$\leftarrow\ q_1$		
q_2	q_0	q_1



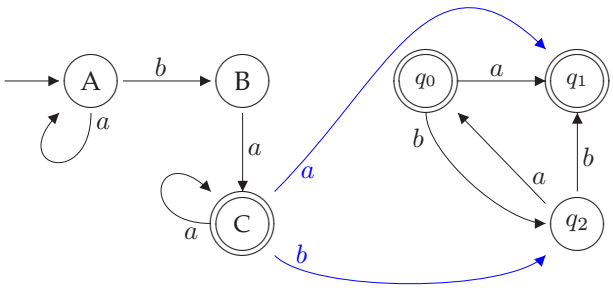
$\mathcal{A}_2$	a	b
$\rightarrow$ A	A	B
B	C	
$\leftarrow$ C	C	

V automatu $\mathcal{A}_1$ (prvním) jsou dva koncové stavy. Z nich budou vést nové přechody ke stavům automatu $\mathcal{A}_2$ – ke všem stavům, do kterých vede přechod z počátečního stavu A.



	a	b
$\rightarrow q_0$	q_1, A	q_2, B
q_1	A	B
q_2	q_0	q_1
A	A	B
B	C	
$\leftarrow$ C	C	

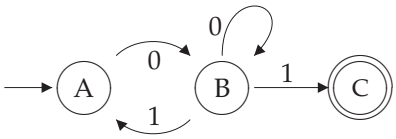
Nyní obrátíme pořadí zřetězovaných automatů. Musíme brát v úvahu to, že počáteční stav automatu, který je teď druhý v pořadí, je zároveň koncovým stavem:

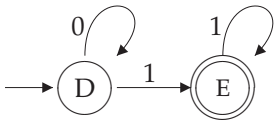


	a	b
$\rightarrow$ A	A	B
B	C	
$\leftarrow$ C	C, q_1	q_2
q_0	q_1	q_2
q_1		
q_2	q_0	q_1

Úkol 2.5

1. Proveďte operaci zřetězení jazyků následujících konečných automatů:





2. Proveďte operaci zřetězení jazyků následujících konečných automatů – nejdřív v pořadí podle zadání ($L(\mathcal{A}_1) \cdot L(\mathcal{A}_2)$) a potom v opačném pořadí ($L(\mathcal{A}_2) \cdot L(\mathcal{A}_1)$). Sestrojte také stavové diagramy.

$\mathcal{A}_1$	a	b
$\rightarrow q_0$	q_0, q_2	q_1
$\leftarrow q_1$		q_2
$\leftarrow q_2$	q_2	

$\mathcal{A}_2$	a	b	c
$\leftrightarrow q_3$	q_4	q_6	
q_4		q_5	
q_5			q_3
$\leftarrow q_6$			

2.3.3 Iterace (Kleeneho uzávěr)

Při iteraci zřetězujeme jazyk sám se sebou, a to $0\times$, $1\times$, $2\times$, ..., a pak všechny výsledné jazyky po zřetězení sjednotíme.

Příklad 2.8

Princip iterace jazyka si ukážeme na tomto konečném jazyce:

$L_1 = \{abc, ba, cd\}$

Iterací získáme jazyk $L = L_1^*$:

$$\begin{aligned} L = \{\, & \varepsilon, && \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots && 0\times \\ & abc, ba, cd, && \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots && 1\times \\ & abcabc, abcba, abccd, baabc, baba, bacd, && \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots && 2\times \\ & abcabcabc, abcabcba, abcabccd, abcbaabc, abcbaba, abcbacd, \ldots\} && \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots && 3\times, \ldots \end{aligned}$$

Výsledkem iterace je vždy nekonečný jazyk obsahující prázdné slovo ε, i kdyby původní jazyk L_1 byl konečný.

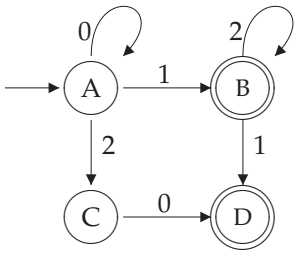
Úprava konečného automatu při iteraci spočívá v těchto dvou krocích:

- vytvoříme nové přechody umožňující „návrat“ z koncových stavů na počátek výpočtu (další slovo z jazyka),
- počáteční stav zařadíme do množiny koncových stavů (pokud tam nebyl).

Nové přechody tedy povedou z původních koncových stavů, a to do všech stavů, do kterých vede přechod z počátečního stavu. Princip je prakticky stejný jako u dříve probíraných operací, kopírujeme počátky přechodů od počátečního stavu se zachováním jejich cíle i označení (signálu).

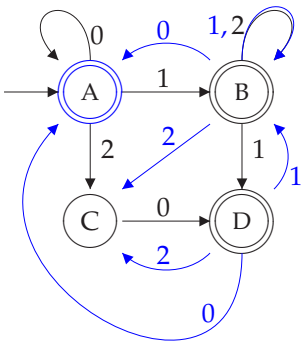
Příklad 2.9

Vytvoříme iteraci jazyka následujícího automatu:



	0	1	2
→ A	A	B	C
← B		D	B
C	D		
← D			

Koncové stavy jsou dva. Z každého přidáme přechody do všech stavů, do kterých směřují přechody z počátečního stavu, a potom počáteční stav označíme jako koncový (aby automat přijímal také prázdné slovo ε).

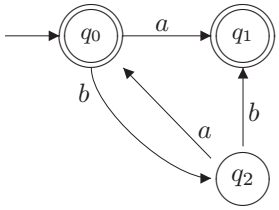


	0	1	2
↔ A	A	B	C
← B	A	D, B	B, C
C	D		
← D	A	B	C

Kdyby stav A už v původním automatu patřil do množiny koncových stavů, tak by samozřejmě koncovým stavem zůstal.

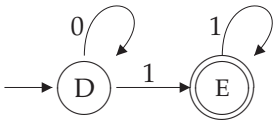
Úkol 2.6

1. Zkonstruujte konečný automat jazyka, který je iterací jazyka následujícího automatu:



$\mathcal{A}_1$	a	b
↔ q_0	q_1	q_2
← q_1		
q_2	q_0	q_1

2. Zkonstruujte konečný automat jazyka, který je iterací jazyka následujícího automatu:



	0	1
→ D	D	E
← E		E

Pro výsledný automat také vytvořte třetí typ reprezentace – δ-funkci s plnou specifikací.

2.3.4 Pozitivní iterace

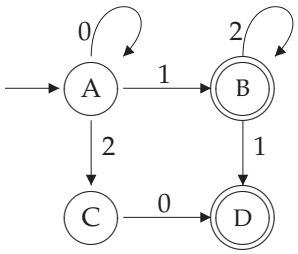
Pozitivní iterace je podobná operace jako předchozí, ale zatímco iterace znamená řetězení $0\times,\ 1\times,\ 2\times,\ \ldots$, při pozitivní iteraci začínáme při řetězení až $1\times,\ 2\times,\ \ldots$. Matematicky můžeme zapsat:

Iterace: $L^* = \bigcup_{i\geq 0} L^i$
Pozitivní iterace: $L^+ = \bigcup_{i\geq 1} L^i$

Postup je stejný jako u iterace, ale pokud počáteční stav původního automatu nepatřil do množiny koncových stavů (tj. slovo ε nepatřilo do jazyka), nebude koncovým stavem ani po úpravě automatu. Postup si ukážeme na stejném jazyce a automatu jako u iterace.

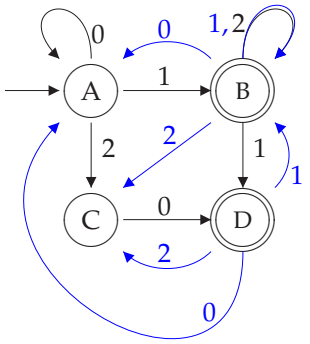
Příklad 2.10

Vytvoříme pozitivní iteraci jazyka následujícího automatu:



	0	1	2
→ A	A	B	C
← B		D	B
C	D		
← D			

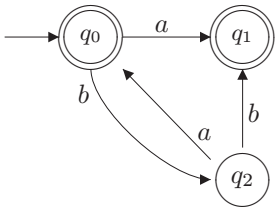
Přidáváme pouze nové přechody, počáteční stav nebudeme zařazovat do množiny koncových stavů:



	0	1	2
→ A	A	B	C
← B	A	D, B	B, C
C	D		
← D	A	B	C

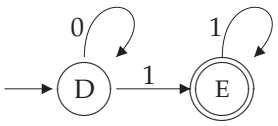
Úkol 2.7

1. Zkonstruujte konečný automat jazyka, který je pozitivní iterací jazyka následujícího automatu:



$\mathcal{A}_1$	a	b
↔ q_0	q_1	q_2
← q_1		
q_2	q_0	q_1

2. Zkonstruujte konečný automat jazyka, který je pozitivní iterací jazyka následujícího automatu:



	0	1
→ D	D	E
← E		E

2.3.5 Zrcadlový obraz (reverze)

Reverze je převrácení. Zrcadlový obraz slova sestrojíme tak, že je zrcadlově „převrátíme“, tj. ze slova $abcd$ získáme slovo $(abcd)^R = dcba$.

Zrcadlový obraz jazyka je jazyk obsahující všechna slova původního jazyka, ale převrácená. Operace zrcadlení je sama k sobě inverzní – když slovo (nebo jazyk) revertujeme dvakrát, dostáváme původní slovo (jazyk). Zrcadlení můžeme zapsat takto:
$L^R = \{w \mid w^R \in L\}$

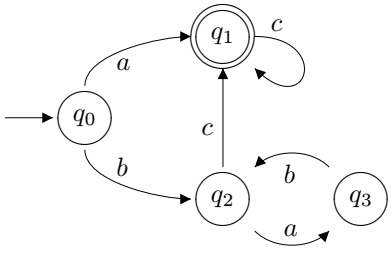
Pokud budeme při reverzi pracovat s konečným automatem, především převrátíme všechny cesty, které v automatu existují (tj. převrátíme všechny přechody) a zaměníme počáteční a koncové stavy.

Dále musíme vyřešit jeden problém – počáteční stav musí být vždy jen jeden, ale koncových stavů může být obecně jakýkoliv počet. Proto rozlišíme dva případy – pokud původní automat má jen jediný koncový stav, stačí postup naznačený v předchozím odstavci. Jestliže však má více koncových stavů, musíme zajistit, aby po reverzi existoval jen jediný počáteční stav. Proto vytvoříme nový stav, který v sobě bude shrnovat vlastnosti původních koncových stavů (resp. nových „počátečních“ stavů) týkající se přechodů, které z nich po reverzi vycházejí.

Pokud počáteční stav původního automatu patřil do množiny koncových stavů, bude koncovým stavem také počáteční stav po reverzi.

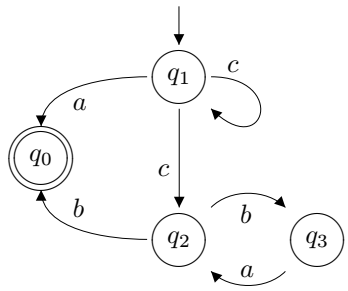
Příklad 2.11

Provedeme operaci zrcadlení jazyka tohoto konečného automatu:



	a	b	c
→ q_0	q_1	q_2	
← q_1			q_1
q_2	q_3		q_1
q_3		q_2	

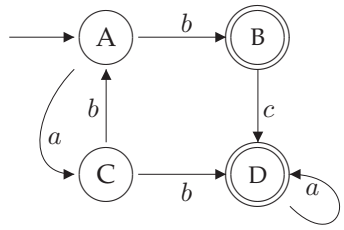
Obrátíme všechny přechody. Protože máme jen jediný koncový stav, stačí dále jen zaměnit počáteční a koncový stav. U tabulky se zaměňují označení řádků s obsahem buněk na tomto řádku.



	a	b	c
$\leftarrow\ q_0$			
$\rightarrow\ q_1$	q_0		q_1, q_2
q_2		q_0, q_3	
q_3	q_2		

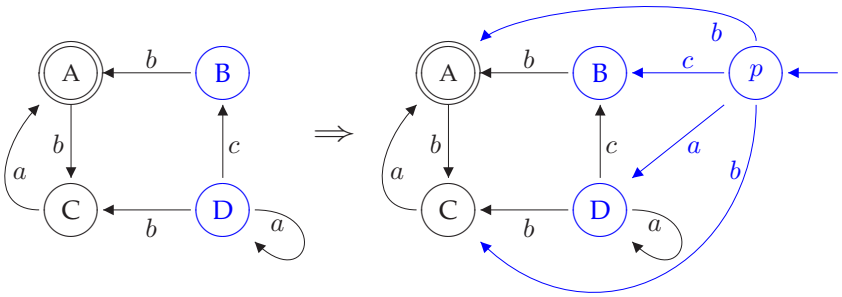
Příklad 2.12

Podíváme se na složitější případ – konečný automat s více koncovými stavy. Postup bude podobný, ale navíc řešíme nutnost existence jediného počátečního stavu. Je dán tento konečný automat:



	a	b	c
$\rightarrow$ A	C	B	
$\leftarrow$ B			D
C		A, D	
$\leftarrow$ D	D		

Jsou zde dva koncové stavy. Nejdřív přesměrujeme všechny přechody a označíme nový koncový stav, a až v druhém kroku (automat vpravo) vytvoříme nový počáteční stav podobným postupem, jaký jsme použili například u sjednocení (tam šlo také o „simulaci“ – nahrazení dvou počátečních stavů jediným).

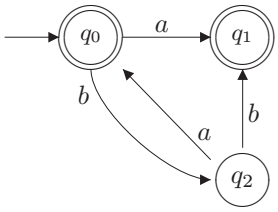


	a	b	c
$\rightarrow\ p$	D	A, C	B
$\leftarrow$ A		C	
B		A	
C	A		
D	D	C	B

Poslední případ, na který se podíváme, je konečný automat s více koncovými stavy, kde také počáteční stav patří do množiny koncových stavů.

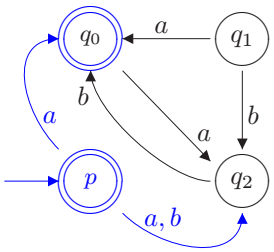
Příklad 2.13

Pro operaci zrcadlení je dán tento konečný automat:



	a	b
$\leftrightarrow$ q_0	q_1	q_2
$\leftarrow$ q_1		
q_2	q_0	q_1

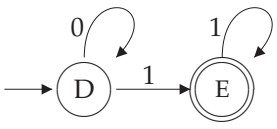
Narozdíl od předchozích příkladů budou ve výsledném automatu dva koncové stavy. Stav q_0 bude koncový, protože v původním automatu je počátečním stavem, a stav p bude koncový, protože do jazyka původního automatu patří slovo ε, jehož převrácením je opět slovo ε pro jeho rozpoznání musí být počáteční stav (tj. p) v množině koncových stavů.



	a	b
$\leftrightarrow$ p	q_0, q_2	q_2
$\leftarrow$ q_0	q_2	
q_1	q_0	q_2
q_2		q_0

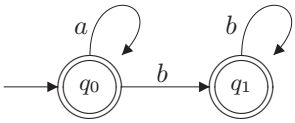
Úkol 2.8

1. Zkonstruujte konečný automat jazyka, který je reverzí (zrcadlovým obrazem) jazyka následujícího automatu:



	0	1
$\rightarrow$ D	D	E
$\leftarrow$ E		E

2. Vytvořte konečný automat jazyka, který je reverzí jazyka následujícího automatu. Pro oba automaty (uvedený i ten, který vytvoříte) sestrojte tabulku přechodů.



3. Vytvořte konečný automat rozpoznávající jazyk $L = \{\varepsilon$, tisk, tis, síto$\}$ tak, aby jednotlivá slova jazyka byla rozpoznávaná koncovým stavem (tj. pro každé slovo vlastní koncový stav). Potom proveďte reverzi tohoto automatu.

4. Vytvořte zrcadlový obraz jazyka tohoto konečného automatu (má jediný koncový stav, který je zároveň počátečním stavem):

	a	b
$\leftrightarrow\ q_0$	q_1	q_2
q_1		q_2
q_2	q_2	q_0

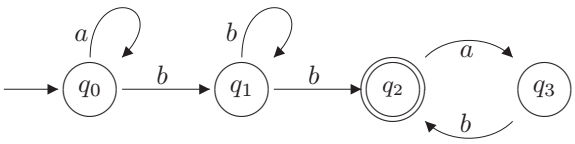
2.3.6 Průnik

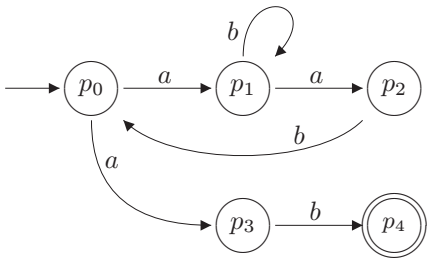
Když vytváříme konečný automat pro průnik dvou jazyků pomocí automatů, které je rozpoznávají, tak vlastně simulujeme (paralelně) činnost obou těchto automatů. Po přečtení každého signálu ze vstupu provedeme jeden krok v obou automatech zároveň.

Ve výsledném automatu jsou proto stavy uspořádanými dvojicemi stavů z prvního a druhého původního automatu.

Příklad 2.14

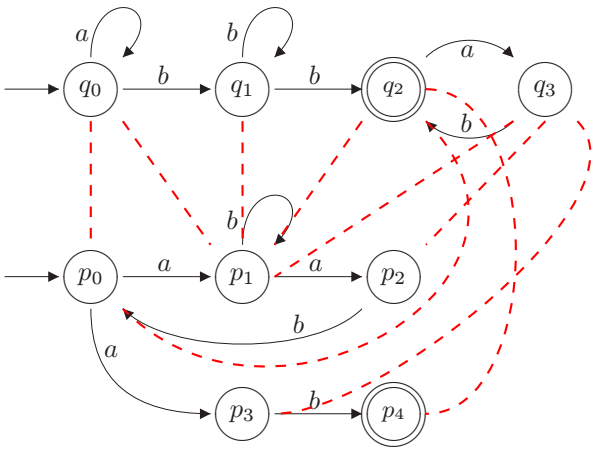
Sestrojíme konečný automat rozpoznávající průnik jazyků následujících dvou automatů:





Nejdřív si vyznačíme průběh „simulace“. Není to nutné (a u složitějšího automatu je to prakticky neproveditelné), ale na diagramu lépe pochopíme paralelnost obou zpracování téhož slova (červeně jsou spojeny stavy, ve kterých jsou automaty ve stejném kroku zpracování slova) – obrázek vpravo.

Například pro slovo $abbabab$ paralelně procházíme dvojicemi stavů $[q_0, p_0]$, pak $[q_0, p_1]$, $[q_1, p_1]$, $[q_2, p_1]$, $[q_3, p_2]$, $[q_2, p_0]$, dále $[q_3, p_3]$ a $[q_2, p_4]$.



Protože by bylo hodně náročné (a také nedeterministické a těžko naprogramovatelné) takto vyhledávat všechny dvojice stavů, ve kterých se nachází výpočet v obou automatech ve stejném kroku, použijeme jinou (trochu „otrockou“) metodu:

- jako množinu stavů použijeme množinu všech uspořádaných dvojic, kde první prvek je stav prvního automatu a druhý prvek je stav druhého automatu,
- vytvoříme δ-funkci nebo tabulku přechodů, ve které zachytíme společné přechody na stejný signál v obou automatech,
- odstraníme nepotřebné stavy.

Počátečním stavem bude uspořádaná dvojice obsahující počáteční stavy obou původních automatů, koncové stavy budou všechny uspořádané dvojice, ve kterých jsou oba původní stavy koncovými.

Jednodušší je využití tabulky přechodů. Podle tabulek původních automatů vytvoříme tabulku pro výsledný automat (kombinace řádků ve stejném sloupci).

	a	b
$\rightarrow\ q_0$	q_0	q_1
q_1		q_1, q_2
$\leftarrow\ q_2$	q_3	
q_3		q_2

	a	b
$\rightarrow\ p_0$	p_1, p_3	
p_1	p_2	p_1
p_2		p_0
p_3		p_4
$\leftarrow\ p_4$		

Tabulka přechodů výsledného konečného automatu má $4 \cdot 5 = 20$ řádků:

	a	b
$\rightarrow\ [q_0, p_0]$	$[q_0, p_1], [q_0, p_3]$	
$[q_0, p_1]$	$[q_0, p_2]$	$[q_1, p_1]$
$[q_0, p_2]$		$[q_1, p_0]$
$[q_0, p_3]$		$[q_1, p_4]$
$[q_0, p_4]$		
$[q_1, p_0]$		
$[q_1, p_1]$		$[q_1, p_1], [q_2, p_1]$
$[q_1, p_2]$		$[q_1, p_0], [q_2, p_0]$
$[q_1, p_3]$		$[q_1, p_4], [q_2, p_4]$
$[q_1, p_4]$		

(pokračování)	a	b
$[q_2, p_0]$	$[q_3, p_1], [q_3, p_3]$	
$[q_2, p_1]$	$[q_3, p_2]$	
$[q_2, p_2]$		
$[q_2, p_3]$		
$\leftarrow\ [q_2, p_4]$		
$[q_3, p_0]$		
$[q_3, p_1]$		$[q_2, p_1]$
$[q_3, p_2]$		$[q_2, p_0]$
$[q_3, p_3]$		$[q_2, p_4]$
$[q_3, p_4]$		

Nyní odstraníme nepotřebné (tj. nedosažitelné a nadbytečné) stavy tak, jak jsme se učili v předchozích kapitolách, obsah množin je průběžně tříděn pro usnadnění porovnávání.

$S_0 = \{[q_0, p_0]\}$
$S_1 = \{[q_0, p_0], [q_0, p_1], [q_0, p_3]\}$

$S_2 = \{[q_0,p_0],[q_0,p_1],[q_0,p_3],[q_0,p_2],[q_1,p_1],[q_1,p_4]\}$
$S_3 = \{[q_0,p_0],[q_0,p_1],[q_0,p_2],[q_0,p_3],[q_1,p_1],[q_1,p_4],[q_1,p_0],[q_2,p_1]\}$
$S_4 = \{[q_0,p_0],[q_0,p_1],[q_0,p_2],[q_0,p_3],[q_1,p_0],[q_1,p_1],[q_1,p_4],[q_2,p_1],[q_3,p_2]\}$
$S_5 = \{[q_0,p_0],[q_0,p_1],[q_0,p_2],[q_0,p_3],[q_1,p_0],[q_1,p_1],[q_1,p_4],[q_2,p_1],[q_3,p_2],[q_2,p_0]\}$
$S_6 = \{[q_0,p_0],[q_0,p_1],[q_0,p_2],[q_0,p_3],[q_1,p_0],[q_1,p_1],[q_1,p_4],[q_2,p_0],[q_2,p_1],[q_3,p_2],[q_3,p_1],$
$[q_3,p_3]\}$
$S_7 = \{[q_0,p_0],[q_0,p_1],[q_0,p_2],[q_0,p_3],[q_1,p_0],[q_1,p_1],[q_1,p_4],[q_2,p_0],[q_2,p_1],[q_3,p_1],[q_3,p_2],$
$[q_3,p_3],[q_2,p_4]\}$
$S_8 = \{[q_0,p_0],[q_0,p_1],[q_0,p_2],[q_0,p_3],[q_1,p_0],[q_1,p_1],[q_1,p_4],[q_2,p_0],[q_2,p_1],[q_2,p_4],[q_3,p_1],$
$[q_3,p_2],[q_3,p_3]\} = S_7$

Odstranili jsme stavy $[q_0,p_4],[q_1,p_2],[q_1,p_3],[q_2,p_2],[q_2,p_3],[q_3,p_0],[q_3,p_4]$, které jsou nedosažitelné z počátečního stavu. Dále pracujeme s touto tabulkou přechodů:

	a	b
$\rightarrow\ [q_0,p_0]$	$[q_0,p_1],[q_0,p_3]$	
$[q_0,p_1]$	$[q_0,p_2]$	$[q_1,p_1]$
$[q_0,p_2]$		$[q_1,p_0]$
$[q_0,p_3]$		$[q_1,p_4]$
$[q_1,p_0]$		
$[q_1,p_1]$		$[q_1,p_1],[q_2,p_1]$
$[q_1,p_4]$		
$[q_2,p_0]$	$[q_3,p_1],[q_3,p_3]$	
$[q_2,p_1]$	$[q_3,p_2]$	
$\leftarrow\ [q_2,p_4]$		
$[q_3,p_1]$		$[q_2,p_1]$
$[q_3,p_2]$		$[q_2,p_0]$
$[q_3,p_3]$		$[q_2,p_4]$

Odstraníme nadbytečné symboly (ze kterých neexistuje cesta do koncového stavu):

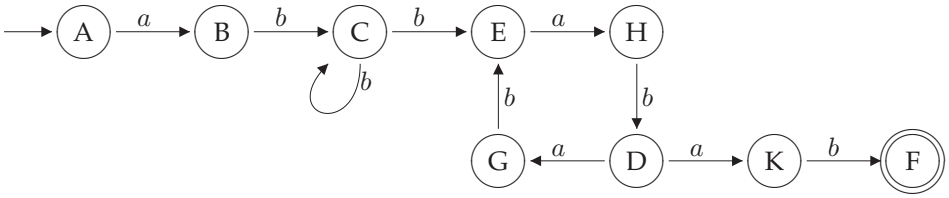
$E_0 = \{[q_2,p_4]\}$
$E_1 = \{[q_2,p_4],[q_3,p_3]\}$
$E_2 = \{[q_2,p_4],[q_3,p_3],[q_2,p_0]\}$
$E_3 = \{[q_2,p_0],[q_2,p_4],[q_3,p_3],[q_3,p_2]\}$
$E_4 = \{[q_2,p_0],[q_2,p_1],[q_2,p_4],[q_3,p_2],[q_3,p_3]\}$
$E_5 = \{[q_2,p_0],[q_2,p_1],[q_2,p_4],[q_3,p_2],[q_3,p_3],[q_1,p_1],[q_3,p_1]\}$
$E_6 = \{[q_1,p_1],[q_2,p_0],[q_2,p_1],[q_2,p_4],[q_3,p_1],[q_3,p_2],[q_3,p_3],[q_0,p_1]\}$
$E_7 = \{[q_0,p_1],[q_1,p_1],[q_2,p_0],[q_2,p_1],[q_2,p_4],[q_3,p_1],[q_3,p_2],[q_3,p_3],[q_0,p_0]\}$
$E_8 = \{[q_0,p_0],[q_0,p_1],[q_1,p_1],[q_2,p_0],[q_2,p_1],[q_2,p_4],[q_3,p_1],[q_3,p_2],[q_3,p_3]\} = E_7$

Po odstranění stavů $[q_0,p_2],[q_0,p_3],[q_1,p_0],[q_1,p_4]$ dostáváme tuto tabulku přechodů (s původním označením stavů a po přeznačení na písmena):

	a	b
$\rightarrow [q_0, p_0]$	$[q_0, p_1]$	
$[q_0, p_1]$		$[q_1, p_1]$
$[q_1, p_1]$		$[q_1, p_1], [q_2, p_1]$
$[q_2, p_0]$	$[q_3, p_1], [q_3, p_3]$	
$[q_2, p_1]$	$[q_3, p_2]$	
$\leftarrow [q_2, p_4]$		
$[q_3, p_1]$		$[q_2, p_1]$
$[q_3, p_2]$		$[q_2, p_0]$
$[q_3, p_3]$		$[q_2, p_4]$

	a	b
$\rightarrow$ A	B	
B		C
C		C, E
D	G, K	
E	H	
$\leftarrow$ F		
G		E
H		D
K		F

Stavový diagram (stavy jsou přeznačené):



Úkol 2.9

1. Vytvořte deterministické konečné automaty pro jazyky $L_1 = \{\text{if, then, else}\}$ a $L_2 = \{\text{if, iff, elif}\}$, v každém z automatů stačí jediný koncový stav pro všechna slova daného jazyka. Potom pomocí automatů vytvořte průnik těchto jazyků.
2. Sestrojte konečný automat, který je průnikem jazyků následujících automatů (pracujte s tabulkami přechodů). Odstraňte všechny nedosažitelné a nadbytečné stavy a sestrojte stavový diagram.

	0	1
$\rightarrow$ A	B	A
$\leftarrow$ B	C	
C		D
D	B	

	0	1
$\rightarrow$ E		F
F	F	H
$\leftarrow$ H	H	H

Pro kontrolu: po odstranění nepotřebných stavů by měl automat mít osm stavů, z toho jeden koncový, jeden cyklus přes jeden stav a jeden cyklus přes tři stavy.

2.3.7 Homomorfismus

Homomorfismus je takové zobrazení, které

- zachovává neutrální prvek (tj. u řetězců prázdný řetězec ε zobrazí opět na ε)
- zachovává zobrazení operace, pro kterou je definován (u řetězců jde o zřetězení, tj. $h(a \cdot b) = h(a) \cdot h(b)$)
- výsledkem zobrazení prvku (signálu, znaku) je vždy řetězec (u substituce, což je zobecnění homomorfismu, je výsledkem zobrazení množina řetězců).

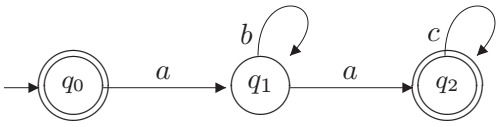
Když konstruujeme konečný automat homomorfismu pro některý jazyk, využíváme plně způsobu definice daného homomorfismu – pokud je v předpisu tohoto zobrazení například $h(a) = xyz$, tj. znak a má být zobrazen na řetězec xyz, vezmeme postupně všechny přechody označené znakem a a nahradíme je cestami rozpoznávajícími slovo xyz. Každá z těch cest musí být samozřejmě samostatná, vždy jeden konkrétní přechod označený signálem a nahradíme jednou samostatnou cestou pro xyz.

Příklad 2.15

Podle následujícího automatu sestrojte konečný automat rozpoznávající jazyk podle zadaného homomorfismu.

	a	b	c
$\leftrightarrow$ q_0	q_1		
q_1	q_2	q_1	
$\leftarrow$ q_2			q_2

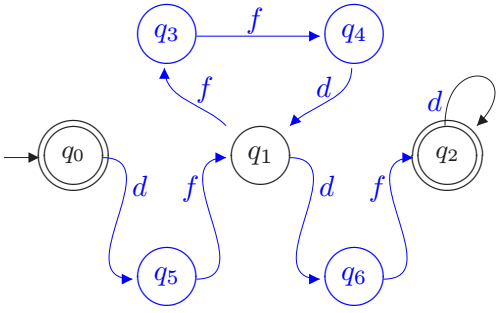
Homomorfismus:
$h(a) = df$,
$h(b) = fffd$,
$h(c) = d$



Po úpravě vypadá konečný automat následovně:

	d	f
$\leftrightarrow$ q_0	q_5	
q_1	q_6	q_3
$\leftarrow$ q_2	q_2	
q_3		q_4
q_4	q_1	
q_5		q_1
q_6	q_2	

Původní abeceda:
$\Sigma_1 = \{a, b, c\}$
Nová abeceda:
$\Sigma_2 = \{d, f\}$



Jazyk rozpoznávaný původním automatem je $L = \{\varepsilon\} \cup \{ab^iac^j \mid i, j \geq 0\}$, po úpravě dostáváme automat rozpoznávající jazyk $h(L) = \{\varepsilon\} \cup \{df(ffd)^idfd^j \mid i, j \geq 0\}$.

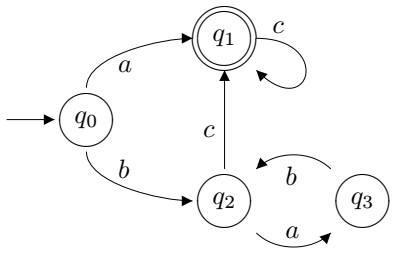
Úkol 2.10

Je dán konečný automat $\mathcal{A}$ s jazykem $L = L(\mathcal{A})$ a homomorfismy h_1 a h_2.

$h_1(a) = 01$
$h_1(b) = 110$
$h_1(c) = 1$

$h_2(a) = dd$
$h_2(b) = g$
$h_3(c) = dg$

	a	b	c
$\rightarrow q_0$	q_1	q_2	
$\leftarrow q_1$			q_1
q_2	q_3		q_1
q_3		q_2	



Zjistěte jazyky $h_1(L)$ a $h_2(L)$ a vytvořte také konečné automaty pro tyto jazyky (úpravou automatu $\mathcal{A}$). Jazyk automatu $\mathcal{A}$ je $L(\mathcal{A}) = \{ac^i \mid i \geq 0\} \cup \{b(ab)^i cc^j \mid i, j \geq 0\}$.

Dále pro výsledné automaty sestrojte jejich δ-funkci včetně plné specifikace, nezapomeňte na změnu abecedy.

Kapitola 3

Regulární výrazy

3.1 Úpravy regulárních výrazů

Pro operace používané v regulárních výrazech platí podobná pravidla jako pro operace v aritmetických výrazech. Například operátory $\cdot$ a $+$ jsou komutativní, asociativní a distributivní. Prvky $\emptyset$ a ε plní v regulárních výrazech podobnou roli jako v aritmetice nula a jednička. Těchto vlastností lze využít při úpravách složitějších regulárních výrazů.

Příklad 3.1

$\emptyset + a = a$
$\varepsilon \cdot a = a$
$\varepsilon^* = \varepsilon$
$ab + c = c + ab = c + ba$
$a(b + c) = ab + ac$
$ab^* + ab = a(b^* + b) = ab^*$
$ab^* + a(c + d) = a(b^* + c + d)$
$(a^*b^*)^* = (ab^*)^* = \{a, b\}^* = (a + b)^*$
$(a + \varepsilon)^* = a^*$
$(a + \varepsilon) \cdot a^* = a^*$
$bb^* + \varepsilon = b^*$
$a(ba)^*b = ab(ab)^* = (ab)^*ab$
$a + (cb)^*a = (\varepsilon + (cb)^*)\, a = (cb)^*a$
$aba + (ab)^*a = ((ab) + (ab)^*)a = (ab)^*a$

Úkol 3.1

Zjednodušte tyto regulární výrazy:

- $ab^* + a^*(a + \varepsilon)$
- $a^*b(ab)^* + a^*b(ab)^*bb^*$
- $a + bc(bc)^*a$
- $a^* + b(a + \varepsilon) \cdot \emptyset$
- $a^* + a\{a, b\}^*b^*$
- $(ab)^*abaa^* + ab(ab)^*a^*$
- $\{a, b\}^*ba + (ab)^*ba$
- $01(01)^*0 + 0^*(10)^*$
- $01(01)^*0 + 0^*(10)^*\{0, 1\}^*$

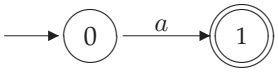
3.2 Sestrojení konečného automatu podle regulárního výrazu

Při sestrojení konečného automatu podle zadaného regulárního výrazu využíváme vztahu mezi prvky regulárního výrazu ($+, \cdot, ^*$) a operacemi nad množinami řetězců ($\cup, \cdot, ^*$). V předchozí kapitole jsme se naučili pracovat s regulárními operacemi (sjednocení, zřetězení a iteraci) a tyto postupy použijeme také zde.

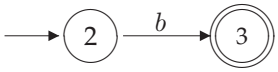
Příklad 3.2

Sestrojíme konečný automat pro regulární výraz $a^*b + (b^*a + bbc)^*$. Postupně vytvoříme konečné automaty pro jednotlivé podvýrazy a zároveň je zjednodušíme.

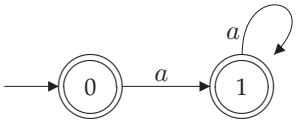
$REG_1 = a$



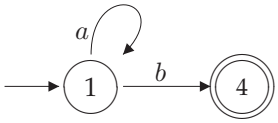
$REG_2 = b$



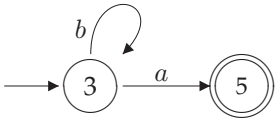
$REG_3 = (REG_1)^* = a^*$



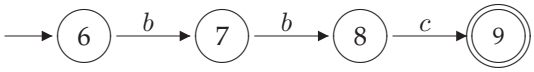
$REG_4 = (REG_3 \cdot REG_2) = a^*b$



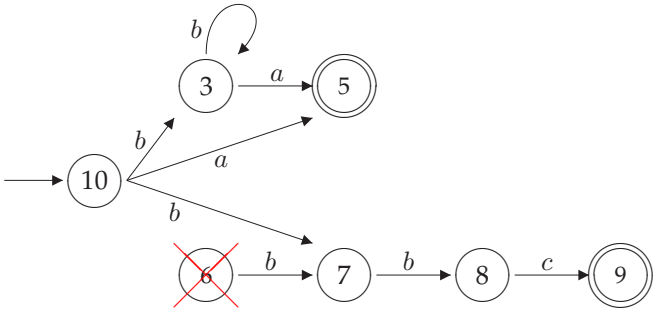
$REG_5 = ((REG_2)^* \cdot REG_1) = b^*a$



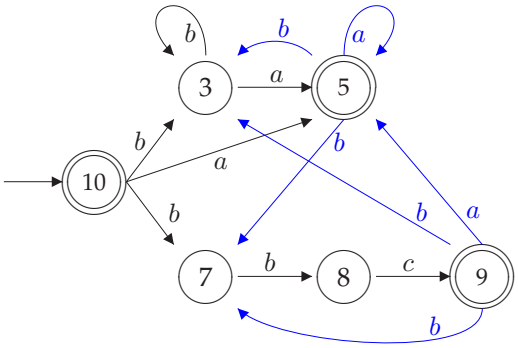
$REG_6 = bbc$



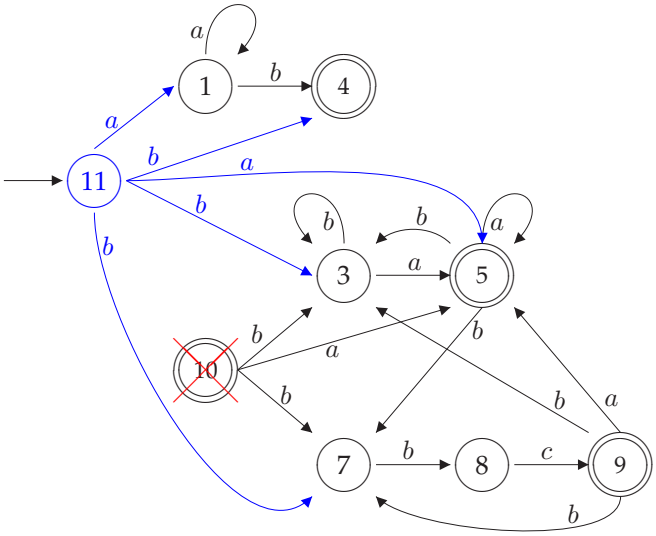
$REG_7 = (REG_5 + REG_6) = b^*a + bbc$



$REG_8 = (REG_7)^* = (b^*a + bbc)^*$



Výsledný automat: $REG_9 = (REG_4 + REG_8) = a^*b + (b^*a + bbc)^*$



	a	b	c
$\rightarrow 11$	1, 5	3, 4, 7	
1	1	4	
3	5	3	
$\leftarrow 4$			
$\leftarrow 5$	5	3, 7	
7		8	
8			9
$\leftarrow 9$	5	3, 7	

Stav 10 je nedosažitelný z počátečního stavu, proto může být odstraněn (v tabulce vpravo se již nenachází).

Úkol 3.2

Sestrojte konečné automaty podle těchto regulárních výrazů:

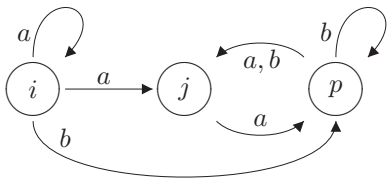
- $(a + b^*) \cdot b^*$
- $a \cdot (b + (ab)^*) + b$
- $(01 + 10)^*10^*$
- $(abc)^* \cdot a \cdot (ba + c)$

3.3 Vytvoření regulárního výrazu podle konečného automatu

Pokud máme zjistit jazyk rozpoznávaný zadaným konečným automatem, obvykle vytváříme regulární výraz (který pak můžeme převést do množinové reprezentace). Postupujeme takto:

1. Pokud je to nutné, přeznačíme stavy (tak, aby byly označeny čísly od 1).
2. Jako bázi rekurze (indukce) použijeme přímé cesty mezi dvěma stavy, na kterých se nenachází žádný další stav (tj. přechody); značíme je R_{ij}^0 a dokážeme je jednoduše zjistit přímo z automatu (v jakékoliv reprezentaci), například:

	a	b
...		
i	i, j	p
j	p	
p	j	j, p
...		



$$R_{ii}^0 = a + \varepsilon \quad R_{jp}^0 = a$$
$$R_{ij}^0 = a \quad R_{pi}^0 = \emptyset$$
$$R_{ip}^0 = b \quad R_{pj}^0 = a + b$$
$$R_{ji}^0 = \emptyset \quad R_{pp}^0 = b + \varepsilon$$
$$R_{jj}^0 = \varepsilon$$

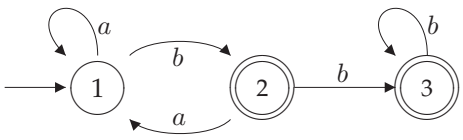
3. Indukcí budeme postupně cesty prodlužovat (spojovat). Vytváříme množiny R_{ij}^k – je to regulární výraz odpovídající cestě ze stavu i do stavu j vedoucí pouze přes stavy označené maximálně číslem k. Postupujeme podle vzorce:

$$R_{ij}^k = R_{ij}^{k-1} + \left(R_{ik}^{k-1} \cdot (R_{kk}^{k-1})^* \cdot R_{kj}^{k-1} \right)$$

   Používáme výhradně množiny zjištěné v předchozím kroku $(k-1)$.
4. Postupujeme až k množinám s horním indexem rovným počtu stavů (tj. v posledním kroku vytvoříme množiny pro cesty mezi stavy, které vedou přes jakékoliv stavy bez omezení). V posledním kroku není třeba vypracovat množiny pro všechny kombinace stavů (dolní index), ale pouze pro cesty z počátečního stavu do koncových stavů.
5. Výsledný regulární výraz je součet (tedy sjednocení) regulárních výrazů odpovídajících cestám z počátečního stavu do jednotlivých koncových stavů.

Příklad 3.3

Zjistíme regulární výraz odpovídající jazyku rozpoznávanému následujícím automatem:



	a	b
$\rightarrow 1$	1	2
$\leftarrow 2$	1	3
$\leftarrow 3$		3

Nejdřív vytvoříme množiny R_{ij}^0 – regulární výrazy pro nejkratší cesty bez mezistavů.

$R^0_{11} = a + \varepsilon$ $R^0_{21} = a$ $R^0_{31} = \emptyset$
$R^0_{12} = b$ $R^0_{22} = \varepsilon$ $R^0_{32} = \emptyset$
$R^0_{13} = \emptyset$ $R^0_{23} = b$ $R^0_{33} = b + \varepsilon$

Horní index znamená maximální číslo stavu, přes který může vést cesta mezi stavy uvedenými v dolním indexu (omezení z horního indexu se netýká okrajových stavů cesty). Například R^2_{13} je regulární výraz odpovídající cestě ze stavu 1 do stavu 3 takové, že vede přes stavy s číslem maximálně 2, tedy přes stavy 1 a 2. Horní index budeme postupně zvyšovat.

V prvním kroku se setkáme se vztahem $(a + \varepsilon)^* = a^*$ a dále $(a + \varepsilon) \cdot a^* = a^*$.

$R^1_{11} = (a+\varepsilon) + (a+\varepsilon) \cdot (a+\varepsilon)^* \cdot (a+\varepsilon) =$ $R^1_{22} = \varepsilon + a \cdot (a + \varepsilon)^* \cdot b = \varepsilon + aa^*b$
$= a + \varepsilon + a^* = a^*$ $R^1_{23} = b + a \cdot (a + \varepsilon)^* \cdot \emptyset = b$
$R^1_{12} = b + (a+\varepsilon) \cdot (a+\varepsilon)^* \cdot b = b + a^*b = a^*b$ $R^1_{31} = \emptyset + \emptyset \cdot (a + \varepsilon)^* \cdot (a + \varepsilon) = \emptyset$
$R^1_{13} = \emptyset + (a + \varepsilon) \cdot (a + \varepsilon)^* \cdot \emptyset = \emptyset$ $R^1_{32} = \emptyset + \emptyset \cdot (a + \varepsilon)^* \cdot b = \emptyset$
$R^1_{21} = a + a \cdot (a+\varepsilon)^* \cdot (a+\varepsilon) = a + aa^* = aa^*$ $R^1_{33} = (b + \varepsilon) + \emptyset \cdot (a + \varepsilon)^* \cdot \emptyset = b + \varepsilon$

V druhém kroku budeme hodně používat vztah $(\varepsilon + aa^*b)^* = (aa^*b)^*$.

$R^2_{11} = a^* + a^*b \cdot (\varepsilon + aa^*b)^* \cdot aa^* = a^* + a^*b(aa^*b)^*aa^*$
$R^2_{12} = a^*b + a^*b \cdot (\varepsilon + aa^*b)^* \cdot (\varepsilon + aa^*b) = a^*b + a^*b(aa^*b)^* = a^*b(aa^*b)^*$
$R^2_{13} = \emptyset + a^*b \cdot (\varepsilon + aa^*b)^* \cdot b = a^*b(aa^*b)^*b$
$R^2_{21} = aa^* + (\varepsilon + aa^*b) \cdot (\varepsilon + aa^*b)^* \cdot aa^* = (aa^*b)^*aa^*$
$R^2_{22} = \varepsilon + aa^*b + (\varepsilon + aa^*b) \cdot (\varepsilon + aa^*b)^* \cdot (\varepsilon + aa^*b) = (aa^*b)^*$
$R^2_{23} = b + (\varepsilon + aa^*b) \cdot (\varepsilon + aa^*b)^* \cdot b = (aa^*b)^*b$
$R^2_{31} = \emptyset + \emptyset \cdot (\varepsilon + aa^*b)^* \cdot aa^* = \emptyset$
$R^2_{32} = \emptyset + \emptyset \cdot (\varepsilon + aa^*b)^* \cdot (\varepsilon + aa^*b) = \emptyset$
$R^2_{33} = b + \varepsilon + \emptyset \cdot (\varepsilon + aa^*b)^* \cdot b = b + \varepsilon$

Protože máme pouze tři stavy, třetí krok je poslední. Zde není třeba vypočítávat výrazy přes všechny dvojice stavů. Potřebujeme pouze ty cesty, které vedou z počátečního stavu do některého koncového. V automatu jsou dva koncové stavy, dostaneme tedy dva regulární výrazy.

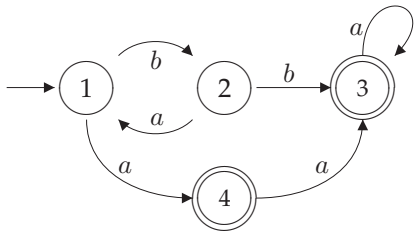
$R^3_{12} = a^*b(aa^*b)^* + a^*b(aa^*b)^*b \cdot (b + \varepsilon)^* \cdot \emptyset = a^*b(aa^*b)^*$
$R^3_{13} = a^*b(aa^*b)^*b + a^*b(aa^*b)^*b \cdot (b + \varepsilon)^* \cdot (b + \varepsilon) = a^*b(aa^*b)^*b + a^*b(aa^*b)^*b \cdot b^* =$
$= a^*b(aa^*b)^*b \cdot (\varepsilon + b^*) = a^*b(aa^*b)^*bb^*$

Výsledkem je regulární výraz odpovídající jazyku rozpoznávanému daným automatem:

$$\begin{aligned} R &= R^3_{12} + R^3_{13} = \\ &= a^*b(aa^*b)^* + a^*b(aa^*b)^*bb^* = a^*b(aa^*b)^* \cdot (\varepsilon + bb^*) = a^*b(aa^*b)^*b^* \end{aligned}$$

Příklad 3.4

Zjistíme regulární výraz odpovídající jazyku rozpoznávanému následujícím automatem:



	a	b
$\rightarrow 1$	4	2
2		3
$\leftarrow 3$	3	
$\leftarrow 4$	3	

Nejdřív vytvoříme množiny R_{ij}^0:

$$R_{11}^0 = \varepsilon \quad R_{21}^0 = a \quad R_{31}^0 = \emptyset \quad R_{41}^0 = \emptyset$$
$$R_{12}^0 = b \quad R_{22}^0 = \varepsilon \quad R_{32}^0 = \emptyset \quad R_{42}^0 = \emptyset$$
$$R_{13}^0 = \emptyset \quad R_{23}^0 = b \quad R_{33}^0 = a + \varepsilon \quad R_{43}^0 = a$$
$$R_{14}^0 = a \quad R_{24}^0 = \emptyset \quad R_{34}^0 = \emptyset \quad R_{44}^0 = \varepsilon$$

Ve čtyřech krocích (číslo kroku je v horním indexu) zjistíme další množiny:

$$R_{11}^1 = \varepsilon + \varepsilon \cdot \varepsilon^* \cdot \varepsilon = \varepsilon \quad R_{21}^1 = a + a \cdot \varepsilon = a \quad R_{31}^1 = \emptyset \quad R_{41}^1 = \emptyset$$
$$R_{12}^1 = b + \varepsilon \cdot \varepsilon^* \cdot b = b \quad R_{22}^1 = \varepsilon + a\varepsilon b = \varepsilon + ab \quad R_{32}^1 = \emptyset \quad R_{42}^1 = \emptyset$$
$$R_{13}^1 = \emptyset + \varepsilon \cdot \varepsilon^* \cdot \emptyset = \emptyset \quad R_{23}^1 = b + \emptyset = b \quad R_{33}^1 = a + \varepsilon + \emptyset = a + \varepsilon \quad R_{43}^1 = a + \emptyset = a$$
$$R_{14}^1 = a + \varepsilon \cdot a = a \quad R_{24}^1 = \emptyset + a\varepsilon a = aa \quad R_{34}^1 = \emptyset \quad R_{44}^1 = \varepsilon + \emptyset = \varepsilon$$

$$R_{11}^2 = \varepsilon + b(ab)^* a = (ba)^* \quad R_{31}^2 = \emptyset + \emptyset = \emptyset$$
$$R_{12}^2 = b + b(ab)^*(\varepsilon + ab) = b(ab)^* \quad R_{32}^2 = \emptyset$$
$$R_{13}^2 = \emptyset + b(ab)^* b = b(ab)^* b \quad R_{33}^2 = a + \varepsilon + \emptyset = a + \varepsilon$$
$$R_{14}^2 = a + b(ab)^* aa = a + ba(ba)^* a = (ba)^* a \quad R_{34}^2 = \emptyset$$

$$R_{21}^2 = a + (\varepsilon + ab)(ab)^* a = (ab)^* a \quad R_{41}^2 = \emptyset$$
$$R_{22}^2 = (ab)^* \quad R_{42}^2 = \emptyset$$
$$R_{23}^2 = b + (\varepsilon + ab)(ab)^* b = (ab)^* b \quad R_{43}^2 = a + \emptyset = a$$
$$R_{24}^2 = aa + (\varepsilon + ab)(ab)^* aa = (ab)^* aa \quad R_{44}^2 = \varepsilon + \emptyset = \varepsilon$$

$$R_{11}^3 = (ba)^* + \emptyset = (ba)^* \quad R_{31}^3 = \emptyset$$
$$R_{12}^3 = b(ab)^* + \emptyset = b(ab)^* \quad R_{32}^3 = \emptyset$$
$$R_{13}^3 = b(ab)^* b + b(ab)^* ba^*(a + \varepsilon) = b(ab)^* ba^* \quad R_{33}^3 = a + \varepsilon + (a + \varepsilon)a^*(a + \varepsilon) = a^*$$
$$R_{14}^3 = (ba)^* a + \emptyset = (ba)^* a \quad R_{34}^3 = \emptyset$$

$$R_{21}^3 = (ab)^* a + \emptyset = (ab)^* a \quad R_{41}^3 = \emptyset$$
$$R_{22}^3 = (ab)^* + \emptyset = (ab)^* \quad R_{42}^3 = \emptyset$$
$$R_{23}^3 = (ab)^* b + (ab)^* ba^*(a + \varepsilon) = (ab)^* ba^* \quad R_{43}^3 = a + aa^*(a + \varepsilon) = aa^*$$
$$R_{24}^3 = (ab)^* aa + \emptyset = (ab)^* aa \quad R_{44}^3 = \varepsilon$$

Ve čtvrtém kroku nás zajímají pouze cesty vedoucí z počátečního stavu do koncových stavů:

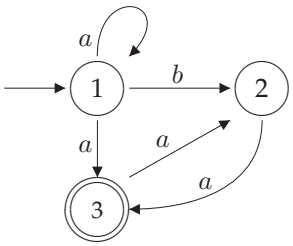
$R_{13}^4 = b(ab)^*ba^* + (ba)^*a \cdot \varepsilon^* \cdot aa^* = b(ab)^*ba^* + (ba)^*aaa^* = (ba)^*bba^* + (ba)^*aaa^*$
$R_{14}^4 = (ba)^*a + (ba)^*a \cdot \varepsilon^* \cdot \varepsilon = (ba)^*a$

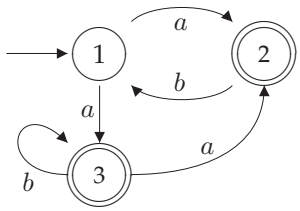
Regulární výraz odpovídající jazyku zadaného automatu je

$$\begin{aligned} R &= R_{13}^4 + R_{14}^4 = \\ &= (ba)^*bba^* + (ba)^*aaa^* + (ba)^*a = (ba)^*(bba^* + aaa^* + a) \end{aligned}$$

Úkol 3.3

Zjistěte regulární výraz odpovídající jazyku následujících konečných automatů:





3.4 Minimalizace a normování konečného automatu

3.4.1 Minimální konečný automat

Minimalizace konečného automatu je snížení počtu stavů při rozpoznávání téhož jazyka, a to na úroveň absolutně nezbytnou vzhledem k rozpoznávanému jazyku. Účelem může být samotné snížení počtu stavů (například z důvodu snadnějšího naprogramování či snížení složitosti automatu) a nebo zajištění porovnatelnosti dvou automatů.

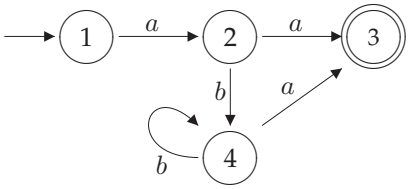
Při minimalizaci automatu $\mathcal{A}$ s množinou stavů $1, 2, \ldots, n$ postupujeme takto:

- automat $\mathcal{A}$ by měl být deterministický; pokud není, převedeme ho do deterministického tvaru,
- podle automatu $\mathcal{A}$ s n stavy vytvoříme n konečných automatů $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$, které jsou určeny téměř stejně (včetně přechodové funkce a koncových stavů) jako automat $\mathcal{A}$, liší se pouze koncovým stavem – automat $\mathcal{A}_i$ má počáteční stav i ($1 \leq i \leq n$),

- postupem popsaným v předchozí kapitole zjistíme regulární výrazy odpovídající všem automatům $\mathcal{A}_i$ vytvořeným v předchozím bodu, zde vlastně zjišťujeme nejdůležitější charakteristiku cesty v grafu automatu z uzlu (stavu) i do koncových stavů,
- regulární výrazy vhodně upravíme, aby byly snadno porovnatelné,
- pokud pro některé dva automaty $\mathcal{A}_i$ a $\mathcal{A}_j$ ($i \neq j$) vychází stejný regulární výraz, znamená to, že stavy i a j jsou rovnocenné (ekvivalentní) v základní charakteristice – na cestě z těchto dvou stavů jsou rozpoznávána tatáž slova (a žádná jiná, jde o ekvivalenci), proto jeden z nich vlastně můžeme vyřadit, a to následovně (chceme odstranit stav j a jeho funkci nahradit stavem i):
  - nejdřív všechny přechody směřující do stavu j přesměrujeme do stavu i,
  - pak můžeme stav j odstranit včetně všech přechodů, které z něho vycházejí.

Příklad 3.5

Minimalizujeme následující konečný automat:



	a	b
$\rightarrow 1$	2	
2	3	4
$\leftarrow 3$		
4	3	4

Automat je deterministický, proto můžeme začít se zjišťováním regulárních výrazů.

$R^0_{11} = \varepsilon$ $R^0_{12} = a$ $R^0_{13} = \emptyset$ $R^0_{14} = \emptyset$
$R^0_{21} = \emptyset$ $R^0_{22} = \varepsilon$ $R^0_{23} = a$ $R^0_{24} = b$
$R^0_{31} = \emptyset$ $R^0_{32} = \emptyset$ $R^0_{33} = \varepsilon$ $R^0_{34} = \emptyset$
$R^0_{41} = \emptyset$ $R^0_{42} = \emptyset$ $R^0_{43} = a$ $R^0_{44} = b + \varepsilon$

$R^1_{11} = \varepsilon$ $R^1_{12} = a$ $R^1_{13} = \emptyset$ $R^1_{14} = \emptyset$
$R^1_{21} = \emptyset$ $R^1_{22} = \varepsilon$ $R^1_{23} = a$ $R^1_{24} = b$
$R^1_{31} = \emptyset$ $R^1_{32} = \emptyset$ $R^1_{33} = \varepsilon$ $R^1_{34} = \emptyset$
$R^1_{41} = \emptyset$ $R^1_{42} = \emptyset$ $R^1_{43} = a$ $R^1_{44} = b + \varepsilon$

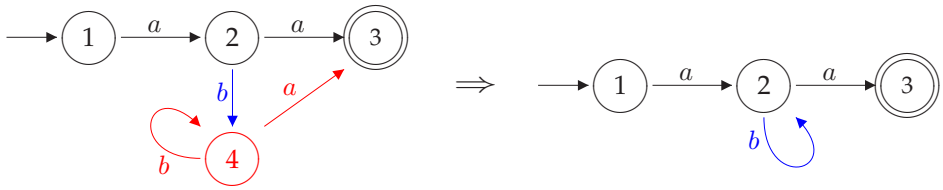
$R^2_{11} = \varepsilon$ $R^2_{12} = a$ $R^2_{13} = aa$ $R^2_{14} = ab$
$R^2_{21} = \emptyset$ $R^2_{22} = \varepsilon$ $R^2_{23} = a$ $R^2_{24} = b$
$R^2_{31} = \emptyset$ $R^2_{32} = \emptyset$ $R^2_{33} = \varepsilon$ $R^2_{34} = \emptyset$
$R^2_{41} = \emptyset$ $R^2_{42} = \emptyset$ $R^2_{43} = a$ $R^2_{44} = b + \varepsilon$

$R^3_{11} = \varepsilon$ $R^3_{12} = a$ $R^3_{13} = aa$ $R^3_{14} = ab$
$R^3_{21} = \emptyset$ $R^3_{22} = \varepsilon$ $R^3_{23} = a$ $R^3_{24} = b$
$R^3_{31} = \emptyset$ $R^3_{32} = \emptyset$ $R^3_{33} = \varepsilon$ $R^3_{34} = \emptyset$
$R^3_{41} = \emptyset$ $R^3_{42} = \emptyset$ $R^3_{43} = a$ $R^3_{44} = b + \varepsilon$

Zajímají nás pouze cesty vedoucí z jednotlivých stavů do koncových stavů (což je jen stav 3), proto zjistíme pouze tyto regulární výrazy:

$R^4_{13} = aa + abb^*a = ab^*a$
$R^4_{23} = a + bb^*a = b^*a$
$R^4_{33} = \varepsilon$
$R^4_{43} = a + (b + \varepsilon)b^*a = a + b^*a = b^*a$

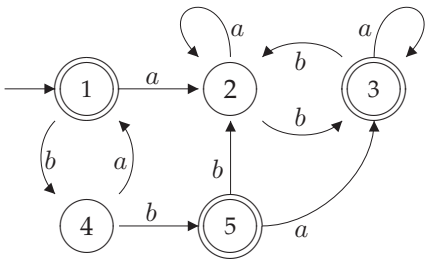
Při porovnání zjistíme, že výrazy R^4_{23} a R^4_{43} se rovnají, proto jeden z nich můžeme odstranit. Je jedno, který zvolíme, jen nesmíme zapomenout všechny přechody vedoucí do rušeného stavu přesměrovat do ekvivalentního stavu.



	a	b
$\rightarrow 1$	2	
2	3	2
$\leftarrow 3$		

Příklad 3.6

Minimalizujeme následující konečný automat (počáteční stav je prvkem množiny koncových stavů):



	a	b
$\leftrightarrow 1$	2	4
2	2	3
$\leftarrow 3$	3	2
4	1	5
$\leftarrow 5$	3	2

Protože v automatu je více stavů než u předchozího a výpis všech pomocných množin by byl nepřehledný, vypíšeme pouze ty množiny, které nejsou prázdné (nemají hodnotu $\emptyset$).

$R^0_{11} = \varepsilon$ $R^0_{12} = a$ $R^0_{14} = b$ $R^0_{22} = a + \varepsilon$ $R^0_{23} = b$ $R^0_{32} = b$ $R^0_{33} = a + \varepsilon$ $R^0_{41} = a$ $R^0_{44} = \varepsilon$ $R^0_{45} = b$ $R^0_{52} = b$ $R^0_{53} = a$ $R^0_{55} = \varepsilon$

$R^1_{11} = \varepsilon$ $R^1_{12} = a$ $R^1_{14} = b$ $R^1_{22} = a + \varepsilon$ $R^1_{23} = b$ $R^1_{32} = b$ $R^1_{33} = a + \varepsilon$ $R^1_{41} = a$ $R^1_{42} = aa$ $R^1_{44} = \varepsilon + ab$ $R^1_{45} = b$ $R^1_{52} = b$ $R^1_{53} = a$ $R^1_{55} = \varepsilon$

$R^2_{11} = \varepsilon$ $R^2_{22} = a^*$ $R^2_{41} = a$ $R^2_{45} = b$
$R^2_{12} = aa^*$ $R^2_{23} = a^*b$ $R^2_{42} = aaa^*$ $R^2_{52} = ba^*$
$R^2_{13} = aa^*b$ $R^2_{32} = ba^*$ $R^2_{43} = aaa^*b$ $R^2_{53} = a + ba^*b$
$R^2_{14} = b$ $R^2_{33} = a + \varepsilon + ba^*b$ $R^2_{44} = \varepsilon + ab$ $R^2_{55} = \varepsilon$

$R^3_{11} = \varepsilon$ $R^3_{41} = a$
$R^3_{12} = aa^* + aa^*(a + ba^*b)^*ba^*$ $R^3_{42} = aaa^* + aaa^*b(a + ba^*b)^*ba^*$
$R^3_{13} = aa^*b(a + ba^*b)^*$ $R^3_{43} = aaa^*b(a + ba^*b)^*$
$R^3_{14} = b$ $R^3_{44} = \varepsilon + ab$
$R^3_{22} = a^* + a^*b(a + ba^*b)^*ba^*$ $R^3_{45} = b$
$R^3_{23} = a^*b(a + ba^*b)^*$ $R^3_{52} = ba^*$
$R^3_{32} = (a + ba^*b)^*ba^*$ $R^3_{53} = (a + ba^*b)(a + ba^*b)^*$
$R^3_{33} = (a + ba^*b)^*$ $R^3_{55} = \varepsilon$

$R^4_{11} = \varepsilon + b(ab)^*a = (ba)^*$
$R^4_{12} = aa^* + aa^*(a + ba^*b)^*ba^* + b(ab)^*(aaa^* + aaa^*b(a + ba^*b)^*ba^*)$
$R^4_{13} = aa^*b(a + ba^*b)^* + b(ab)^*aaa^*b(a + ba^*b)^* = (\varepsilon + b(ab)^*a)(aa^*b(a + ba^*b)^*) =$
$= (ba)^*aa^*b(a + ba^*b)^*$
$R^4_{14} = b$
$R^4_{15} = b(ab)^*b = (ba)^*bb$
$R^4_{22} = a^* + a^*b(a + ba^*b)^*ba^*$
$R^4_{23} = a^*b(a + ba^*b)^*$
$R^4_{32} = (a + ba^*b)^*ba^*$
$R^4_{33} = (a + ba^*b)^*$
$R^4_{41} = (ab)^*a$
$R^4_{42} = (ab)^*(aaa^* + aaa^*b(a + ba^*b)^*ba^*)$
$R^4_{43} = (ab)^*aaa^*b(a + ba^*b)^*$
$R^4_{44} = (ab)^*$
$R^4_{45} = (ab)^*b$
$R^4_{52} = ba^*$
$R^4_{53} = (a + ba^*b)(a + ba^*b)^*$
$R^4_{55} = \varepsilon$

V posledním kroku se zaměříme pouze na cesty končící v některém koncovém stavu (to jsou stavy 1, 3 a 5), vypíšeme i prázdné množiny.

$R^5_{11} = (ba)^*$ $R^5_{21} = \emptyset$
$R^5_{13} = (ba)^*aa^*b(a + ba^*b)^* +$ $R^5_{23} = a^*b(a + ba^*b)^*$
$+(ba)^*bb(a + ba^*b)(a + ba^*b)^*$ $R^5_{25} = \emptyset$
$R^5_{15} = (ba)^*bb$

$$R^5_{31} = \emptyset$$
$$R^5_{33} = (a + ba^*b)^*$$
$$R^5_{35} = \emptyset$$

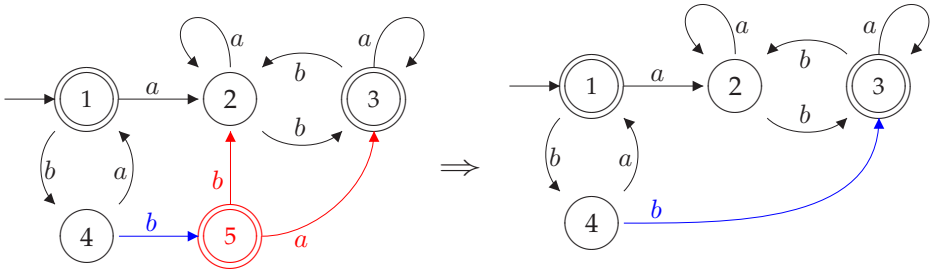
$$R^5_{41} = (ab)^*a$$
$$R^5_{43} = (ab)^*aaa^*b(a + ba^*b)^* + \\ +(ab)^*b(a + ba^*b)(a + ba^*b)^*$$
$$R^5_{45} = (ab)^*b$$

$$R^5_{51} = \emptyset$$
$$R^5_{53} = (a + ba^*b)(a + ba^*b)^*$$
$$R^5_{55} = \varepsilon$$

Zbývá zjistit jazyky jednotlivých automatů $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_5$ (odlišujících se pouze svým počátečním stavem – $1, 2, \ldots, 5$):

$$\begin{aligned} L(\mathcal{A}_1) = L(\mathcal{A}) &= R^5_{11} + R^5_{13} + R^5_{15} = \\ &= (ba)^* + (ba)^*\Big(aa^*b(a + ba^*b)^* + bb(a + ba^*b)(a + ba^*b)^* + bb\Big) = \\ &= (ba)^* + (ba)^*\Big(aa^*b(a + ba^*b)^* + bb(a + ba^*b)^*\Big) = \\ &= (ba)^* + (ba)^*(aa^*b + bb)(a + ba^*b)^* \end{aligned}$$
$$L(\mathcal{A}_2) = R^5_{21} + R^5_{23} + R^5_{25} = a^*b(a + ba^*b)^*$$
$$L(\mathcal{A}_3) = R^5_{31} + R^5_{33} + R^5_{35} = (a + ba^*b)^*$$
$$\begin{aligned} L(\mathcal{A}_4) = R^5_{41} + R^5_{43} + R^5_{45} &= (ab)^*a + (ab)^*aaa^*b(a + ba^*b)^* + (ab)^*b(a + ba^*b)^* = \\ &= (ab)^*a + (ab)^*(aaa^*b + b)(a + ba^*b)^* \end{aligned}$$
$$L(\mathcal{A}_5) = R^5_{51} + R^5_{53} + R^5_{55} = (a + ba^*b)^*$$

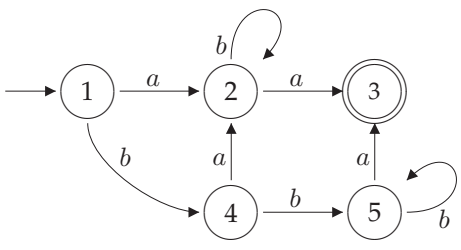
Jazyky $L(\mathcal{A}_3)$ a $L(\mathcal{A}_5)$ jsou stejné, proto jeden z těchto stavů můžeme odstranit.



	a	b
$\leftrightarrow 1$	2	4
2	2	3
$\leftarrow 3$	3	2
4	1	3

Úkol 3.4

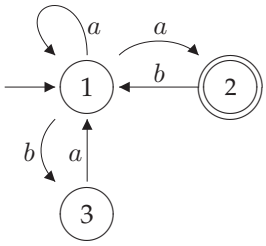
1. Minimalizujte následující konečný automat:



	a	b
$\rightarrow 1$	2	4
2	3	2
$\leftarrow 3$		
4	2	2
5	3	5

2. Následující (nedeterministický) konečný automat

   - převeďte na deterministický (po převodu a odstranění nepotřebných stavů by měl mít čtyři stavy),
   - vhodně přeznačte stavy (čísla od 1) a určete jazyk tohoto konečného automatu,
   - minimalizujte ho.



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

   Při úpravách regulárních výrazů lze využít například těchto vztahů:

   - $b(ab)^* = (ba)^*b$
   - $(ab)^*a = a(ba)^*$
   - $b(ab)^*a = (ba)^*ba$
   - $\varepsilon + b(ab)^*a = \varepsilon + (ba)^*ba = (ba)^*$
   - $\varepsilon + (ba)^*aa^*b\,((ba)^*aa^*b)^* = ((ba)^*aa^*b)^*$

   Pozor, sice platí rovnost $\varepsilon + b^*a = b^*$, ale například regulární výrazy $\varepsilon + b^*ac^*$ a b^*c^* nejsou ekvivalentní!

3.4.2 Normovaný konečný automat

Účelem normování (čehokoliv) je uvést daný objekt (prvek, automat, atd.) do stavu s určitými žádanými vlastnostmi, a to za účelem snadnějšího provádění dalších operací s tímto objektem (například porovnávání nebo programování).

Když normujeme konečný automat, chceme, aby měl vhodně označené stavy (tedy normujeme označení stavů) – stavy by měly být pojmenovány čísly od 1 a navíc by jejich označení mělo být seřazeno.

Kromě vlastního označení mají stavy jen jedinou další charakteristiku – regulární výraz odpovídající cestě z daného stavu do koncových stavů. Tuto charakteristiku použijeme pro vytvoření metriky při řazení označení stavů. Postupujeme takto:

- minimalizujeme konečný automat,
- pro každý stav určíme regulární výraz cesty z tohoto stavu do koncových stavů,
- v regulárních výrazech najdeme nejmenší slova,

- stavy porovnáváme takto:
  1. stav, jehož regulární výraz má kratší nejmenší slovo, dostane vyšší index než stav s delším nejmenším slovem, například stav s nejmenším slovem ab bude mít vyšší číslo v označení než stav s nejmenším slovem bba,
  2. pokud dva stavy mají nejmenší slova stejně dlouhá, pak zvolíme navíc další metriku – abecedu, například stav s nejmenším slovem ab bude mít vyšší číslo v označení než stav s nejmenším slovem bb,
  3. pokud se nejmenší slova dvou stavů shodují (tj. předchozí metriky selhávají), pak v regulárních výrazech těchto stavů hledáme druhé nejkratší slovo a to porovnáváme, pokud se opět shodují, hledáme třetí, atd., dokud nenajdeme rozdíl.

Přestože poslední uvedený způsob porovnání může trvat hodně dlouho, u minimalizovaného automat je vždy konečný, protože regulární výrazy pro žádné dva stavy nejsou ekvivalentní (kdyby byly ekvivalentní, jeden z nich bychom vyřadili při minimalizaci).

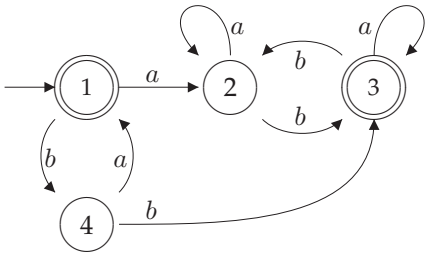
Příklad 3.7

Seřadíme následující regulární výrazy podle výše uvedených kritérií.

Reg. výraz	*Nejkratší slova*	*Zjištěné pořadí*	*Přidělený index*
$ba(ba)^*$	$ba, baba, bababa, \ldots$	8	3
$aabb^*$	$aab, aabb, aabbb, \ldots$	10	1
$ab(ab)^*$	$ab, abab, ababab, \ldots$	6	5
$aaba^*$	$aab, aaba, aabaa, \ldots$	9	2
$ab(ba)^*$	$ab, abba, abbaba, \ldots$	7	4
b^*aa^*b	$ab, aab, bab, aaab, baab, bbab, \ldots$	5	6
$b^*(aa)^*b$	$b, bb, aab, bbb, baab, aaaab, \ldots$	3	8
$a^*b + b^*a$	$b, a, ab, ba, aab, aab, bba, \ldots$	2	9
$ab + ba$	ab, ba	4	7
$(ab + ba)^*$	$\varepsilon, ab, ba, abab, abba, baab, baba, \ldots$	1	10

Příklad 3.8

Znormujeme automat, který jsme získali minimalizací v příkladu 3.6 na straně 42.



	a	b
$\leftrightarrow 1$	2	4
2	2	3
$\leftarrow 3$	3	2
4	1	3

Využijeme také regulární výrazy pro jednotlivé stavy:

$L(\mathcal{A}_1) = L(\mathcal{A}) = R^5_{11} + R^5_{13} + R^5_{15} = (ba)^* + (ba)^*(aa^*b + bb)(a + ba^*b)^*$

$L(\mathcal{A}_2) = R^5_{21} + R^5_{23} + R^5_{25} = a^*b(a + ba^*b)^*$

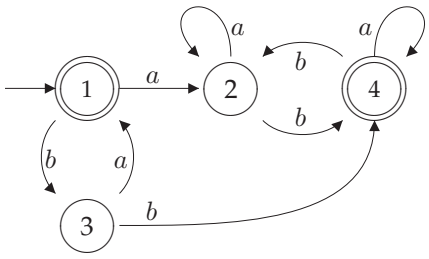
$L(\mathcal{A}_3) = R^5_{31} + R^5_{33} + R^5_{35} = (a + ba^*b)^*$

$L(\mathcal{A}_4) = R^5_{41} + R^5_{43} + R^5_{45} = (ab)^*a + (ab)^*(aaa^*b + b)(a + ba^*b)^*$

Z těchto regulárních výrazů vybereme nejkratší slova:

Stav	*Nejkratší slova*	*Nové označení stavu*
1	$ab, bb, ba, aba, \ldots$	1
2	$b, ba, ab, \ldots$	2
3	$\varepsilon, a, bb, \ldots$	4
4	$a, b, \ldots$	3

Stavy přeznačíme a vše ostatní zachováme. Po přeznačení dostaneme tento konečný automat:



	a	b
$\leftrightarrow 1$	2	3
2	2	4
$\leftarrow 4$	4	2
3	1	4

Úkol 3.5

1. Setřiďte podle výše zadaných kritérií následujících 16 regulárních výrazů:

   - $a(a + b)a$
   - $a(a + b)^*a$
   - $a(a + b)^*b$
   - $a^*(a + b)a$
   - $a^*(a + b)a^*$
   - $a^*b + a$
   - a^*b
   - b^*a
   - $(a^*b + a)^*$
   - $(a^*b + b)^*$
   - $(b^*a + b)^*$
   - $(b^*a + b)^*$

   Pozor, může se stát, že některé dva regulární výrazy jsou ekvivalentní.

2. Podle konečného automatu *v zadání* příkladu 3.4 na straně 39 dopočtěte zbývající množiny R^4_{ij}, minimalizujte tento automat a normujte ho.

3. Podle konečného automatu *v zadání* příkladu 2.12 na straně 26 vytvořte všechny potřebné množiny R^k_{ij}, minimalizujte (pokud není minimální) a normujte ho.

Kapitola 4

Regulární gramatiky

4.1 Konečný automat podle regulární gramatiky

Pokud podle regulární gramatiky vytváříme konečný automat, tak vlastně tento konečný automat simuluje generování svého vstupu v původní gramatice. Jeden krok automatu skládající se z načtení signálu ze vstupu a přechodu do následujícího stavu odpovídá použití jednoho pravidla v gramatice, a to takového, které generuje stejný terminál (signál).

Postupujeme takto:

- množinu terminálů gramatiky použijeme pro *abecedu* automatu,
- množinu neterminálů použijeme pro *stavy*,
- stav vytvořený ze startovacího symbolu gramatiky bude *počátečním stavem*,
- do množiny stavů také přidáme nový stav (obvykle značený X), který se stane *koncovým*,
- pokud je v jazyce generovaném gramatikou slovo ε, pak počáteční stav automatu bude také patřit do množiny koncových stavů, aby automat rozpoznával slovo ε,
- δ-funkci vytvoříme podle pravidel gramatiky.

Příklad 4.1

Vytvoříme konečný automat rozpoznávající jazyk generovaný touto gramatikou:
$G = (\{S, A, B\},\ \{a, b\},\ P,\ S)$ s těmito pravidly:
$S \to aS \mid bA$
$A \to aA \mid aB \mid a$
$B \to bB \mid b$

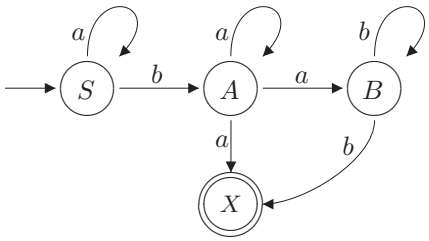
V gramatice není žádné ε-pravidlo (tj. pravidlo, na jehož pravé straně je řetězec ε), proto počáteční stav nebude patřit do množiny koncových stavů (tato množina bude jed-

noprvková). Jako abecedu použijeme množinu terminálů, stavy vytvoříme z neterminálů a přidáme jeden koncový stav X.

Přechody tvoříme podle pravidel gramatiky – například podle pravidla $A \to aB$ přidáme do δ-funkce $\delta(A, a) \ni B$. Postup je dobře vidět na δ-funkci:

$\mathcal{A} = (\{S, A, B, X\},\ \{a, b\},\ \delta,\ S,\ \{X\})$
$\delta(S, a) = \{S\}$ *podle* $S \to aS$
$\delta(S, b) = \{A\}$ *podle* $S \to bA$
$\delta(A, a) = \{A, B, X\}$ *podle* $A \to aA \mid aB \mid a$
$\delta(B, b) = \{B, X\}$ *podle* $B \to bB \mid b$

Podobně sestrojíme stavový diagram a tabulku přechodů:



	a	b
$\to S$	S	A
A	A, B, X	
B		B, X
$\leftarrow X$		

Jazyk generovaný gramatikou a rozpoznávaný automatem je
$L(G) = L(\mathcal{A}) = a^*ba^*a(\varepsilon + b^*b)$

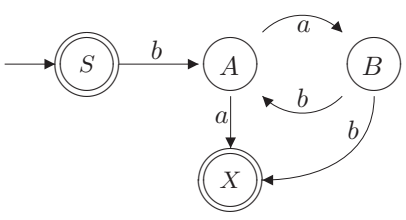
Příklad 4.2

Narozdíl od předchozího příkladu zde máme gramatiku s ε-pravidlem:
$G = (\{S, A, B\},\ \{a, b\},\ P,\ S)$, v množině P jsou pravidla
$S \to bA \mid \varepsilon$
$A \to aB \mid a$
$B \to bA \mid b$

V jazyce generovaném gramatikou je také prázdné slovo ε a toto slovo musí přijímat i konečný automat, který vytvoříme. To lze zajistit pouze tak, že počáteční stav automatu přidáme do množiny koncových stavů.

$\mathcal{A} = (\{S, A, B, X\},\ \{a, b\},\ \delta,\ S,\ \{S, X\})$

$\delta(S, b) = \{A\}$
$\delta(A, a) = \{B, X\}$
$\delta(B, b) = \{A, X\}$



	a	b
$\leftrightarrow S$		A
A	B, X	
B		A, X
$\leftarrow X$		

Jazyk generovaný gramatikou a rozpoznávaný automatem je
$L(G) = L(\mathcal{A}) = \varepsilon + b(ab)^*a + b(ab)^*ab = \varepsilon + b(ab)^*a(\varepsilon + b)$

Úkol 4.1

Podle následujících gramatik sestrojte ekvivalentní konečný automat (tj. rozpoznávající jazyk generovaný touto gramatikou).

$G_1 = (\{S, A, B\},\ \{a, b, c\},\ P,\ S)$
$S \to aS \mid bS \mid cA$
$A \to aA \mid cB \mid a$
$B \to bB \mid cA \mid b$

$G_2 = (\{A, B, C\},\ \{0, 1\},\ P,\ A)$
$A \to 0B \mid 1C \mid \varepsilon$
$B \to 0C \mid 1C$
$C \to 1B \mid 0$

$G_3 = (\{R, Y, Z, W\},\ \{a, b, c\},\ P,\ R)$
$R \to aR \mid bR \mid cZ$
$Y \to cR \mid aY \mid b$
$Z \to aY \mid aW$
$W \to bR \mid b$

$G_4 = (\{S, A, B, C\},\ \{0, 1, 2\},\ P,\ S)$
$S \to 0A \mid 1B \mid 2C \mid \varepsilon$
$A \to 0B \mid 0$
$B \to 1C \mid 1$
$C \to 2A$

4.2 Regulární gramatika podle konečného automatu

Při zjišťování gramatiky generující tentýž jazyk, který je rozpoznáván zadaným konečným automatem, je nejjednodušší obrátit postup, který jsme použili pro vytvoření automatu podle gramatiky.

Když jsme sestrojili konečný automat podle regulární gramatiky, automat měl vždy tyto vlastnosti:

- pokud v jazyce generovaném gramatikou není slovo ε, pak
  - existuje jediný koncový stav (nově přidaný), obvykle pojmenovaný X,
  - ze stavu X nevede žádný přechod (tj. když se výpočet dostane do koncového stavu, nelze dále pokračovat).
- pokud v jazyce generovaném gramatikou je slovo ε, pak
  - kromě nově přidaného koncového stavu X je v množině koncových stavů i počáteční stav,
  - ze stavu X nevede žádný přechod,
  - do počátečního stavu nevede žádný přechod (ekvivalent k faktu, že pokud v gramatice máme pravidlo $S \to \varepsilon$, pak se symbol S nesmí vyskytovat na pravé straně žádného pravidla).

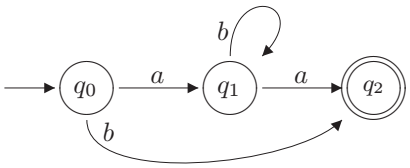
Abychom tedy mohli použít opačný postup k postupu předchozímu, musíme zadaný automat nejdřív upravit do tvaru odpovídajícího výše uvedeným požadavkům. Úprava spočívá v těchto krocích (jejich pořadí je důležité):

1. Jestliže počáteční stav patří do množiny koncových stavů a zároveň do tohoto stavu vede některý přechod:
   - vytvoříme nový počáteční stav (také zařadíme do množiny koncových stavů), do kterého nebudou vést žádné přechody, ale z něho budou vést tytéž přechody jako z původního počátečního stavu,
   - původní počáteční stav zůstane v množině koncových stavů.
2. Pokud je v množině koncových stavů více než jeden stav (případně kromě počátečního stavu, toho se tento bod netýká):
   - vytvoříme nový koncový stav X (nebo jinak označený),
   - ze stavu X nepovede žádný přechod, ale do tohoto stavu povedou tytéž přechody, které vedou do původních koncových stavů,
   - původní koncové stavy sice v automatu necháme, ale vyřadíme je z množiny koncových stavů (netýká se počátečního stavu).
3. Jestliže z koncového stavu vedou přechody, problém vyřešíme stejně jako v předchozím bodě.

Příklad 4.3

Nejdřív si ukážeme jednodušší případ – konečný automat odpovídající uvedeným požadavkům.

Jak vidíme, stav q_2 zde vlastně zastupuje stav X podle předchozího postupu. To znamená, že v gramatice bude množina neterminálů dvouprvková – $\{q_0, q_1\}$, ze stavu q_2 žádný neterminál nevytvoříme a pravidla, která s ním souvisejí, budou sloužit k ukončení generování slova. Můžeme také přejmenovat neterminály.



	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_2	q_1
$\leftarrow q_2$		

$G = (\{q_0, q_1\},\ \{a, b\},\ P,\ q_0)$

$q_0 \rightarrow aq_1 \mid b$

$q_1 \rightarrow a \mid bq_1$

$G = (\{A, B\},\ \{a, b\},\ P,\ A)$

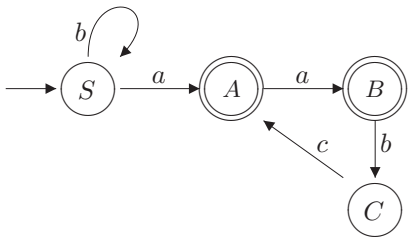
$A \rightarrow aB \mid b$

$B \rightarrow a \mid bB$

Gramatika, kterou jsme vytvořili, generuje jazyk $L(G) = ab^*a + b$, což je také jazyk rozpoznávaný zadaným automatem.

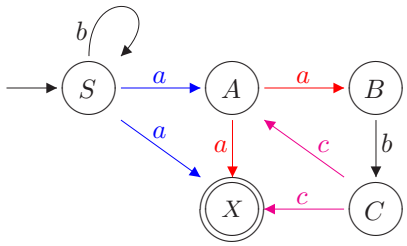
Příklad 4.4

Vytvoříme regulární gramatiku generující jazyk tohoto automatu:



Tento automat má dva koncové stavy, z nichž žádný není počáteční, proto musíme redukovat počet koncových stavů. Navíc z obou koncových stavů vycházejí přechody, to je další důvod pro transformaci.

Vytvoříme stav X, do kterého nasměrujeme všechny přechody mířící do původních koncových stavů. Stav X pak bude jediným koncovým stavem. Po úpravě sestrojíme regulární gramatiku stejně jako u předchozího příkladu.



	a	b	c
$\rightarrow S$	A, X	S	
A	B, X		
B		C	
C			A, X
$\leftarrow X$			

$G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$

$S \rightarrow aA \mid a \mid bS$

$A \rightarrow aB \mid a$

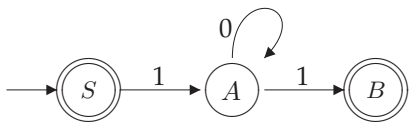
$B \rightarrow bC$

$C \rightarrow cA \mid c$

Jazyk generovaný gramatikou je $L(G) = b^*a(abc)^*(\varepsilon + a)$.

Příklad 4.5

Následující automat nemusíme upravovat, můžeme přímo napsat regulární gramatiku – startovací symbol patří do množiny koncových stavů, ale do něho nevede žádný přechod.



	0	1
$\leftrightarrow S$		A
A	A	B
$\leftarrow B$		

$G = (\{S, A\},\ \{0, 1\},\ P,\ S)$

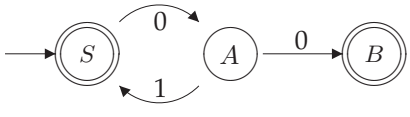
$S \rightarrow 1A \mid \varepsilon$

$A \rightarrow 0A \mid 1$

Generovaný (rozpoznávaný) jazyk je $L(G) = 10^*1$.

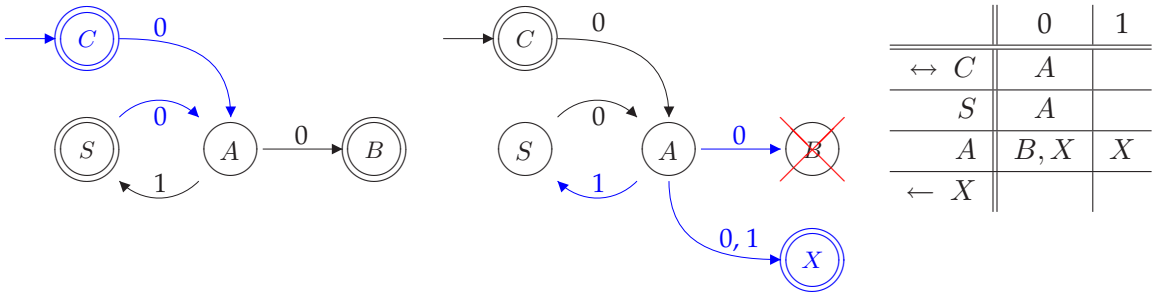
Příklad 4.6

V posledním příkladu si ukážeme převod automatu, jehož počáteční stav je zároveň stavem koncovým a vedou do něho přechody.



Nejdřív automat upravíme. Přidáme nový (počáteční) stav C, který také bude patřit do množiny koncových stavů (aby automat mohl rozpoznat prázdné slovo ε) – tento stav bude

využit pouze na začátku výpočtu. Potom přidáme nový koncový stav X, který využijeme na konci každého výpočtu. Původní koncové stavy (kromě počátečního stavu C vyřadíme z množiny koncových stavů.



	0	1
$\leftrightarrow C$	A	
S	A	
A	B, X	X
$\leftarrow X$		

Ze stavu B nevede cesta do koncového stavu, proto může být odstraněn. Výsledná gramatika je následující:

$G = (\{C, S, A\},\ \{0, 1\},\ P,\ C)$

$C \rightarrow 0A$

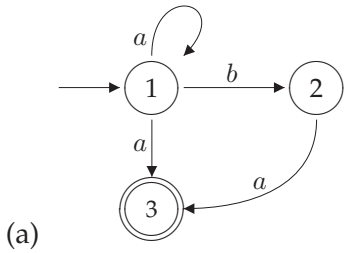
$S \rightarrow 0A$

$A \rightarrow 0B \mid 0 \mid 1$

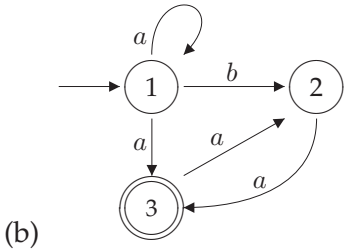
Jazyk generovaný gramatikou G vyjádřený regulárním výrazem je $L(G) = 0(10)^*(0 + 1)$.

Úkol 4.2

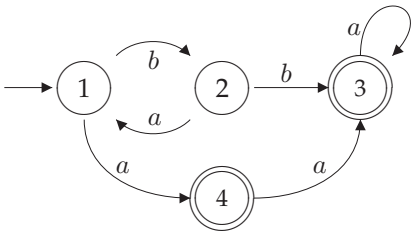
K následujícím konečným automatům vytvořte ekvivalentní regulární gramatiky:



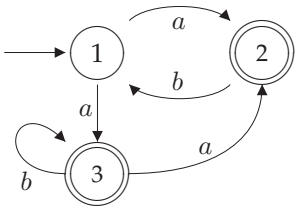
(a)



(b)

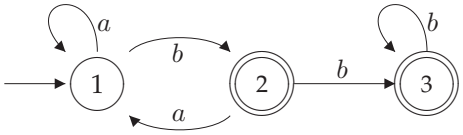


(c)

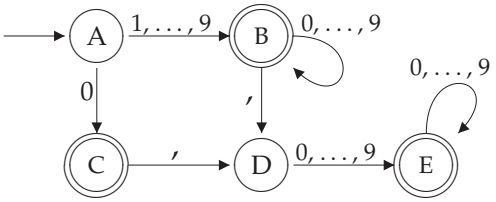


(d)

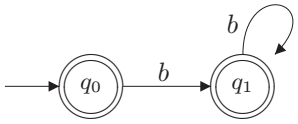
(e)



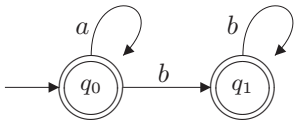
(f)



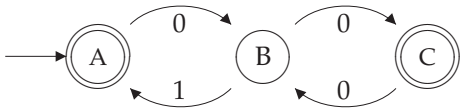
(g)



(h)



(i)



Kapitola 5

Bezkontextové gramatiky

5.1 Sestrojení gramatiky a derivační strom

Bezkontextové gramatiky mají obecnější tvar pravidel než regulární – na levé straně pravidla je vždy jeden neterminál (jako u regulárních), ale na pravé straně je jakýkoliv řetězec skládající se z terminálů a neterminálů (včetně prázdného řetězce).

Když chceme sestrojit bezkontextovou gramatiku pro zadaný jazyk, pokusíme se popsat strukturu tohoto jazyka s využitím rekurze.

Příklad 5.1

Sestrojíme bezkontextovou gramatiku pro jazyk $L = \{a^n b a^n b \mid n \geq 0\}$ a vytvoříme derivaci (odvození) slova $aabaab$.

$G = (\{S, A\},\ \{a, b\},\ P,\ S)$

$S \to Ab$

$A \to aAa \mid b$

Neterminál A generuje téměř celé slovo, až na poslední symbol b. Část slova generovaná tímto neterminálem je symetrická, proto pravidla jsou vlastně lineární (každá lineární gramatika je zároveň bezkontextová, tedy zadání je v tomto směru splněno). Následuje derivace slova $aabaab$:

$S \Rightarrow Ab \Rightarrow aAab \Rightarrow aaAaab \Rightarrow aabaab$

Derivační strom je vlastně graf, který je stromem (má jediný kořen, každý uzel kromě kořene má právě jednoho předka a hrany jsou orientované, i když orientaci neznačíme), tvořený podle pravidel gramatiky použitých v derivaci, podle které je derivační strom vytvořen.

Příklad 5.2

Sestrojíme bezkontextovou gramatiku pro jazyk $L = \{0^n 1 0^{2n} 1^i \mid i, n \geq 1\}$ a vytvoříme derivaci slova 0010000111 a derivační strom k této derivaci.

$G = (\{S, A, B\},\ \{0, 1\},\ P,\ S)$

$S \to AB$

$A \to 0A00 \mid 0100$

$B \to 1B \mid 1$

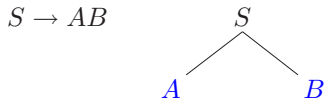
Derivace slova 0010000111 je následující:

$S \Rightarrow AB \Rightarrow 0A00B \Rightarrow 0010000B \Rightarrow 00100001B \Rightarrow 001000011B \Rightarrow 0010000111$

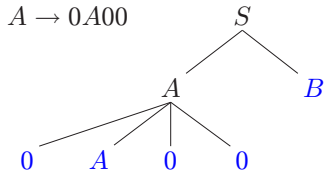
Modře je vyznačena pravá část pravidla, které bylo v daném kroku odvození použito (například v druhém kroku jsme použili pravidlo $A \to 0A00$).

Derivační strom se vždy vztahuje ke konkrétní derivaci. Podle výše uvedené derivace postupně sestrojíme derivační strom takto (jsou uvedeny i mezikroky):

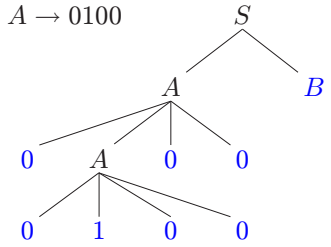
$S \Rightarrow AB$



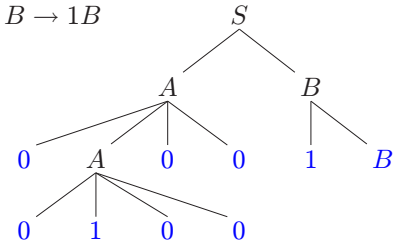
$S \Rightarrow AB \Rightarrow 0A00B$



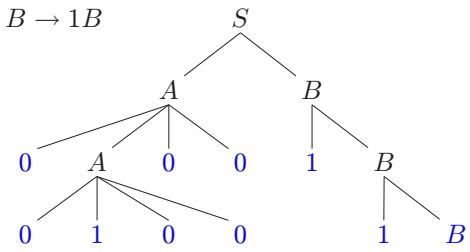
$S \Rightarrow AB \Rightarrow 0A00B \Rightarrow 0010000B$



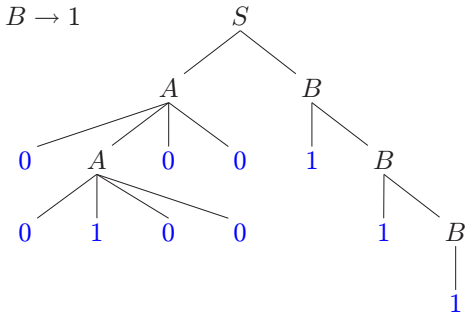
$S \Rightarrow \ldots \Rightarrow 0010000B \Rightarrow 00100001B$



$S \Rightarrow \ldots \Rightarrow 00100001B \Rightarrow 001000011B$



$S \Rightarrow \ldots \Rightarrow 001000011B \Rightarrow 0010000111$



V jednotlivých krocích je kromě samotného stromu uvedeno použité pravidlo a také momentální stav derivace. Modře je vyznačena větná forma, kterou jsme v daném kroku vytvořili. Můžeme si všimnout, že v derivačním stromě vždy jde o obsah listů stromu v daném kroku. Derivační strom dané derivace je poslední v pořadí.

Úkol 5.1

1. Podle gramatiky a uvedené derivace z příkladu 5.1 sestrojte derivační strom.
2. Podle zadané gramatiky vytvořte derivaci slova $ababa$ a k ní derivační strom.

   $G = (\{S, A, B\},\ \{a, b\},\ P,\ S)$

   $S \to bAB \mid aBA \mid \varepsilon$
   $A \to abAab \mid \varepsilon$
   $B \to baBba \mid \varepsilon$
3. Podle gramatiky z předchozího příkladu vytvořte derivaci jakéhokoliv slova o délce alespoň 6 začínajícího symbolem b a k této derivaci sestrojte derivační strom.
4. Sestrojte gramatiky generující následující jazyky. V každém jazyce vyberte jakékoliv slovo o délce alespoň 3 symboly, vytvořte jeho derivaci a sestrojte k ní derivační strom.

   (a) $L_1 = \{(ab)^i c^j (ba)^i \mid i, j \geq 0\}$
   (b) $L_2 = \{0^n 11^n \mid n \geq 3\} \cup \{\varepsilon\}$
   (c) $L_3 = \{0^i 10^i \mid i \geq 1\} \cup \{1^i 01^i \mid i \geq 1\}$
   (d) $L_4 = \{abc, bad, faa, diff\}$ (nezapomeňte, že na pravé straně pravidla může být jakýkoliv řetězec)

5.2 Vlastnosti bezkontextových gramatik

5.2.1 Redukce gramatiky

Účelem redukce gramatiky je snížení počtu neterminálů při zachování generovaného jazyka. Odstraňujeme

- *nadbytečné neterminály* (tj. neterminály, ze kterých nelze vygenerovat žádné terminální slovo),
- *nedostupné symboly* (terminální i neterminální – nelze je vygenerovat ze startovacího symbolu).

Zachováváme toto pořadí – nejdřív nadbytečné a pak teprve nedostupné. Používáme podobný postup jako při redukci množiny stavů konečného automatu.

Příklad 5.3

Redukujeme následující gramatiku:

$G = (\{S, A, B, C, D\},\ \{a, b, c, d\},\ P,\ S)$

$S \to bS \mid bD \mid \varepsilon$

$A \to aCB \mid bAD$

$B \to dB \mid d$

$C \to bCC \mid aAbB$

$D \to bA \mid cDb \mid aS \mid \varepsilon$

Nejdřív odstraníme nadbytečné neterminály. Rekurzívně vytvoříme množinu všech symbolů, které jsou buď terminální a nebo z nich lze vygenerovat terminální řetězec (v případě, že jde o neterminál). Bází rekurze je množina všech terminálů. V dalších krocích přidáváme neterminály z levých stran pravidel, na jejichž pravé straně jsou pouze symboly, které jsme do naší množiny přidali v předchozích krocích (a nebo pro ně existuje ε-pravidlo).

$N_0 = \{a, b, c, d\}$

$N_1 = \{a, b, c, d, S, B, D\}$ (podle pravidel $S \to \varepsilon,\ B \to d,\ D \to \varepsilon$)

$N_2 = \{a, b, c, d, S, B, D\}$ (podle pravidel $S \to bS \mid bD,\ B \to dB,\ D \to cDb \mid aS$)

Protože jsme do množiny N_2 oproti množině N_1 nic dalšího nepřidali, rekurze končí. Množina neterminálů bude $N \cap N_2 = \{S, B, D\}$, vyřadíme neterminály A a C včetně všech pravidel, která je obsahují na levé nebo pravé straně. Gramatika po odstranění nadbytečných symbolů je následující:

$G' = (\{S, B, D\},\ \{a, b, c, d\},\ P',\ S)$

$S \to bS \mid bD \mid \varepsilon$

$B \to dB \mid d$

$D \to cDb \mid aS \mid \varepsilon$

Zbývá odstranit nedostupné symboly. Postupujeme opět rekurzívně. Vytvoříme množinu symbolů, které se nacházejí v nějaké větné formě v derivaci ze startovacího symbolu. Začneme množinou obsahující pouze startovací symbol, v každém kroku přidáváme symboly, které se nacházejí na pravých stranách pravidel přepisujících symboly zařazené v předchozích krocích.

$V_0 = \{S\}$

$V_1 = \{S, b, D\}$ (podle pravidel $S \to bS \mid bD$)

$V_2 = \{S, b, D, c, a\}$ (podle pravidel $D \to cDb \mid aS$)

$V_3 = V_2$

Z postupu vyplývá, že pokud v některém kroku přidáme pouze terminální symboly, v následujícím kroku již nelze nic přidat (protože terminály se nepřepisují žádným pravi-

dlem) a rekurze končí. Je zřejmé, že nedostupný neterminál je jeden (B) a terminál taktéž jeden (d). Redukovaná gramatika (tj. již bez nadbytečných i nedostupných symbolů) je následující:

$G'' = (\{S, D\},\ \{a, b, c\},\ P'',\ S)$

$S \to bS \mid bD \mid \varepsilon$

$D \to cDb \mid aS \mid \varepsilon$

Úkol 5.2

Redukujte tyto gramatiky:

$G_1 = (\{S, A, B, C, D\},\ \{a, b, c, d\},\ P,\ S)$

$S \to aBa \mid bA \mid c \mid ABc$

$A \to aA \mid dD \mid bDa$

$B \to aB \mid bA \mid bS$

$C \to aSc \mid cC \mid c$

$D \to dD \mid aAb$

$G_2 = (\{S, A, B, C, D, E\},\ \{0, 1, 2, 3\},\ P,\ S)$

$S \to AB0 \mid 2E \mid A1 \mid \varepsilon$

$A \to 0A0 \mid 1B \mid 1S$

$B \to 01B \mid 1D \mid EB$

$C \to 2A \mid 1S \mid 0$

$D \to D1 \mid 3B$

$E \to 0E1 \mid 1B$

5.2.2 Převod na nezkracující bezkontextovou gramatiku

Nezkracující bezkontextová gramatika buď vůbec neobsahuje ε-pravidla (pokud v jazyce generovaném gramatikou není prázdné slovo), a nebo existuje jediné takové pravidlo, a to pro startovací symbol gramatiky, pak ale se startovací symbol nesmí vyskytovat na pravé straně žádného pravidla (to znamená, že v každé derivaci se vyskytuje pouze na jejím začátku, pak už ne).

Při převodu na nezkracující gramatiku odstraňujeme ε-pravidla tak, že „simulujeme“ jejich použití.

Příklad 5.4

K následující gramatice vytvoříme ekvivalentní gramatiku s nezkracujícími pravidly.

$G = (\{S, A, B\},\ \{a, b\},\ P,\ S)$

$S \to aaB \mid bAb$

$A \to aBa \mid \varepsilon$

$B \to AbAa \mid a$

Téměř všechna pravidla jsou nezkracující, je zde pouze jediné ε-pravidlo – $A \to \varepsilon$. Přepisovaný symbol A se nachází na pravé straně dvou pravidel – druhého a pátého. Obě tato pravidla necháme a přidáme jejich varianty se „simulovaným“ použitím ε-pravidla:

$G' = (\{S, A, B\},\ \{a, b\},\ P',\ S)$
$S \to aaB \mid bAb \mid bb$
$A \to aBa$
$B \to AbAa \mid bAa \mid Aba \mid ba \mid a$

U prvního ze zpracovávaných pravidel máme jen jeden výskyt neterminálu A, tedy existuje pouze jediná další varianta (toto jediné A odstraníme – přepíšeme na ε). U druhého zpracovávaného pravidla jsou dva výskyty neterminálu A, proto musíme vytvořit variant více – odstraníme jen první A, odstraníme jen druhé, a nebo odstraníme obě.

Derivace v gramatice G' bude téměř stejná jako v gramatice G, ale kroky s použitím ε-pravidel jsou „přeskočeny“, například

v gramatice G: $S \Rightarrow aaB \Rightarrow aaAbAa \Rightarrow aabAa \Rightarrow aabaBaa \Rightarrow aabaaaa$
v gramatice G': $S \Rightarrow aaB \Rightarrow aabAa \Rightarrow aabaBaa \Rightarrow aabaaaa$

Příklad 5.5

K zadané gramatice vytvoříme ekvivalentní nezkracující gramatiku (tj. generující tentýž jazyk).

$G = (\{S, A\},\ \{a, b, c\},\ P,\ S)$
$S \to aaS \mid bbA \mid \varepsilon$
$A \to cSbSc \mid \varepsilon$

Gramatika generuje také prázdné slovo. Proto nejdřív použijeme stejný postup jako v předchozím příkladu, ale pak ještě musíme zajistit, aby výsledná gramatika také generovala prázdné slovo a přitom zůstala nezkracující.

Nejdřív vytvoříme pomocnou gramatiku G', která nemá žádná ε-pravidla, její jazyk je $L(G') = L(G) - \{\varepsilon\}$.

$G' = (\{S, A\},\ \{a, b, c\},\ P',\ S)$
$S \to aaS \mid aa \mid bbA \mid bb$
$A \to cSbSc \mid cbSc \mid cSbc \mid cbc$

Přidáme nový neterminál (startovací symbol) a vytvoříme gramatiku, která již bude ekvivalentní původní gramatice.

$G'' = (\{S', S, A\},\ \{a, b, c\},\ P'',\ S')$
$S' \to S \mid \varepsilon$
$S \to aaS \mid aa \mid bbA \mid bb$
$A \to cSbSc \mid cbSc \mid cSbc \mid cbc$

Derivace v gramatice G: $S \Rightarrow aaS \Rightarrow aaaaS \Rightarrow aaaa$
Derivace v gramatice G'': $S' \Rightarrow S \Rightarrow aaS \Rightarrow aaaa$

Příklad 5.6

Někdy se může stát, že „simulací“ ε-pravidla vytvoříme další ε-pravidlo. Pak je třeba postup provádět rekurzívně tak dlouho, dokud nejsou všechna ε-pravidla odstraněna.

$G = (\{S, A, B\},\ \{a, b\},\ P,\ S)$

$S \to bA \mid aS \mid bB \mid a$
$A \to baB \mid \varepsilon$
$B \to AA \mid b$

Po odstranění pravidla $A \to \varepsilon$ dostaneme tuto gramatiku:

$G' = (\{S, A, B\},\ \{a, b\},\ P',\ S)$

$S \to bA \mid b \mid aS \mid bB \mid a$
$A \to baB$
$B \to AA \mid A \mid \varepsilon \mid b$

Odstraněním obou symbolů A v pravidle $B \to AA$ vzniklo další ε-pravidlo, které je také třeba odstranit:

$G'' = (\{S, A, B\},\ \{a, b\},\ P'',\ S)$

$S \to bA \mid b \mid aS \mid bB \mid b \mid a$
$A \to baB \mid ba$
$B \to AA \mid A \mid b$

Rekurze při řešení pravidel se zbavíme tak, že ji přeneseme jinam. Můžeme použít řešení podobné tomu z redukce gramatiky – vytvoříme (rekurzívně) množinu N_ε všech neterminálů, které lze (třeba i po více než jednom kroku) přepsat na prázdné slovo ε.

V množině $N_{\varepsilon,0}$ bude pouze prázdný řetězec, tj. $N_{\varepsilon,0} = \{\varepsilon\}$. V dalších krocích do této množiny přidáváme neterminály, které lze přepsat na řetězec skládající se pouze ze symbolů, které byly do této množiny přidány v předchozích krocích, tj.
$N_{\varepsilon,i} = N_{\varepsilon,i-1} \cup \{X \in N \mid X \to \alpha,\ \alpha \in N_{\varepsilon,i-1}\}$.

Příklad 5.7

Podle pravidel gramatiky

$G = (\{S, A, B, C, D\},\ \{a, b, c, d\},\ P,\ S)$

$S \to aAa \mid aa$
$A \to BC \mid b$
$B \to aC \mid cD \mid \varepsilon$
$C \to AAa \mid \varepsilon$
$D \to AAB \mid d$

vytvoříme tyto množiny:

$N_{\varepsilon,0} = \{\varepsilon\}$
$N_{\varepsilon,1} = \{B, C\}$ podle $B \to \varepsilon,\ C \to \varepsilon$, prázdné slovo už nezařazujeme

$N_{\varepsilon,2} = \{B, C, A\}$ podle $A \to BC$
$N_{\varepsilon,3} = \{B, C, A, D\}$ podle $D \to AAB$

V dalším kroku již nelze další neterminál přidat (ostatně všechny už v množině jsou), proto $N_\varepsilon = N_{\varepsilon,3} = \{B, C, A, D\}$.

Množiny použijeme takto: pokud je na pravé straně pravidla některý z neterminálů obsažených v množině N_ε, pak vytvoříme další pravidla se „simulovaným" použitím ε-pravidel na tento neterminál, ε-pravidla odstraníme.

Postup je prakticky stejný jako předchozí, ale zpracování gramatiky již není třeba provádět rekurzívně (rekurze byla použita při generování množiny N_ε). Fialovou barvou jsou zvýrazněny neterminály obsažené v množině N_ε (a tedy vytvoříme pravidlo, ve kterém je odstraníme), modrou barvou pak nová pravidla.

$S \to aAa \mid aa \mid aa$ (vlastně máme dvě stejná pravidla, jedno může být odstraněno)
$A \to BC \mid C \mid B \mid b$
$B \to aC \mid a \mid cD \mid c$
$C \to AAa \mid Aa \mid a$
$D \to AAB \mid AB \mid B \mid d$

Příklad 5.8

Postup předvedený v předchozím příkladu použijeme na gramatiku z příkladu 5.6.

$G = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
$S \to bA \mid aS \mid bB \mid a$
$A \to baB \mid \varepsilon$
$B \to AA \mid b$

Opět rekurzívně vytvoříme množinu N_ε:

$N_{\varepsilon,0} = \{\varepsilon\}$
$N_{\varepsilon,1} = \{A\}$
$N_{\varepsilon,2} = \{A, B\} = N_{\varepsilon,3} = N_\varepsilon$

Vychází nám tato gramatika:

$G' = (\{S, A, B\},\ \{a, b\},\ P',\ S)$
$S \to bA \mid b \mid aS \mid bB \mid b \mid a$
$A \to baB \mid ba$
$B \to AA \mid A \mid b$

Oproti původnímu řešení není třeba několikrát přepisovat pravidla gramatiky.

Úkol 5.3

K následujícím gramatikám vytvořte ekvivalentní nezkracující gramatiky.

$G_1 = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
$S \to aSbA \mid ab$
$A \to aAbA \mid \varepsilon$
$B \to bAbB \mid b$

$G_2 = (\{S, A\},\ \{0, 1\},\ P,\ S)$
$S \to S1S1 \mid A \mid \varepsilon$
$A \to 0A0 \mid 1$

$G_3 = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
$S \to BAa \mid aS \mid \varepsilon$
$A \to aA \mid \varepsilon$
$B \to bBA \mid \varepsilon$

$G_4 = \{S, A, B\},\ \{a, b\},\ P,\ S)$
$S \to AB \mid BA \mid b$
$A \to aaA \mid \varepsilon$
$B \to AA \mid bS \mid a$

5.2.3 Odstranění jednoduchých pravidel

Jednoduchá pravidla jsou taková pravidla, na jejichž pravé straně máme jen jediný symbol, a to neterminál, například $A \to B$. Jednoduchá pravidla zbytečně prodlužují výpočet a v některých algoritmech mohou být překážkou při programování.

Pokud tento typ pravidel chceme odstranit, můžeme jednoduše nahradit symbol na pravé straně jednoduchého pravidla celou množinou pravidel, na která se tento symbol přepisuje. Například pokud existují pravidla $B \to \alpha \mid \beta \mid \gamma$, pak jednoduché pravidlo $A \to B$ nahradíme pravidly $A \to \alpha \mid \beta \mid \gamma$ (v derivaci to bude znamenat zkrácení odvození o jeden krok – zpracování jednoduchého pravidla).

Protože touto náhradou můžeme opět pro daný neterminál dostat jednoduché pravidlo, postup je rekurzivní a lze ho zjednodušit přenesením rekurze na množiny neterminálů podobně jako v předchozích postupech. Vytváříme množiny N_A postupně pro všechny neterminály $A \in N$, které inicializujeme jednoprvkovou množinou obsahující neterminál v označení množiny (tj. v tomto případě A). Jde o množiny neterminálů, jejichž pravidla se pomocí jednoduchých pravidel mají postupně připsat do množiny pravidel pro daný neterminál A.

Příklad 5.9

Odstraníme jednoduchá pravidla v následující gramatice:

$G = (\{S, A, B, C\},\ \{a, b\},\ P,\ S)$
$S \to aBa \mid A$
$A \to aA \mid B$
$B \to bB \mid AB \mid C \mid \varepsilon$
$C \to bA \mid b$

Rekurzívně vytvoříme množiny $N_S,\ N_A,\ N_B,\ N_C$. Na začátku rekurze, v bázi, bude v dané množině pouze ten symbol, pro který množinu tvoříme, tj. například $N_{S,0} = \{S\}$.

$N_{S,0} = \{S\}$
$N_{S,1} = \{S, A\}$ (podle $S \to A$)
$N_{S,2} = \{S, A, B\}$ (podle $A \to B$)
$N_{S,3} = \{S, A, B, C\} = N_{S,4} = N_S$ (podle $B \to C$)

$N_{A,0} = \{A\}$
$N_{A,1} = \{A, B\}$ (podle $A \to B$
$N_{A,2} = \{A, B, C\} = N_{A,3} = N_A$ (podle $B \to C$)

$N_{B,0} = \{B\}$
$N_{B,1} = \{B, C\} = N_{B,2} = N_A$ (podle $B \to C$)

$N_{C,0} = \{C\} = N_{C,1} = N_C$

Z gramatiky odstraníme všechna jednoduchá pravidla. Po jejich odstranění pak použijeme vytvořené množiny – pokud například pro neterminál A je $N_A = \{A, B, C\}$, pak v gramatice pro neterminál A použijeme všechna pravidla, která v původní gramatice byla pro neterminály A, B, C.

$G' = (\{S, A, B, C\},\ \{a, b\},\ P',\ S)$

$S \to aBa \mid aA \mid\ bB \mid AB \mid \varepsilon \mid\ bA \mid b$
$A \to aA \mid\ bB \mid AB \mid \varepsilon \mid bA \mid b$
$B \to bB \mid AB \mid \varepsilon \mid bA \mid b$
$C \to bA \mid b$

Pokud nám vadí pouze jednoduchá pravidla samotná, předchozí postup je dostačující. Jestliže však je překážkou samotné chování jednoduchých pravidel (například prodlužování derivace), je třeba předem odstranit ε-pravidla. Například v gramatice z příkladu 5.10 po odstranění jednoduchých pravidel existuje derivace $S \Rightarrow AB \Rightarrow A \Rightarrow \ldots$, kde nacházíme ekvivalent použití jednoduchého pravidla $S \to A$ z původní gramatiky.

Příklad 5.10

Gramatiku z příkladu 5.10 předem upravíme – převedeme na nezkracující. Pak teprve odstraníme jednoduchá pravidla.

$G = (\{S, A, B, C\},\ \{a, b\},\ P,\ S)$
$S \to aBa \mid A$
$A \to aA \mid B$
$B \to bB \mid AB \mid C \mid \varepsilon$
$C \to bA \mid b$

Po převodu na nezkracující gramatiku:

$N_\varepsilon = \{B, A, S\}$

$G' = (\{S, A, B, C\},\ \{a, b\},\ P',\ S)$
$S \to aBa \mid aa \mid A \mid \varepsilon$
$A \to aA \mid a \mid B$
$B \to bB \mid b \mid AB \mid A \mid B \mid C$
$C \to bA \mid b$

Pravidlo $S \to \varepsilon$ můžeme nechat, protože se S nevyskytuje na pravé straně žádného pravidla (není nutné vytvářet nový startovací symbol). Odstraníme jednoduchá pravidla:

$N_S = \{S, A, B, C\}$
$N_A = \{A, B, C\}$
$N_B = \{B, A, C\}$ (všimněte si rozdílu oproti podobné množině v příkladu 5.10)
$N_C = \{C\}$

$G'' = (\{S, A, B, C\},\ \{a, b\},\ P'',\ S)$
$S \to aBa \mid aa \mid aA \mid a \mid \varepsilon \mid aA \mid a \mid bB \mid b \mid AB \mid bA \mid b$
$A \to aA \mid a \mid bB \mid b \mid AB \mid bA \mid b$
$B \to bB \mid b \mid AB \mid aA \mid a \mid bA \mid b$
$C \to bA \mid b$

Úkol 5.4

Odstraňte jednoduchá pravidla z následujících gramatik:

$G_1 = (\{S, A, B\},\ \{0, 1\},\ P,\ S)$
$S \to 0A1 \mid B \mid \varepsilon$
$A \to 1S \mid S \mid B$
$B \to 1A \mid S \mid 0$

$G_2 = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
$S \to aA \mid bB \mid A$
$A \to bAb \mid S \mid \varepsilon$
$B \to aBa \mid A \mid a$

5.2.4 Gramatika bez cyklu a vlastní gramatika

Gramatika bez cyklu je taková gramatika, kde neexistuje žádná derivace $A \Rightarrow^+ A$ pro jakýkoliv neterminál A. Už z definice je zřejmé, jak gramatiku upravit, aby splňovala tuto vlastnost – stačí ji převést na nezkracující gramatiku (bez ε-pravidel) a odstranit jednoduchá pravidla.

Vlastní gramatika je gramatika bez cyklu, nezkracující a bez nadbytečných symbolů. Postup je tedy následující:

- převedeme gramatiku na nezkracující,
- odstraníme jednoduchá pravidla,
- odstraníme nadbytečné symboly postupem, který již známe z redukce gramatiky.

5.2.5 Levá a pravá rekurze

Gramatika je rekurzivní zleva, pokud v ní existuje derivace $A \Rightarrow^+ A\alpha$, gramatika je rekurzívní zprava, pokud v ní existuje derivace $A \Rightarrow^+ \alpha A$, A je některý neterminál gramatiky, α je jakýkoliv řetězec z terminálů a neterminálů.

Levá nebo pravá rekurze nám může bránit v některých dalších úpravách a nebo v naprogramování postupu popsaného gramatikou. Obvykle nám vadí jen levá a nebo jen pravá rekurze, proto jejich odstranění řešíme jednoduše převodem na tu, která nám nevadí. Při odstraňování rekurze vycházíme z toho, že k témuž řetězci lze dojít více různými způsoby. Ukážeme si odstranění *přímé rekurze*, jejíž existence je patrná přímo z pravidla.

Příklad 5.11

Následující gramatiku zbavíme přímé levé rekurze.

$G = (\{S, A, B\},\ \{a, b, c\},\ P,\ S)$

$S \to aAb \mid b$

$A \to Aa \mid bB \mid ab$

$B \to Ba \mid Bb \mid ca$

Levá rekurze je v pravidlech $A \to Aa$ a dále $B \to Ba \mid Bb$. Nejdřív vyřešíme první z těchto pravidel – pro neterminál A. Množinu pravidel $A \to Aa \mid bB \mid ab$ nahradíme jinou množinou, která generuje tentýž řetězec. Jak může vypadat derivace z neterminálu A?

$A \Rightarrow Aa \Rightarrow Aaa \Rightarrow Aaaa \Rightarrow^* Aa^i \Rightarrow bBa^i$ (generujeme zprava doleva, a to nejdřív rekurzívním pravidlem, rekurze je pak ukončena vlevo některým nerekurzívním pravidlem).

Stejný řetězec lze generovat pravou rekurzí, a to přidáním nového neterminálu a úpravou pravidel (původní pravidla jsou $A \to Aa \mid bB \mid ab$):

$A \to bBA' \mid abA' \mid bB \mid ab$

$A' \to aA' \mid a$

Derivace ze symbolu A pak vypadá následovně:

$A \Rightarrow bBA' \Rightarrow bBaA' \Rightarrow bBaaA' \Rightarrow^* bBa^{i-1}A' \Rightarrow bBa^i$

Zbývá upravit pravidla pro neterminál B – $B \to Ba \mid Bb \mid ca$, nahradíme je pravidly

$B \to caB' \mid ca$

$B' \to aB' \mid bB' \mid a \mid b$

Vytvořili jsme tedy ekvivalentní gramatiku bez přímé levé rekurze:

$G' = (\{S, A, A', B, B'\},\ \{a, b, c\},\ P',\ S)$

$S \to aAb \mid b$

$A \to bBA' \mid abA' \mid bB \mid ab$

$A' \to aA' \mid a$

$B \to caB' \mid ca$

$B' \to aB' \mid bB' \mid a \mid b$

Uvedený postup je použitelný pouze na *vlastní gramatiku* (tj. gramatiku bez cyklu, nezkracující, bez nadbytečných symbolů). Gramatika z předchozího příkladu je vlastní, proto nebylo třeba ji předem upravovat.

Příklad 5.12

Odstraníme přímou levou rekurzi v následující gramatice:

$G = (\{S, A, B\},\ \{a, b, c\},\ P,\ S)$

$S \to bAa \mid c \mid A$
$A \to Ba \mid b \mid Abb$
$B \to aBb \mid Bbc \mid ca \mid \varepsilon$

Tato gramatika není vlastní. Je třeba nejdřív převést ji na nezkracující, odstranit jednoduchá pravidla a teprve potom můžeme odstranit levou rekurzi.

1. Odstraníme ε-pravidla (resp. jediné, $B \to \varepsilon$):

   $G' = (\{S, A, B\},\ \{a, b, c\},\ P',\ S)$

   $S \to bAa \mid c \mid A$
   $A \to Ba \mid a \mid b \mid Abb$
   $B \to aBb \mid ab \mid Bbc \mid bc \mid ca$

2. Odstraníme jednoduché pravidlo $S \to A$:

   $G'' = (\{S, A, B\},\ \{a, b, c\},\ P'',\ S)$

   $S \to bAa \mid c \mid Ba \mid a \mid b \mid Abb$
   $A \to Ba \mid a \mid b \mid Abb$
   $B \to aBb \mid ab \mid Bbc \mid bc \mid ca$

3. Odstraníme přímou levou rekurzi. Pravidla $A \to Ba \mid a \mid b \mid Abb$ nahradíme pravidly

   $A \to BaA' \mid aA' \mid bA' \mid Ba \mid a \mid b$
   $A' \to bbA' \mid bb$

   Dále pravidla $B \to aBb \mid ab \mid Bbc \mid bc \mid ca$ nahradíme pravidly

   $B \to aBbB' \mid abB' \mid aB' \mid |bcB' \mid caB' \mid aBb \mid ab \mid bc \mid ca$
   $B' \to bcB' \mid bc$

Vytvořili jsme tuto gramatiku:

$G''' = (\{S, A, A', B, B'\},\ \{a, b, c\},\ P''',\ S)$

$S \to bAa \mid c \mid Ba \mid a \mid b \mid Abb$
$A \to BaA' \mid aA' \mid bA' \mid Ba \mid a \mid b$
$A' \to bbA' \mid bb$
$B \to aBbB' \mid abB' \mid aB' \mid |bcB' \mid caB' \mid aBb \mid ab \mid bc \mid ca$
$B' \to bcB' \mid bc$

Příklad 5.13

Odstraníme přímou pravou rekurzi v této gramatice:

$G = (\{S, A, B\},\ \{a, b, c\},\ P,\ S)$

$S \to aAb \mid AB$
$A \to abA \mid c \mid Sc$
$B \to bbB \mid cB \mid a$

Jde o vlastní gramatiku, nemusíme ji předem do tohoto tvaru upravovat. Postupujeme stejně jako u odstranění levé rekurze, jen zaměníme levou a pravou stranu. Přidáme nové neterminály a upravíme pravidla.

Z neterminálu A může být například tato derivace:

$A \Rightarrow abA \Rightarrow ababA \Rightarrow^* (ab)^i A \Rightarrow (ab)^i c$

Pravidla $A \to abA \mid c \mid Sc$ nahradíme pravidly

$A \to A'c \mid A'Sc \mid c \mid Sc$
$A' \to A'ab \mid ab$

Derivace z neterminálu A se změní:

$A \Rightarrow A'c \Rightarrow A'abc \Rightarrow A'ababc \Rightarrow^* A'(ab)^{i-1}c \Rightarrow (ab)^i c$

Vygenerovaný řetězec je tentýž, ale zatímco u pravé rekurze byl generován zleva doprava, u levé rekurze je generován opačně – zprava doleva.

Podobně upravíme pravidla $B \to bbB \mid cB \mid a$:

$B \to B'a \mid a$
$B' \to B'bb \mid B'c \mid bb \mid c$

Vychází nám tato gramatika:

$G' = (\{S, A, A', B, B'\},\ \{a, b, c\},\ P',\ S)$

$S \to aAb \mid AB$
$A \to A'c \mid A'Sc \mid c \mid Sc$
$A' \to A'ab \mid ab$
$B \to B'a \mid a$
$B' \to B'bb \mid B'c \mid bb \mid c$

Rekurze může být také nepřímá, například v pravidlech

$A \to Bba \mid a$
$B \to Aab \mid b$

Levou rekurzi, včetně nepřímé, odstraníme tak, že pravidla převedeme do tvaru, kdy pravá strana pravidla začíná neterminálem pouze za přesně určených okolností. Postup:

1. Stanovíme pořadí neterminálů. Můžeme si je například označit indexy, přejmenovat nebo jednoduše určit jejich pořadí. Nechť pořadí neterminálů je $(A_1, A_2, \ldots, A_n)$.

Účelem algoritmu je upravit pravidla tak, aby mohla začínat pouze neterminály s vyšším indexem než je index přepisovaného neterminálu. Tím zcela odstraníme levou rekurzi.

2. Pro neterminály $A_i, A_j,\ 1 \leq i, j \leq n$ postupně transformujeme pravidla. Pro první krok stanovíme $i = 1, j = 1$.
   - Pokud $j < i$ a existuje pravidlo $A_i \to A_j\alpha$, kde α je jakýkoliv řetězec (tj. pravidlo pro A_i začíná neterminálem A_j), pak symbol A_j v pravidle nahradíme postupně všemi pravými stranami pravidel pro A_j, například
     $A_i \to A_jabB$
     $A_j \to bDA_ia \mid CA_ja$
     První z uvedenených pravidel nahradíme pravidly $A_i \to bdA_iaabB \mid CA_jaabB$.
     Provedeme $j = j + 1$.
   - Pokud $j = i$, pak odstraníme přímou levou rekurzi podle postupu, který už známe. Provedeme $j = j + 1$.
   - Pokud $j > i$, provedeme $i = i + 1,\ j = 1$.
3. Bod 2 postupu provedeme pro všechna i, $1 \leq i \leq n$.

Aby byl postup použitelný, musí být uplatněn pouze na *vlastní gramatiku*. Postup pro pravou rekurzi je obdobný, jen zaměníme levou za pravou stranu.

Příklad 5.14

Odstraníme levou rekurzi v následující gramatice:

$G = (\{S, A, B\},\ \{a, b, c\},\ P,\ S)$

$S \to aA \mid AB \mid b$
$A \to SBa \mid a$
$B \to Bb \mid Aa \mid c$

Přímá levá rekurze je zde jen v jednom pravidle, ale nepřímá rekurze se týká více pravidel. Gramatika je vlastní, není třeba ji do tohoto tvaru upravovat. Stanovíme pořadí neterminálů: $(A_1, A_2, A_3) = (S, A, B)$.

- $i = 1,\ j = 1$:
  není třeba upravovat, pravidla pro S neobsahují přímou rekurzi.
- $i = 2,\ j = 1$:
  jedno z pravidel pro A začíná neterminálem s „nižším indexem“, S; upravíme:
  $A \to aABa \mid ABBa \mid bBa \mid a$
- $i = 2,\ j = 2$:
  řešíme přímou rekurzi (protože $i = j$):
  $A \to aABaA' \mid bBaA' \mid aA' \mid aABa \mid bBa \mid a$
  $A' \to BBaA' \mid BBa$

- $i = 3,\ j = 1$:
  žádné z pravidel pro B nezačíná symbolem S
- $i = 3,\ j = 2$:
  změna se týká druhého pravidla pro symbol B (začíná symbolem A, jehož index je 2). Při nahrazení symbolu A v tomto pravidle použijeme již upravená pravidla pro A:

  $B \to Bb \mid aABaA'a \mid bBaA'a \mid aA'a \mid aABaa \mid bBaa \mid aa \mid c$
- $i = 3,\ j = 3$:
  odstraníme přímou rekurzi:

  $B \to aABaA'aB' \mid bBaA'aB' \mid aA'aB' \mid aABaaB' \mid bBaaB' \mid aaB' \mid cB' \mid$
  $\quad aABaA'a \mid bBaA'a \mid aA'a \mid aABaa \mid bBaa \mid aa \mid c$
  $B' \to bB' \mid b$

Celá gramatika bez levé rekurze:

$G' = (\{S, A, A', B, B'\},\ \{a, b, c\},\ P',\ S)$
$S \to aA \mid AB \mid b$
$A \to aABaA' \mid bBaA' \mid aA' \mid aABa \mid bBa \mid a$
$A' \to BBaA' \mid BBa$
$B \to aABaA'aB' \mid bBaA'aB' \mid aA'aB' \mid aABaaB' \mid bBaaB' \mid aaB' \mid cB' \mid$
$\quad aABaA'a \mid bBaA'a \mid aA'a \mid aABaa \mid bBaa \mid aa \mid c$
$B' \to bB' \mid b$

Odstraněním přímé rekurze rozšiřujeme množinu neterminálů, proto je nutné *dynamicky* upravovat také nadefinovanou posloupnost neterminálů určující pořadí. Neterminály s pravidly, které takto vytvoříme, je třeba zahrnout do algoritmu.

Příklad 5.15

Odstraníme levou rekurzi v následující gramatice:

$G = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
$S \to aA \mid bB$
$A \to BaA \mid ab$
$B \to BaA \mid b$

- $i = 1,\ j = 1$: není třeba upravovat, pravidla pro S neobsahují přímou rekurzi.
- $i = 2,\ j = 1, 2$: není třeba upravovat, pravidla pro A nezačínají symboly S a A.
- $i = 3,\ j = 1, 2$: není třeba upravovat, pravidla pro B nezačínají symboly S a A.
- $i = 3,\ j = 3$: odstraníme přímou rekurzi.

  $B \to bB' \mid b$
  $B' \to AaB' \mid Aa$

Měníme posloupnost: $(A_1, A_2, A_3, A_4) = (S, A, B, B')$

- $i = 4,\ j = 1$: není třeba upravovat, pravidla pro B' nezačínají symbolem S.
- $i = 4,\ j = 2$:

  $B' \to BaAaB' \mid abaB' \mid BaAa \mid aba$
- $i = 4,\ j = 3$:

  $B' \to bB'aAaB' \mid baAaB' \mid abaB' \mid bB'aAa \mid baAa \mid aba$
- $i = 4,\ j = 4$: není třeba upravovat.

Výsledná gramatika:

$G' = (\{S, A, B, B'\},\ \{a, b\},\ P',\ S)$
$S \to aA \mid bB$
$A \to BaA \mid ab$
$B \to bB' \mid b$
$B' \to AaB' \mid Aa$
$B' \to bB'aAaB' \mid baAaB' \mid abaB' \mid bB'aAa \mid baAa \mid aba$

Úkol 5.5

1. Odstraňte levou rekurzi v těchto gramatikách (zvažte, zda není třeba gramatiku předem upravit):

   $G_1 = (\{S, A, B\},\ \{a, b, c\},\ P,\ S)$
   $S \to aAB \mid b \mid SB \mid \varepsilon$
   $A \to bA \mid Aac \mid AB \mid a$
   $B \to BbA \mid c$

   $G_2 = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
   $S \to AB \mid BA$
   $A \to ABbA \mid bA \mid \varepsilon$
   $B \to SA \mid a \mid BbA$

2. Odstraňte pravou rekurzi v těchto gramatikách (příp. gramatiku předem upravte):

   $G_1 = (\{S, A, B\},\ \{0, 1\},\ P,\ S)$
   $S \to 11A \mid 01B \mid \varepsilon$
   $A \to 11A \mid B01 \mid 10$
   $B \to 01B \mid 00AB \mid 1A1 \mid 01$

   $G_2 = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
   $S \to aAb \mid abB \mid ASA$
   $A \to bA \mid B \mid \varepsilon$
   $B \to bbB \mid abAB \mid abb$

5.3 Normální formy bezkontextových gramatik

Normální formy (čehokoliv) slouží k usnadnění porovnávání a případných dalších úprav. V případě bezkontextových gramatik lze využít Chomského normální formu (CNF) a Greibachovu normální formu (GNF).

5.3.1 Chomského normální forma

Při převodu gramatiky do Chomského normální formy je třeba, aby pravidla byla ve tvaru

- $A \to BC$, tedy na pravé straně dva neterminály,
- $A \to a$, tedy na pravé straně jeden terminál, a nebo
- $S \to \varepsilon$ (S je startovací symbol gramatiky), a to pouze v případě, že S se nevyskytuje na pravé straně žádného pravidla (tj. toto pravidlo lze využít pouze na začátku derivace).

Postupujeme takto:

1. Převedeme gramatiku do tvaru vlastní gramatiky (tj. převod na nezkracující gramatiku, odstraníme jednoduchá pravidla, odstraníme nadbytečné neterminály).
2. Pravidla, která mají na pravé straně jen jeden symbol (půjde vždy o terminál, protože jednoduchá pravidla jsme již odstranili) necháme – odpovídají požadavkům na normální formu; dále zpracováváme jen pravidla, jejichž pravá strana má délku alespoň 2.
3. Pro každé pravidlo $A \to \alpha$, kde $|\alpha| > 1$, provedeme tyto úpravy:
   - každý terminál a v řetězci α nahradíme *novým* (nově vytvořeným) neterminálem N_a (podle dolního indexu poznáme, který terminál byl nahrazen), takže například pravidlo $A \to BaAbca$ zaměníme za pravidlo $A \to BN_aAN_bN_cN_a$,
   - vytvoříme nová pravidla $N_a \to a$ pro každý terminál a.
4. Délku pravých stran všech pravidel omezíme na nejvýše 2 takto: pro každé pravidlo $A \to B_1B_2B_3 \ldots B_n,\ n > 2$, vytvoříme množinu pravidel

   $A \to B_1X_1$
   $X_1 \to B_2X_2$
   $\ldots$
   $X_{n-3} \to X_{n-2}B_{n-2}$
   $X_{n-2} \to B_{n-1}B_n$

   Například pravidlo $A \to BCDEF$ nahradíme množinou pravidel

   $A \to BX_1$
   $X_1 \to CX_2$
   $X_2 \to DX_3$
   $X_3 \to EF$

Neterminály $X_1, \ldots, X_{n-2}$, které používáme při propojení generování pravé strany původního pravidla, musí být *nové*, tedy různé pro různá pravidla, která takto upravujeme.

Příklad 5.16

Následující gramatiku převedeme do Chomského normální formy:

$G = (\{S, A, B\},\ \{a, b, c\},\ P,\ S)$

$S \to aBAc \mid c$
$A \to Ab \mid \varepsilon$
$B \to bB \mid AS \mid a$

Tato gramatika není vlastní. Nejdřív odstraníme ε-pravidlo $A \to \varepsilon$.

$S \to aBAc \mid aBc \mid c$
$A \to Ab \mid b$
$B \to bB \mid AS \mid S \mid a$

Jedno z pravidel je jednoduché ($B \to S$), to je třeba odstranit:

$S \to aBAc \mid aBc \mid c$
$A \to Ab \mid b$
$B \to bB \mid AS \mid aBAc \mid aBc \mid c \mid a$

Všechna pravidla, jejichž pravá strana je dlouhá alespoň 2 symboly, upravíme – terminály nahradíme příslušnými neterminály:

$S \to N_aBAN_c \mid N_aBN_c \mid c$
$A \to AN_b \mid b$
$B \to N_bB \mid AS \mid N_aBAN_c \mid N_aBN_c \mid c \mid a$
$N_a \to a$
$N_b \to b$
$N_c \to c$

Všechna pravidla s pravou stranou delší než 2 zkrátíme a získáme tuto gramatiku:

$G_{CNF} = (\{S, A, B,\ N_a, N_b, N_c,\ X_1, X_2, X_3, X_4, X_5, X_6\},\ \{a, b, c\},\ P',\ S)$

$S \to N_aX_1 \mid N_aX_3 \mid c$
$X_1 \to BX_2$
$X_2 \to AN_c$
$X_3 \to BN_c$
$A \to AN_b \mid b$

$B \to N_bB \mid AS \mid N_aX_4 \mid N_aX_6 \mid c \mid a$
$X_4 \to BX_5$
$X_5 \to AN_c$
$X_6 \to BN_c$

$N_a \to a$
$N_b \to b$
$N_c \to c$

Jedna z derivací v původní gramatice G:

$S \Rightarrow aBAc \Rightarrow aaAc \Rightarrow aaAbc \Rightarrow aabc$

Derivace stejného slova v gramatice G'' (po předběžné úpravě):

$S \Rightarrow aBAc \Rightarrow aaAc \Rightarrow aabc$

Obdobně ve výsledné gramatice G_{CNF}:

$S \Rightarrow N_aX_1 \Rightarrow aX_1 \Rightarrow aBX_2 \Rightarrow aBAN_c \Rightarrow aBAc \Rightarrow aaAc \Rightarrow aabc$

Úkol 5.6

Následující gramatiky převeďte do Chomského normální formy:

$G_1 = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
$S \to AaB \mid ba \mid aA$
$A \to aAB \mid aAb \mid ab$
$B \to bBASb \mid b$

$G_2 = (\{S, A, B\},\ \{0, 1\},\ P,\ S)$
$S \to 0B \mid 1A \mid \varepsilon$
$A \to 10A1 \mid AB \mid \varepsilon$
$B \to A1B01 \mid 1S1 \mid 0$

$G_3 = (\{E, F, G\},\ \{a, b\},\ P,\ E)$
$E \to aaF \mid GF \mid b$
$F \to aFaa \mid \varepsilon$
$G \to aFGbG \mid \varepsilon$

$G_4 = (\{S, A, B\},\ \{0, 1\},\ P,\ S)$
$S \to 01AB0 \mid 10BA1 \mid \varepsilon$
$A \to 1AB1AB1 \mid 111$
$B \to 0B \mid 0$

5.3.2 Greibachova normální forma

Gramatika je v Greibachově normální formě, pokud všechna pravidla této gramatiky jsou v jednom z těchto tvarů:

- $A \to aB_1B_2 \dots B_n,\ n \geq 0$, tedy na pravé straně je vždy jeden terminál a následují pouze neterminály (nemusí být žádný neterminál),
- $S \to \varepsilon$ (S je startovací symbol gramatiky), a to pouze v případě, že S se nevyskytuje na pravé straně žádného pravidla (tj. toto pravidlo lze využít pouze na začátku derivace).

Podobně jako u převodu na Chomského normální tvar, i zde budeme gramatiku předem upravovat. Možnost existence ε-pravidla pouze pro startovací symbol znamená nutnost převést gramatiku na nezkracující. Dále odstraníme jednoduchá pravidla a případně také nadbytečné symboly.

Kromě toho je zde požadavek na terminální symbol na začátku pravidla. Pokud pravidlo začíná neterminálem, logickým postupem by bylo dosazovat za tento neterminál postupně všechna pravidla, na který lze neterminál přepsat, a to rekurzívně tak dlouho, dokud nezískáme pravidlo začínající terminálem. Pokud je však v gramatice levá rekurze, dosazovali bychom do nekonečna, proto je třeba předem odstranit levou rekurzi. Celý postup je následující:

1. Převedeme gramatiku do tvaru vlastní gramatiky (převod na nezkracující, odstranění jednoduchých pravidel, odstranění nadbytečných symbolů).
2. Odstraníme levou rekurzi. Pokud jsme jejím odstraněním vytvořili ε-pravidla nebo jednoduchá pravidla, opakujeme bod 1.

3. Pravidla, jejichž pravá strana má délku 1, již vyhovují požadavkům na normální formu (a také případné ε-pravidlo pro startovací symbol).
4. Pravidla, jejichž pravá strana má délku ≥ 2, dále zpracováváme:
   - pokud pravidlo začíná neterminálem, dosadíme za tento neterminál všechna pravidla, která ho přepisují, tj. například pravidlo $A \to BbAa$ při existenci pravidel $B \to \alpha \mid \beta \mid \gamma$ nahradíme pravidly $A \to \alpha bAa \mid \beta bAa \mid \gamma bAa$,
   - to provádíme rekurzívně tak dlouho, dokud všechna pravidla nezačínají terminálním symbolem,
   - Všechny terminální symboly a na pravých stranách pravidel, kromě prvního terminálu v řetězci, nahradíme příslušnými symboly N_a (podobně jako při převodu na Chomského normální tvar).
5. Vytvoříme nová pravidla $N_a \to a$ pro všechny terminály a.

Příklad 5.17

Následující gramatiku převedeme do Greibachovy normální formy.

$G = (\{S, A, B, C\},\ \{a, b, c\},\ P,\ S)$

$S \to aBba \mid baA \mid \varepsilon$
$A \to BaAB \mid bb \mid \varepsilon$
$B \to CBab \mid abc$
$C \to bCb \mid a$

Sestrojíme ekvivalentní vlastní gramatiku – odstraníme ε-pravidla.

Podle množiny $N_\varepsilon = \{S, A\}$ je zřejmé, že existují pouze dvě ε-pravidla. Jedno z nich je pro startovací symbol, který se však nevyskytuje na pravé straně žádného pravidla. Proto není nutné přidávat nový neterminál a měnit startovací symbol gramatiky.

$G' = (\{S, A, B, C\},\ \{a, b, c\},\ P',\ S)$

$S \to aBba \mid baA \mid ba \mid \varepsilon$
$A \to BaAB \mid BaB \mid bb$
$B \to CBab \mid abc$
$C \to bCb \mid a$

Při převodu do Greibachovy normální formy nejdřív zajistíme, aby nejlevějším symbolem v pravých stranách pravidel byl vždy terminál. To provedeme přepsáním toho neterminálu, který je na začátku upravovaného řetězce.

Pravidlo $A \to BaAB$ nahradíme těmito pravidly:

$A \to CBabaAB \mid abcaAB$

Jedno z pravidel opět začíná neterminálem, tedy úpravu provedeme znovu:

$A \to bCbBabaAB \mid aBabaAB \mid abcaAB$

Pravidlo $A \to BaB$ nahradíme pravidly

$A \to CBabaB \mid abcaB$

Po druhé úpravě:

$A \to bCbBabaB \mid aBabaB \mid abcaB$

Pravidlo $B \to CBab$ nahradíme pravidly

$B \to bCbBab \mid aBab$

V této fázi jsme získali následující pravidla:

$S \to aBba \mid baA \mid ba \mid \varepsilon$
$A \to bCbBabaAB \mid aBabaAB \mid abcaAB \mid bCbBabaB \mid aBabaB \mid abcaB \mid bb$
$B \to bCbBab \mid aBab \mid abc$
$C \to bCb \mid a$

Zbývá splnit podmínku, podle které v pravých stranách pravidel jsou všechny symboly kromě nejlevějšího neterminální.

$G_{GNF} = (\{S, A, B, C, N_a, N_b, N_c\},\ \{a, b, c\},\ P'',\ S)$
$S \to aBN_bN_a \mid bN_aA \mid bN_a \mid \varepsilon$
$A \to bCN_bBN_aN_bN_aAB \mid aBN_aN_bN_aAB \mid aN_bN_cN_aAB \mid bCN_bBN_aN_bN_aB \mid$
$\quad \mid aBN_aN_bN_aB \mid aN_bN_cN_aB \mid bN_b$
$B \to bCN_bBN_aN_b \mid aBN_aN_b \mid aN_bN_c$
$C \to bCN_b \mid a$
$N_a \to a$
$N_b \to b$
$N_c \to c$

Příklad 5.18

Následující gramatiku převedeme do Greibachovy normální formy.

$G = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
$S \to aA \mid AB \mid b$
$A \to SBa \mid a$
$B \to Bb \mid a$

V gramatice nejsou žádná ε-pravidla ani jednoduchá pravidla, ale je tu levá rekurze, kterou musíme předem odstranit. Přímá rekurze je v jednom z pravidel pro B, nepřímá rekurze je v pravidlech pro neterminály S a A. Budeme postupovat podle algoritmu popsaného na straně 68. Stanovíme pořadí neterminálů $(A_1, A_2, A_3) = (S, A, B)$.

- $i = 1,\ j = 1$: není třeba upravovat, pravidla pro S neobsahují přímou rekurzi.
- $i = 2,\ j = 1$: dosadíme pravé strany pravidel pro S do prvního z pravidel pro neterminál A:

  $A \to aABa \mid ABBa \mid bBa \mid a$

- $i = 2,\ j = 2$: odstraníme přímou rekurzi:

  $A \to aABaA' \mid bBaA' \mid aA' \mid aABa \mid bBa \mid a$
  $A' \to BBaA' \mid BBa$

- $i = 3,\ j = 1, j = 2$: beze změny.

- $i = 3,\ j = 3$: odstraníme přímou rekurzi:

  $B \to aB' \mid a$
  $B' \to bB' \mid b$

Upravená gramatika bez levé rekurze (také nepřímé) je následující:

$G' = (\{S, A, A', B, B'\},\ \{a, b\},\ P',\ S)$
$S \to aA \mid AB \mid b$
$A \to aABaA' \mid bBaA' \mid aA' \mid aABa \mid bBa \mid a$
$A' \to BBaA' \mid BBa$
$B \to aB' \mid a$
$B' \to bB' \mid b$

Teď už můžeme gramatiku převést do Greibachovy normální formy. Nejdřív zajistíme, aby pravé strany pravidel začínaly vždy terminálním symbolem.

$S \to aA \mid aABaA'B \mid bBaA'B \mid aA'B \mid aABaB \mid bBaB \mid aB \mid b$
$A \to aABaA' \mid bBaA' \mid aA' \mid aABa \mid bBa \mid a$
$A' \to aB'BaA' \mid aBaA' \mid aB'Ba \mid aBa$
$B \to aB' \mid a$
$B' \to bB' \mid b$

V dalším kroku všechny terminály na pravých stranách pravidel, kromě nejlevějšího, zaměníme za příslušné neterminální symboly.

$G_{GNF} = (\{S, A, A', B, B', N_a\},\ \{a, b\},\ P'',\ S)$
$S \to aA \mid aABN_aA'B \mid bBN_aA'B \mid aA'B \mid aABN_aB \mid bBN_aB \mid aB \mid b$
$A \to aABN_aA' \mid bBN_aA' \mid aA' \mid aABN_a \mid bBN_a \mid a$
$A' \to aB'BN_aA' \mid aBN_aA' \mid aB'BN_a \mid aBN_a$
$B \to aB' \mid a$
$B' \to bB' \mid b$
$N_a \to a$

Úkol 5.7

Převeďte následující gramatiky do Greibachovy normální formy.
