evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
Slezská univerzita v Opavě
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Šárka Vavrečková

Teorie jazyků a automatů II

Základy teoretické informatiky

Sbírka příkladů pro cvičení

Ústav informatiky
Filozoficko-přírodovědecká fakulta v Opavě
Slezská univerzita v Opavě

Opava
24. listopadu 2016

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

Teorie jazyků a automatů II – sbírka příkladů pro cvičení

RNDr. Šárka Vavrečková, Ph.D.

Dostupné na: http://vavreckova.zam.slu.cz/formal.html

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

Sázeno v systému LaTeX

Tato inovace předmětu *Praktikum z operačních systémů* je spolufinancována Evropským sociálním fondem a Státním rozpočtem ČR, projekt č. CZ.1.07/2.3.00/09.0197, „Posílení konkurenceschopnosti výzkumu a vývoje informačních technologií v Moravskoslezském kraji“.

Předmluva

Co najdeme v těchto skriptech

Tato skripta jsou určena pro studenty informatických oborů na Ústavu informatiky Slezské univerzity v Opavě, do cvičení předmětů *Teorie jazyků a automatů II* a *Základy teoretické informatiky II*, jde vlastně o jakousi cvičebnici a sbírku příkladů.

Navazujeme na obdobná skripta pro cvičení z předmětu z předchozího semestru, tedy předpokládají se již základní znalosti v oblasti teoretické informatiky, ale ve skutečnosti se oboje skripta tematicky poněkud překrývají. Je to z toho důvodu, že během několika let procházejí oba předměty rozsáhlými změnami, a snahou je při přeskupování témat nepotrhat souvislosti.

V tomto předmětu pokračujeme pokročilejšími tématy týkajícími se regulárních a bezkontextových jazyků a přecházíme k jazykům typu 0 a 1, kterými jsme se v předchozím semestru téměř nezabývali. Cílem předmětu je nacvičit si způsob myšlení používaný kromě jiného při provádění důkazů – využívání logické dedukce.

Značení

Ve skriptech se používají následující barevné ikony:

-  Nové *pojmy*, značení apod. jsou označeny modrým symbolem, který vidíme zde vlevo. Tuto ikonu (stejně jako následující) najdeme na začátku odstavce, ve kterém je nový pojem zaváděn.
-  Konkrétní *postupy* jsou značeny také modrou ikonou. Jsou sice probírány na přednáškách, ale máme je i zde, abychom si je při procvičování připomněli.
-  Některé části textu jsou označeny fialovou ikonou, což znamená, že jde o *nepovinné úseky*, které nejsou probírány (většinou; studenti si je mohou podle zájmu vyžádat nebo sami prostudovat). Jejich účelem je dobrovolné rozšíření znalostí studentů o pokročilá témata, na která obvykle při výuce nezbývá moc času.
-  Žlutou ikonou jsou označeny odkazy, na kterých lze získat *další informace* o tématu. Nejčastěji u této ikony najdeme webové odkazy na stránky, kde se dané tématice jejich autoři věnují podrobněji.
-  Červená je ikona pro *upozornění* a poznámky.

Pokud je množství textu patřícího k určité ikoně větší, je celý blok ohraničen prostředím s ikonami na začátku i konci:

Příklad

Takto vypadá prostředí s řešeným příkladem. Příklady jsou obvykle komentovány, aby byl jasný postup jejich řešení.

Úkol

Otázky a úkoly, náměty na vyzkoušení, které se doporučuje při procvičování učiva provádět, jsou uzavřeny v tomto prostředí. Pokud je v prostředí více úkolů, jsou číslovány.

Definice

V takovém prostředí definujeme pojem či vysvětlujeme sice relativně známý, ale komplexní pojem s více významy či vlastnostmi.

Obsah

Kapitola 1

Regulární jazyky

1.1 Opakování

Nejdřív si zopakujeme něco z toho, co jsme se naučili v minulém semestru. Základy musíme znát dokonale, protože na nich budeme stavět po celý semestr.

Příklad

Jazyky zadané regulárním výrazem zapíšeme v množinovém zápisu. Ke každému jazyku napíšeme množinu všech jeho slov, která jsou kratší než 5.

$L_1 = (ab)^*$
Množinový zápis:
$L_1 = \{(ab)^n \ : \ n \geq 0\}$

Slova kratší než 5: $\{w \in L_1 \ : \ |w| < 5\} = \{\varepsilon, ab, abab\}$

Mnemotechnická pomůcka: za index v množinovém zápisu dosazujeme postupně čísla od nejnižší hodnoty zadaného rozsahu, tj. od 0. Všimněte si symbolického zápisu – nelze napsat rovnítko mezi jazyk L_1 a množinu slov tohoto jazyka kratších než 5, je třeba to zapsat jinak.

$L_2 = a^*b^*a$
Množinový zápis: $L_2 = \{a^m b^n a \ : \ m, n \geq 0\}$
Všimněte si: je třeba použít dva navzájem nezávislé indexy!

Slova kratší než 5: $\{w \in L_2 \ : \ |w| < 5\} = \{a, aa, ba, aaa, aba, bba, a^4, a^2ba, ab^2a, b^3a\}$

Mnemotechnická pomůcka: za indexy m a n v množinovém zápisu dosazujeme čísla navzájem nezávisle ($m = n = 0$, $m = 1$ a $n = 0$, $m = 0$ a $n = 1$, atd.). Pozor, prázdné slovo do jazyka nepatří!

Příklad

Určíme typ zadaného jazyka v Chomského hierarchii. Pokud se jedná o regulární jazyk, vytvoříme ekvivalentní regulární výraz. Vypíšeme všechna slova jazyka kratší než 5.

$L_1 = \{(aa)^i(bb)^j \ : \ i, j \geq 0\}$
Jde o regulární jazyk. Ekvivalentní regulární výraz: $L_1 = (aa)^*(bb)^*$

Slova kratší než 5: $\{w \in L_1 \ : \ |w| < 5\} = \{\varepsilon, a^2, b^2, a^2b^2, a^4, b^4\}$

$L_2 = \{(aa)^i(bb)^i \ : \ i \geq 0\}$

Tento jazyk je sice podobný předchozímu, ale není to regulární jazyk (je bezkontextový). Proto nelze vytvořit ekvivalentní regulární výraz.

Jak poznáme, že není regulární? Pokud je třeba (při zpracování slov jazyka automatem) synchronizovat dvě nebo dokonce více částí slova (například proto, že totéž písmeno indexu se vyskytuje na více místech v množinové reprezentaci), nejde o regulární jazyk. To lze dokázat například pomocí Pumping lemmatu (budeme brát později).

Slova kratší než 5: $\{w \in L_2 \ : \ |w| < 5\} = \{\varepsilon, a^2b^2\}$

Srovnejte s obdobnou množinou pro jazyk L_1 z tohoto příkladu.

$L_3 = \{0^i1^i0^i \ : \ i \geq 1\}$

Jde o jazyk typu 1 v Chomského hierarchii, není to regulární ani bezkontextový jazyk.

Slova kratší než 5: $\{w \in L_3 \ : \ |w| < 5\} = \{010\}$

Příklad

Vypíšeme všechna slova jazyka kratší než 5.

$L_1 = \{a^ib^j \ : \ 1 \leq i \leq j\} \cup \{c^k \ : \ k \leq 3\}$

Slova kratší než 5: $\{w \in L_1 \ : \ |w| < 5\} = \{ab, ab^2, a^2b^2, \varepsilon, c, c^2, c^3\}$

Poměrně častou chybou bývá „otočení“ relačního znaménka. V předpisu druhé množiny sjednocení máme symbol „menší nebo rovno“, ne naopak, musíme tedy vypsat slova o délce 0 až 3.

$L_2 = \{a^ib^j \ : \ 1 \leq i \leq j\} \cap \{a^ib^j \ : \ 1 \leq j \leq i\}$

Když se pozorně podíváme na množiny, jejichž průnikem je jazyk L_2, zjistíme, že tento jazyk můžeme zapsat jednodušeji, protože první množina nám dává podmínku „symbolů b má být stejně nebo více než symbolů a“, druhá množina předepisuje „symbolů b má být stejně nebo méně než symbolů a“:

$L_2 = \{a^ib^i \ : \ i \geq 1\}$

Slova kratší než 5: $\{w \in L_2 \ : \ |w| < 5\} = \{ab, a^2b^2\}$

Pozor na rozdíl mezi sjednocením a průnikem, záměna těchto dvou množinových operací je poměrně častou chybou.

Příklad

Pro zadané jazyky sestrojíme konečný automat ve všech třech reprezentacích.

$L_1 = ab^*$

δ-funkce:

$\mathcal{A}_1 = (\{q_0, q_1\}, \ \{a, b\}, \ \delta, \ q_0, \ \{q_1\})$

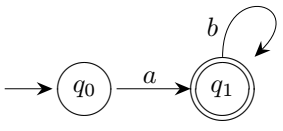
$\delta(q_0, a) = q_1$

$\delta(q_1, b) = q_1$

Tabulka přechodů:

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

Stavový diagram:



Všimněte si, že u δ-funkce je nutné uvádět plnou specifikaci, aby bylo zřejmé, který stav je počáteční a které stavy jsou koncové, u jiných reprezentací jsou tyto informace zaznamenány jinak.

$L_2 = ab^* + \varepsilon$

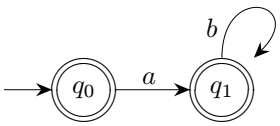
δ-funkce:

$\mathcal{A}_2 = (\{q_0, q_1\},\ \{a, b\},\ \delta,\ q_0,\ \{q_0, q_1\})$
$\delta(q_0, a) = q_1$
$\delta(q_1, b) = q_1$

Tabulka přechodů:

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

Stavový diagram:



Úkol

Srovnejte automaty pro jazyky L_1 a L_2 z předchozího příkladu a všimněte si, jaké důsledky mělo přidání prázdného slova k jazyku u jednotlivých reprezentací automatu.

Příklad

Pro zadaný jazyk sestrojíme konečný automat ve všech třech reprezentacích.

$L = a^*b + b^*a$

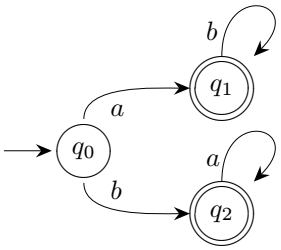
δ-funkce:

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

Tabulka přechodů:

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

Stavový diagram:



Všimněte si, jakým způsobem se v reprezentacích automatu projevila operace sčítání regulárních (pod)výrazů. Jazyk lze zapsat také následovně – v množinovém zápisu s využitím operace sjednocení množin:

$L = \{ab^i \ :\ i \geq 0\} \cup \{ba^i \ :\ i \geq 0\}$

1.2 Konečné jazyky

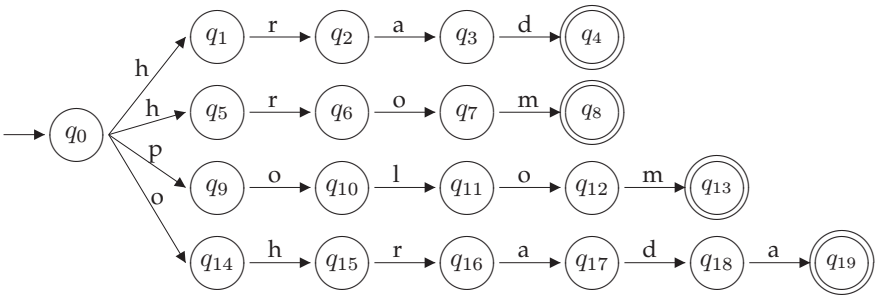
Všechny konečné jazyky jsou regulární, proto pro ně dokážeme sestrojit konečný automat.

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.

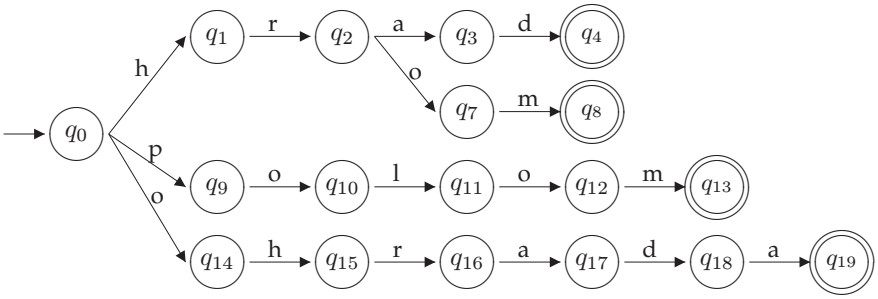
Postup je jednoduchý – pro každé slovo jazyka vytvoříme „větev“ ve stavovém diagramu. Jestliže chceme automat deterministický (opět jde o obvyklý požadavek, pokud tento postup používáme při programování), stačí sloučit počátky těch větví, které stejně začínají (konce větví nesmíme sloučit, nebylo by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

Příklad

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:



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.

Úkoly

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).

1.3 Uzávěrové vlastnosti – operace nad regulárními jazyky

V předchozím semestru jsme se zabývali základními uzávěrovými vlastnostmi regulárních jazyků, a to uzavřeností vzhledem k regulárním operacím (sjednocení, zřetězení, iterace). V tomto semestru se podíváme na některé zbývající operace.

1.3.1 Pozitivní iterace

Pozitivní iterace se používá podobně jako (běžná) iterace, ale zatímco iterace znamená řetězení $0\times$, $1\times$, $2\times$, ..., při pozitivní iteraci začínáme při řetězení až $1\times$, $2\times$, ... Matematicky můžeme zapsat iteraci a pozitivní iteraci následovně:

$$L^* = \bigcup_{i=0}^{\infty} L^i \qquad L^+ = \bigcup_{i=1}^{\infty} L^i$$

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. Není nutné vytvářet nový počáteční stav.

Označme $\mathcal{A}_1 = (Q, \Sigma, \delta_1, q_0, F)$ původní automat, pak automat rozpoznávající jazyk, který je pozitivní iterací původního jazyka, je
$\mathcal{A} = (Q,\ \Sigma,\ \delta,\ q_0,\ F)$.

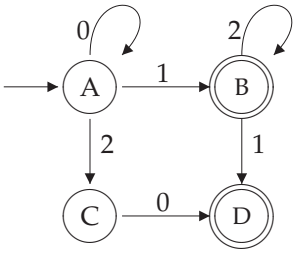
Přechodová funkce:

$$\delta(r,x) = \begin{cases} \delta_1(r,x) \ : \ r \in Q - F, x \in \Sigma & \text{(přechody z jiných než koncových stavů zkopírujeme)}, \\ \delta_1(r,x) \cup \delta_1(q_0,x) \ : \ r \in F, x \in \Sigma \end{cases}$$

Přechody vedoucí z koncových stavů také přejmeme, přidáme kopie přechodů vedoucích z q_0.

Příklad

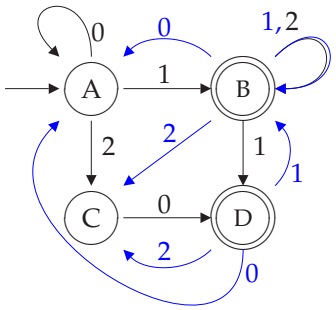
Vytvoříme pozitivní iteraci jazyka následujícího automatu:
$\mathcal{A}_1 = (\{A, B, C, D\}, \{0, 1, 2\}, \delta_1, A, \{B, D\})$



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

$\delta_1(A,0) = A$
$\delta_1(A,1) = B$
$\delta_1(A,2) = C$
$\delta_1(B,1) = D$
$\delta_1(B,2) = B$
$\delta_1(C,0) = D$

Přidáváme pouze nové přechody, počáteční stav nebudeme zařazovat do množiny koncových stavů, protože v původním automatu také nebylo rozpoznáváno slovo ε:



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

$\mathcal{A} = (\{A, B, C, D\}, \{0, 1, 2\}, \delta, A, \{B, D\})$

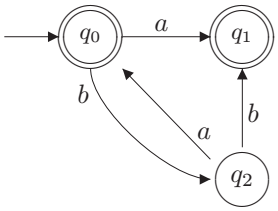
$\delta(A, 0) = A$	$\delta(B, 1) = D$	$\delta(B, 0) = A$	$\delta(D, 0) = A$
$\delta(A, 1) = B$	$\delta(B, 2) = B$	$\delta(B, 1) = B$	$\delta(D, 1) = B$
$\delta(A, 2) = C$	$\delta(C, 0) = D$	$\delta(B, 2) = C$	$\delta(D, 2) = C$

Poznámka:

Pokud původní jazyk L obsahuje slovo ε, pak platí $L^* = L^+$, v opačném případě se tyto jazyky nerovnají a platí $L^* = L^+ \cup \{\varepsilon\}$.

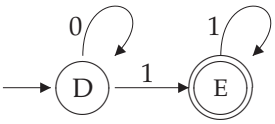
Úkoly

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



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

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



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

1.3.2 Zrcadlový obraz (reverze)

Reverze je převrácení. Zrcadlový obraz slova sestrojíme tak, že je zrcadlově „převrátíme“, tj. například 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:

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

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 související s ukončením výpočtu.

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

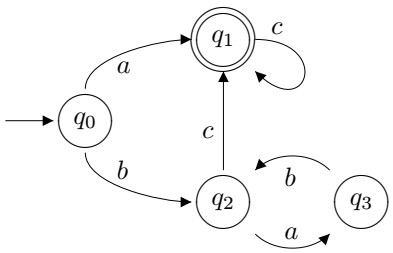
Označme $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$ původní automat, pak automat rozpoznávající jazyk, který je reverzí původního jazyka, je
$\mathcal{A} = (Q_1 \cup \{s_0\},\ \Sigma,\ \delta,\ s_0,\ F)$, množina koncových stavů je $F = \{q_0, s_0\}$, pokud $q_0 \in F_1$, jinak $F = \{q_0\}$ (záleží na tom, zda do původního jazyka patřilo ε). Přechodová funkce:

$$\delta(r,x) = \begin{cases} p \ : \ \delta_1(p,x) = r, x \in \Sigma, \\ t \ : \ \delta_1(t,x) \in F_1, \text{ pokud } r = s_0, x \in \Sigma. \end{cases}$$

Na přechodové funkci vidíme, že podle prvního řádku předpisu se všechny přechody obrátí (tj. zamění se zdroj a cíl přechodu), podle druhého řádku vytvoříme přechody vedoucí z přidaného stavu s_0 (zastupujícího původní koncové stavy).

Příklad

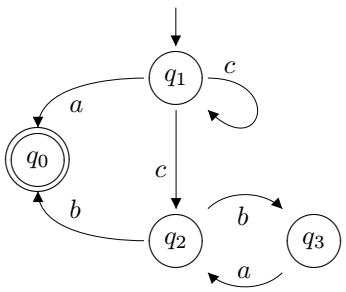
Provedeme operaci zrcadlení jazyka tohoto konečného automatu:
$\mathcal{A}_1 = (\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \delta_1, q_0, \{q_1\})$



	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	

$\delta_1(q_0, a) = q_1$
$\delta_1(q_0, b) = q_2$
$\delta_1(q_1, c) = q_1$
$\delta_1(q_2, a) = q_3$
$\delta_1(q_2, c) = q_1$
$\delta_1(q_3, b) = q_2$

Obrátíme všechny přechody. Protože máme jen jediný koncový stav, stačí pak jen zaměnit počáteční a koncový stav (nebudeme vytvářet nový stav s_0). U tabulky se zaměňují označení řádků s obsahem buněk na tomto řádku.
$\mathcal{A}_R = (\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \delta, q_1, \{q_0\})$



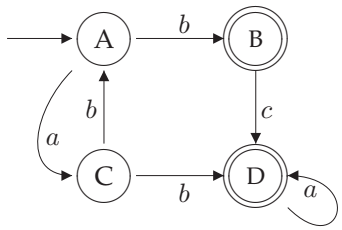
	a	b	c
$\leftarrow q_0$			
$\rightarrow q_1$	q_0		q_1, q_2
q_2		q_0, q_3	
q_3	q_2		

$\delta_1(q_1, a) = q_0$
$\delta_1(q_2, b) = q_0$
$\delta_1(q_1, c) = q_1$
$\delta_1(q_3, a) = q_2$
$\delta_1(q_1, c) = q_2$
$\delta_1(q_2, b) = q_3$

Ukázka odvození slova v automatu $\mathcal{A}_1$ a jeho reverze v automatu $\mathcal{A}_R$:
$\mathcal{A}_1$: $(q_0, babc) \vdash (q_2, abc) \vdash (q_3, bc) \vdash (q_2, c) \vdash (q_1, \varepsilon)$
$\mathcal{A}_R$: $(q_1, cbab) \vdash (q_2, bab) \vdash (q_3, ab) \vdash (q_2, b) \vdash (q_0, \varepsilon)$

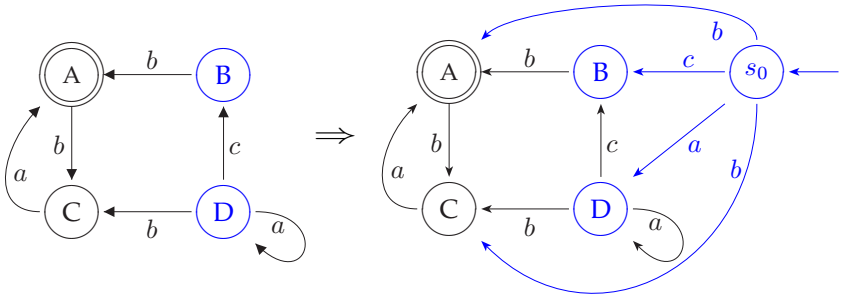
Příklad

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
$\rightarrow$ s_0	D	A, C	B
$\leftarrow$ A		C	
B		A	
C	A		
D	D	C	B

Ukázka odvození slova v automatu $\mathcal{A}_1$ a jeho reverze v automatu $\mathcal{A}_R$:

$\mathcal{A}_1$: $(q_0, babb) \vdash (q_2, abb) \vdash (q_0, bb) \vdash (q_2, b) \vdash (q_1, \varepsilon)$

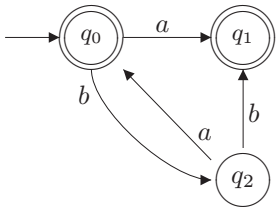
$\mathcal{A}_R$: $(s_0, bbab) \vdash (q_2, bab) \vdash (q_0, ab) \vdash (q_2, b) \vdash (q_0, \varepsilon)$

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

Příklad

Pro operaci zrcadlení je dán tento konečný automat:

$\mathcal{A}_1 = (\{q_0, q_1, q_2\}, \{a, b\}, \delta_1, q_0, \{q_0, q_1\})$

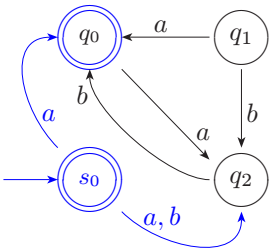


	a	b
$\leftrightarrow$ q_0	q_1	q_2
$\leftarrow$ q_1		
q_2	q_0	q_1

$\delta_1(q_0, a) = q_1$
$\delta_1(q_0, b) = q_2$
$\delta_1(q_2, a) = q_0$
$\delta_1(q_2, b) = q_1$

Na rozdíl od předchozích příkladů budou ve výsledném automatu dva koncové stavy. Stav q_0 bude koncový, protože v původním automatu je počátečním stavem, a stav s_0 bude koncový, protože do jazyka původního automatu patří slovo ε, jehož převrácením je opět slovo ε pro jeho rozpoznání musí být počáteční stav (tj. s_0) v množině koncových stavů.

$\mathcal{A}_R = (\{s_0, q_0, q_1, q_2\}, \{a, b\}, \delta, s_0, \{q_0, s_0\})$

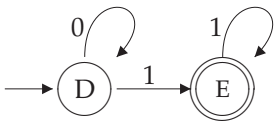


	a	b
$\leftrightarrow$ s_0	q_0, q_2	q_2
$\leftarrow$ q_0	q_2	
q_1	q_0	q_2
q_2		q_0

$\delta(s_0, a) = \{q_0, q_2\}$
$\delta(s_0, b) = \{q_2\}$
$\delta(q_1, a) = \{q_0\}$
$\delta(q_2, b) = \{q_0\}$
$\delta(q_0, a) = \{q_2\}$
$\delta(q_1, b) = \{q_2\}$

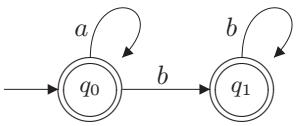
Úkoly

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 $L = \{\varepsilon, \text{tisk}, \text{tis}, \text{síto}\}$ 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

1.3.3 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 (pozor, záleží na pořadí).

Označme

- $\mathcal{A}_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ je první automat,
- $\mathcal{A}_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ je druhý automat,

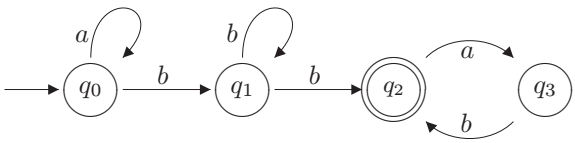
pak automat rozpoznávající jazyk, který je průnikem jazyků obou automatů, je
$\mathcal{A}_P = (Q_1 \times Q_2,\ \Sigma_1 \cap \Sigma_2,\ \delta,\ [q_1, q_2],\ F_1 \times F_2)$, kde operace $\times$ je kartézským součinem uvedených množin. Přechodová funkce:
$\delta\Big([p, r], x\Big) = \Big[\delta_1(p, x), \delta_2(r, x)\Big]$, kde $p \in Q_1, r \in Q_2, x \in \Sigma_2, x \in \Sigma_1 \cap \Sigma_2$

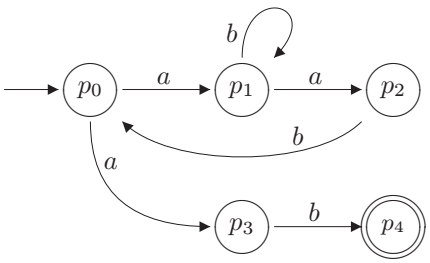
Poznámka:

Všimněte si, že v definici výsledného automatu máme kartézský součin původních množin stavů (a totéž pro koncové stavy). Ve skutečnosti mnoho z těchto stavů je nedostupných nebo nadbytečných, což můžeme vyřešit redukcí.

Příklad

Sestrojíme konečný automat rozpoznávající průnik jazyků následujících dvou automatů:





$\mathcal{A}_1 = (\{q_0, \ldots, q_3\}, \{a, b\}, \delta_1, q_0, \{q_2\})$

$\mathcal{A}_2 = (\{p_0, \ldots, p_4\}, \{a, b\}, \delta_2, p_0, \{p_4\})$

$\delta_1(q_0, a) = \{q_0\}$ $\quad$ $\delta_1(q_2, a) = \{q_3\}$

$\delta_1(q_0, b) = \{q_1\}$ $\quad$ $\delta_1(q_3, b) = \{q_2\}$

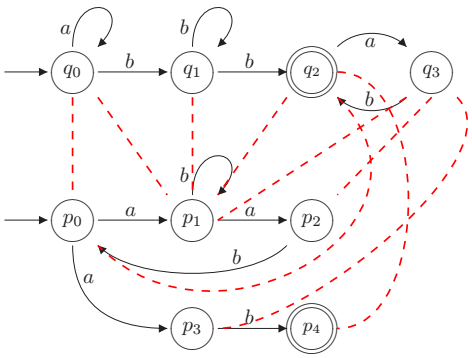
$\delta_1(q_1, b) = \{q_1, q_2\}$

$\delta_2(p_0, a) = \{p_1, p_3\}$ $\quad$ $\delta_2(p_2, b) = \{p_0\}$

$\delta_2(p_1, a) = \{p_2\}$ $\quad$ $\delta_2(p_3, b) = \{p_4\}$

$\delta_2(p_1, b) = \{p_1\}$

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 počá-
teční 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 $4 \times 5 = 20$ 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 předchozím semestru, 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. Zbývá odstranit nadbytečné stavy.

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):

$E_0 = \{[q_2, p_4]\}$

$E_1 = \{[q_2, p_4], [q_3, p_3]\}$

$E_2 = \{[q_2, p_4], [q_3, p_3], [q_2, p_0]\}$

$E_3 = \{[q_2, p_0], [q_2, p_4], [q_3, p_3], [q_3, p_2]\}$

$E_4 = \{[q_2, p_0], [q_2, p_1], [q_2, p_4], [q_3, p_2], [q_3, p_3]\}$

$E_5 = \{[q_2, p_0], [q_2, p_1], [q_2, p_4], [q_3, p_2], [q_3, p_3], [q_1, p_1], [q_3, p_1]\}$

$E_6 = \{[q_1, p_1], [q_2, p_0], [q_2, p_1], [q_2, p_4], [q_3, p_1], [q_3, p_2], [q_3, p_3], [q_0, p_1]\}$

$E_7 = \{[q_0, p_1], [q_1, p_1], [q_2, p_0], [q_2, p_1], [q_2, p_4], [q_3, p_1], [q_3, p_2], [q_3, p_3], [q_0, p_0]\}$

$E_8 = \{[q_0, p_0], [q_0, p_1], [q_1, p_1], [q_2, p_0], [q_2, p_1], [q_2, p_4], [q_3, p_1], [q_3, p_2], [q_3, p_3]\} = E_7$

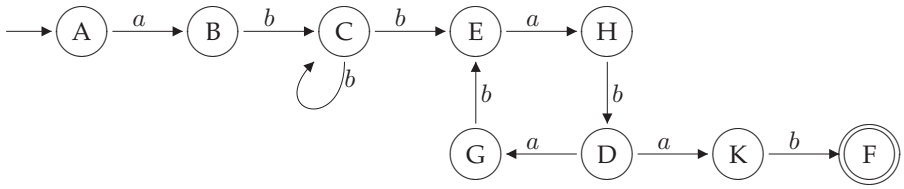
Po odstranění stavů $[q_0, p_2], [q_0, p_3], [q_1, p_0], [q_1, p_4]$ dostáváme tuto tabulku přechodů (s původním označením stavů a po přeznačení na písmena):

	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

Přeznačení stavů není povinné, zde jsme to provedli pouze kvůli zvýšení přehlednosti. Jak vidíme, získali jsme automat s devíti stavy.

Stavový diagram (stavy jsou přeznačené):



Úkoly

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 výše uvedeným postupem vytvořte automat rozpoznávající průnik těchto jazyků.
2. Sestrojte konečný automat, který rozpoznává průnik jazyků následujících automatů (pracujte s tabulkami přechodů). Odstraňte nedosažitelné a nadbytečné stavy a sestrojte stavový diagram.

	0	1
→ A	B	A
← B	C	
C		D
D	B	

	0	1
→ E		F
F	F	H
← 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.

1.4 Pumping lemma pro regulární jazyky

Pumping lemma nám pomáhá určit, zda není daný jazyk regulární. Základní znění lemmatu je následující:

Lemma (Pumping lemma pro regulární jazyky)

Jestliže L je regulární jazyk, pak existuje přirozené číslo $p > 0$ takové, že pro každé slovo $w \in L$ takové, že $|w| > p$, existuje alespoň jedno rozdělení slova w ve formě $w = x \cdot y \cdot z$, kde

$$y \neq \varepsilon \quad \wedge \quad |x \cdot y| \leq p \quad \wedge \quad \forall k \geq 0 \;\; je \;\; x \cdot y^k \cdot z \in L \tag{1.1}$$

V symbolickém zápisu:

Lemma (Pumping lemma pro regulární jazyky – symbolický zápis)

$$L \text{ je regulární jazyk} \quad \Rightarrow \quad \exists p \in \mathbb{N},\ p > 0 \text{ takové, že } \forall w \in L,\ |w| > p \ \exists \text{ rozdělení}$$
$$w = x \cdot y \cdot z \quad \wedge \quad |y| > 0 \quad \wedge \quad |x \cdot y| \leq p \quad \wedge \quad \forall k \geq 0 \;\; je \;\; x \cdot y^k \cdot z \in L \tag{1.2}$$

Co z toho vyplývá?

- Pumping lemma je *implikace* – říká, že *pokud* je jazyk regulární, *pak* má danou vlastnost. Ale nemusí nutně platit, že jazyk, který má danou vlastnost, je regulární (protože se nejedná o ekvivalenci).
- Jestliže chceme tuto větu použít pro důkaz toho, že daný jazyk není regulární, musíme znění upravit – srovnejte původní a „otočené“ znění:

$$L \in \mathscr{L}(REG) \Rightarrow \ \exists p > 0 \ \ \forall w\colon\ |w| > p \ \ \exists \text{ rozdělení} \ldots \ \forall k \geq 0\colon\ x \cdot y^k \cdot z \in L$$
$$\forall p > 0 \ \ \exists w\colon\ |w| > p \ \ \forall \text{ rozdělení} \ldots \ \exists k \geq 0\colon\ x \cdot y^k \cdot z \notin L \ \ \Rightarrow L \notin \mathscr{L}(REG)$$

  Takže najdeme *jedno* dostatečně dlouhé slovo, projdeme *všechna* jeho rozdělení $x \cdot y \cdot z$ vyhovující podmínkám lemmatu a pro každé rozdělení najdeme *jedno* číslo $k \geq 0$ pro pumpování.
- Lemma používáme na „dostatečně dlouhá slova“, tedy delší než p.
- Podmínka $|x \cdot y| \leq p$ je důležitá: dokáže nám hodně redukovat počet vyhovujících rozdělení slova. V praxi to znamená, že v částech x a y je omezený počet symbolů, tedy například se zde nesmí vyskytovat žádný index blížící se délce vybraného slova.

Poznámka:

Uvědomme si, z čeho věta vyplývá – pokud je jazyk regulární, pak existuje konečný automat, který tento jazyk rozpoznává. Když už máme konečný automat, můžeme si za číslo p dosadit jednoduše počet stavů automatu. Pak podmínka *dostatečně dlouhé slovo* (tj. $|w| > p$) znamená, že výpočet slova se bude sestávat z více než p kroků, a tedy musí existovat stav, přes který půjde tento výpočet více než jednou (protože v každém kroku výpočtu je zpracován právě jeden symbol slova a dochází ke změně stavu).

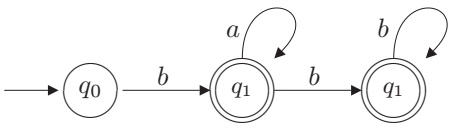
Pokud tedy výpočet prochází přes alespoň jeden stav dvakrát, znamená to, že ve výpočtu je smyčka (tj. v grafu automatu se ve smyčce dostáváme od prvního výskytu „opakovaného“ stavu ve výpočtu k jeho dalšímu výskytu).

Tedy v našem slově najdeme první smyčku ve výpočtu, část slova zpracovanou na cestě k této smyčce prohlásíme za x, část slova zpracovaného ve smyčce (jejím prvním průběhu) za y a zbytek slova bude z. Opravdu hledáme první smyčku na cestě a jako prostřední část rozdělení použijeme první průběh smyčky, to odpovídá podmínce $|x \cdot y| \leq p$.

Příklad

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

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 $p = 3$,
- vezmeme jakékoliv slovo jazyka delší než 3, například $w = ba^3b$ (má délku 5),

- 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 $w = (b)\cdot(a)\cdot(a^2b)$, po pumpování prostřední části dostáváme slova $(b)\cdot(a)^k\cdot(a^2b) = ba^{k+2}b$, pro jakékoliv $k \geq 0$ tato slova patří do jazyka L.

Mohli bychom zvolit jiné dlouhé slovo, například b^8. Odpovídající rozdělení by bylo $(bb)\cdot(b)\cdot(b^5)$.

Vidíme, že prostřední část slova vlastně odpovídá části slova, která prochází cyklem (nejméně jednou). Pumpování s indexem k pak odpovídá $k$násobnému průchodu cyklem v automatu.

Platí Pumping lemma pro všechny regulární jazyky včetně konečných? Samozřejmě ano. Pro každý konečný jazyk existuje konečný automat, který ho rozpoznává. Jako p si u konečného jazyka zvolíme opět buď počet stavů tohoto automatu anebo číslo ještě větší. V Pumping lemma se píše „pro každé slovo w daného jazyka delší než p ...“, 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

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 lemmatu a ukažte, že pro jakékoliv číslo k jde opět o slovo z daného jazyka.

Pracujte s těmito jazyky:

(a) $L_a = \{ab^ia \ : \ i \geq 0\}$
(b) $L_b = \{(ab)^i \ : \ i \geq 0\}$
(c) $L_c = \{a(bb)^i \ : \ i \geq 1\}$
(d) $L_d = \{000(111)^i \ : \ i \geq 0\}$
(e) $L_c = \{\text{disk, data, plat}\}$
(f) $L_d = \mathcal{N}$ (množina přirozených čísel, zde včetně 0)

Obvyklejším způsobem použití Pumping lemmatu 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

Ukážeme, že jazyk $L = \{a^n b^{2n} \ : \ n \geq 0\}$ 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 *konstantu* 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 *některé* $k \geq 0$ slovo $x \cdot y^k \cdot z$ nepatří do jazyka L. U jednotlivých možností budeme volit $k = 0$ nebo $k = 2$.

Hledáme taková rozdělení $w = x \cdot y \cdot z$, kde platí:

- $|x| > 0$,
- $|x \cdot y| \leq p$.

Ve skutečnosti můžeme všechna taková rozdělení popsat takto:

$x \cdot y \cdot z$	*Podmínky*	$x \cdot y^k \cdot z$	$k = 0$
$a^r \cdot a^s \cdot a^{i-r-s} b^{2i}$	$s > 0$	$a^{r+ks+i-r-s} b^{2i}$	$a^{i-s} b^{2i} \notin L$, protože $s > 0$

Ve schématu $a^r \cdot a^s \cdot a^{i-r-s} b^{2i}$ máme zakódována všechna rozdělení odpovídající podmínkám Pumping lemmatu, stačí za r, s dosazovat vhodná „malá“ čísla.

Pro každé možné rozdělení (tedy vlastně jediné „multirozdělení“) jsme našli k takové, že xy^kz nepatří do jazyka, proto L není regulární jazyk.

Poznámka:

Všimněte si, že ve schématu se index i nevyskytuje v prvních dvou částech slova. Protože $|x \cdot y| \leq p$ a víme, že $3i > p$, můžeme bez újmy na obecnosti prohlásit, že ve skutečnosti jsme zvolili dokonce $i > p$, a tedy by se nám do prvních dvou částí slova „nevešlo“. Takže si tímto způsobem usnadňujeme práci s hledáním vhodných rozdělení.

Příklad

Dokážeme, že jazyk $L = \{a^n \ : \ n$ je prvočíslo$\}$ není regulární.

Kdyby tento jazyk byl regulární, pak by pro každé dostatečně dlouhé slovo jazyka existovalo rozdělení s výše uvedenými vlastnostmi. Zvolme slovo $w = a^i$, kde i je některé dostatečně velké prvočíslo.

Nyní určíme rozdělení pro $w = x \cdot y \cdot z$ (v tomto případě není moc možností, jak volit). Zvolme tedy $x = a^r, \ y = a^s, \ z = a^{i-r-s}$, se zachováním podmínek lemmatu (to znamená, že $s > 0, \ r + s \leq p$).

Pro pumpování určíme index $k = i + 1$. To můžeme, protože v modifikaci věty máme $\forall k \geq 0$, a číslo i je z našeho pohledu konstantní (a dostatečně velké, určitě větší než 0).

Určíme tedy $w_k = a^{r+k\cdot s+i-r-s} = a^{k\cdot s+i-s}$, konkrétně $w_{i+1} = a^{(i+1)\cdot s+i-s} = a^{is+i} = a^{i\cdot(s+1)}$. Exponent lze rozložit na součin dvou čísel $i, s+1$, která jsou obě větší než 2, proto se nejedná o prvočíslo a slovo w_{i+1} nepatří do jazyka L. Z toho vyplývá, že L není regulární.

Poznámka:

Všimněte si, že jsme vlastně použili důkaz sporem. Předpokládali jsme, že daný jazyk je regulární, ale ke konci důkazu jsme tento předpoklad popřeli.

Příklad

Dokážeme, že jazyk $L = \left\{a^{n^2} \ : \ n \geq 1\right\}$ není regulární.

Vezměme si „dostatečně dlouhé“ slovo $w = a^{i^2}$ (pro dostatečně velkou konstantu i). Toto slovo rozdělíme na tři části následovně: $a^{i^2} = a^{x_1} \cdot a^{x_2} \cdot a^{x_3}$, tedy platí následující rovnice a nerovnice:

- $i^2 = x_1 + x_2 + x_3$
- $x_2 > 0$
- $x_1 + x_2 \leq p$

Pumpování: $w_k = a^{x_1} \cdot a^{k\cdot x_2} \cdot a^{x_3}$.

Předpokládejme, že jazyk L je regulární. Pak by muselo platit, že počet symbolů a ve slově w_k je druhou mocninou některého přirozeného čísla, tedy $a^{x_1} \cdot a^{k\cdot x_2} \cdot a^{x_3} = a^{m^2}$ pro některé číslo $m \geq 1$. Z toho vyplývá následující vztah: $x_1 + k \cdot x_2 + x_3 = m^2$.

Uvědomte si, co je zde konstantou a co je proměnnou: čísla i, x_1, x_2, x_3 jsou pro nás konstantami, protože je napevno volíme (pro jedno dostatečně dlouhé slovo a konkrétní rozdělení tohoto slova. Naproti tomu k je proměnná (protože u regulárního jazyka by vztah měl platit pro všechna $k \geq 0$), a m je proměnná závislá na proměnné k (když roste hodnota k, roste i hodnota proměnné m).

Vztah $x_1+k\cdot x_2+x_3 = m^2$ můžeme tedy přepsat takto: $k\cdot x_2+(x_1+x_3) = m^2$, přičemž na levé straně rovnice máme lineární funkci proměnné k, na pravé straně rovnice máme mocninnou funkci (závislé) proměnné m, přičemž se jedná o funkce na množině přirozených čísel (nebo celých nezáporných čísel).

Pokud by se opravdu jednalo o regulární jazyk, pak by muselo být možné ke každé hodnotě k najít odpovídající hodnotu pro proměnnou m (tedy každé slovo w_k by mělo patřit do jazyka L), ale jak víme, lineární a mocninná funkce na kladné poloose se kryjí v maximálně dvou bodech $\Rightarrow$ spor. Proto jazyk L není regulární.

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

Jazyk $L = \left\{a^i b^j c^{2j} \ : \ i, j \geq 0\right\}$ není regulární, přesto splňuje Pumping lemma. Můžeme zvolit například slovo $w = a^m$ pro některé dostatečně velké číslo m.

Toto slovo lze rozdělit na tři části $x \cdot y \cdot z = a^r a^s a^{m-r-s}$ při zachování podmínek $s > 0,\ r + s \leq p$ a pumpovat na $w_k = a^{r+k\cdot s+m-r-s} = a^{k\cdot s+m-s}$, přičemž i po pumpování po dosazení jakéhokoliv celého nezáporného čísla za k dostaneme opět slovo patřící do jazyka L. Takže jazyk L splňuje podmínky Pumping lemmatu, třebaže se nejedná o regulární jazyk.

Poznámka:

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.

Úkoly

1. Ukažte, že jazyk $L = \{ba(ab)^n \ : \ n \geq 0\}$ splňuje vlastnosti uvedené v Pumping lemmatu.
2. Pomocí Pumping lemmatu dokažte, že jazyk $L = \{a^n bc^n \ : \ n \geq 1\}$ není regulární.
3. Pomocí Pumping lemmatu dokažte, že jazyk $L = \left\{ww^R \ : \ w \in \{a, b\}^*\right\}$ není regulární. Můžete zvolit například slovo $v = a^i b^{2i} a^i$, které také patří do jazyka L, stačí dokázat, že pro toto slovo neexistuje vhodné rozdělení pro pumpování.
4. Pomocí Pumping lemmatu dokažte, že jazyk $L = \left\{ba^{2^n} \ : \ n \geq 0\right\}$ není regulární.

1.5 Minimalizace konečného automatu

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

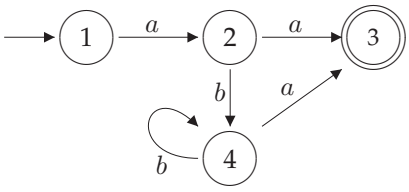
Při minimalizaci automatu $\mathcal{A}$ postupujeme takto:

- automat $\mathcal{A}$ by měl být deterministický; pokud není, převedeme ho do deterministického tvaru,
- odstraníme nedostupné stavy (tj. stavy, do kterých nevede cesta z počátečního stavu) – první část redukce stavů automatu (druhou část nemusíme provádět, je víceméně součástí algoritmu minimalizace),
- pokud automat není totální, zúplníme ho (tj. přidáme „odpadkový koš“),
- dále pracujeme se třídami ekvivalence:
  - vytvoříme pomocnou tabulku, ve které budeme pro přehlednost označovat třídy římskými čísly,
  - rozdělíme stavy do dvou skupin – nekoncové (třída I) a koncové (třída II), takto označíme řádky pomocné tabulky, sloupce budou označeny prvky množiny Σ jako v původní přechodové tabulce,
  - doplníme obsah buněk tabulky: buď I (pokud se z daného stavu – řádek – přechází na daný symbol – sloupec – do stavu ze třídy I) nebo II (pokud se z daného stavu na daný symbol přechází do stavu patřícího do třídy II),

– srovnáme řádky v rámci jednotlivých tříd: pokud řádky v rámci třídy mají stejný obsah, tuto třídu pro daný krok nedělíme, ale pokud se řádky uvnitř třídy liší obsahem buněk, rozdělíme třídu na „podtřídy“, obsah buněk přizpůsobíme novému rozdělení,
– opakujeme předchozí bod tak dlouho, dokud je co dělit.

Příklad

Minimalizujeme následující konečný automat:



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

Automat je deterministický, ale není totální.

$\Rightarrow$ Automat zúplníme:

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

Sestrojíme pomocnou tabulku se třídami:

třída		a	b
I	$\rightarrow 1$	I	I
	2	II	I
	4	II	I
	X	I	I
II	$\leftarrow 3$	I	I

V přechodové tabulce máme na řádku pro stav 1 v buňkách stavy 2 a X, přičemž oba patří do třídy I; proto do pomocné tabulky dáme do obou buněk označení třídy I.

Na řádku přechodové tabulky pro stav 2 vidíme, že se přechází do stavů 3 a 4, které patří do tříd II a I. Tedy do pomocné tabulky dáme označení tříd II a I.

Podobně pro další stavy – podle přechodové tabulky a obsahu tříd.

Jak vidíme, z řádků v první třídě mají stejný obsah 1 a X, a pak řádky 2 a 4. Proto třídu I rozdělíme na dvě části (stavy 1 a X necháme ve třídě I, stavy 2 a 4 přesuneme do nové třídy III). Protože se změnila příslušnost stavů ve třídách, musíme změnit i obsah buněk, čímž vytvoříme novou verzi pomocné tabulky:

třída		a	b
I	$\rightarrow 1$	III	I
	X	I	I
III	2	II	III
	4	II	III
II	$\leftarrow 3$	I	I

Postupujeme stejně jako u první verze pomocné tabulky – podle původní přechodové tabulky a momentálního obsahu tříd. Ze stavu 1 přecházíme do stavů 2 a X, přičemž první z nich teď patří do třídy III a druhý do třídy I, tedy v buňkách budou III a I.

Takto zpracujeme celou tabulku, včetně těch řádků, jejichž třídy jsme v tomto kroku nedělili (tady nám náhodou vychází na řádku označeném 3 tentýž obsah, ale mohlo by to vyjít jinak).

V dalším kroku postupujeme obdobně. Ve třídě I máme dva řádky, každý z nich má jiný obsah buněk, tedy ji rozdělíme. Třídu III nebudeme dělit, její řádky mají stejný obsah buněk.

třída		a	b
I	$\rightarrow 1$	III	IV
IV	X	IV	IV
III	2	II	III
	4	II	III
II	$\leftarrow 3$	IV	IV

Po rozdělení třídy I opět přepíšeme obsah buněk tak, aby odpovídal momentálnímu rozdělení stavů do tříd.

Při pohledu na pomocnou tabulku zjistíme, že máme čtyři třídy, přičemž ve třídě III jsou sice dva stavy, ale obsah buněk na jejich řádcích je stejný, tedy je nebudeme dělit. Ostatní třídy jsou jednoprvkové. Z pomocné tabulky je zřejmé, že stavy 2 a 4 jsou navzájem zastupitelné.

Algoritmus končí a zbývá sestavit výslednou přechodovou tabulku. Máme dvě možnosti: buď použijeme původní označení stavů a stavy 2 a 4 sloučíme do jediného (pro který zvolíme například označení 2), nebo prostě jako označení stavů použijeme názvy tříd.

Původní označení stavů:

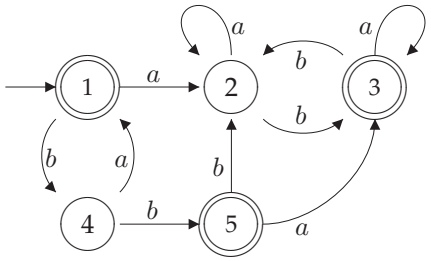
	a	b
$\rightarrow 1$	2	X
X	X	X
2	3	2
$\leftarrow 3$	X	X

Označení stavů třídami:

	a	b
$\rightarrow$ I	III	IV
IV	IV	IV
III	II	III
$\leftarrow$ II	IV	IV

Příklad

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

Automat je deterministický a úplný, tedy nemusíme provádět žádné předběžné úpravy. Vytvoříme sekvenci pomocných tabulek, ve kterých budeme pracovat se třídami:

třída		a	b
I	2	I	II
	4	II	II
II	$\leftrightarrow 1$	I	I
	$\leftarrow 3$	II	I
	$\leftarrow 5$	II	I

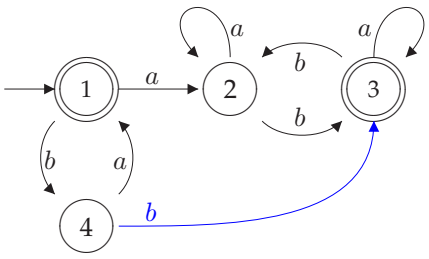
$\Rightarrow$

třída		a	b
I	2	I	IV
III	4	II	IV
II	$\leftrightarrow 1$	I	III
IV	$\leftarrow 3$	IV	I
	$\leftarrow 5$	IV	I

V druhém kroku jsme rozdělili obě původní třídy, máme tedy celkem čtyři různé třídy. Další dělení už se provádět nebude, protože pouze ve třídě IV máme víc než jeden stav, a oba stavy z této třídy mají shodný obsah buněk na řádku. Zbývá vytvořit výsledný automat:

Původní označení stavů:

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



Označení stavů třídami:

	a	b
$\leftrightarrow$ II	I	III
I	I	IV
$\leftarrow$ IV	IV	I
III	II	IV

Příklad

Minimalizujeme následující konečný automat:

	a	b
$\leftrightarrow 1$	2	6
2	4	5
3	6	3
4	1	4
5	3	2
$\leftarrow 6$	5	1

Automat je deterministický a úplný, nemusíme provádět žádné předběžné úpravy. Vytvoříme sekvenci pomocných tabulek, ve kterých budeme pracovat se třídami:

třída		a	b
I	2	I	I
	3	II	I
	4	II	I
	5	I	I
II	$\leftrightarrow 1$	I	II
	$\leftarrow 6$	I	II

$\Rightarrow$

třída		a	b
I	2	III	I
	5	III	I
III	3	II	III
	4	II	III
II	$\leftrightarrow 1$	I	II
	$\leftarrow 6$	I	II

Máme tři třídy, v každé dva stavy, ale dál dělit nemůžeme, protože v rámci tříd je obsah buněk na různých řádcích shodný. Výsledný automat má tři stavy:

Původní označení stavů:

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

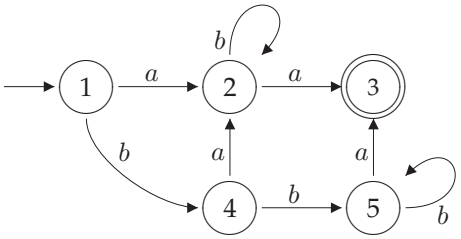
Označení stavů třídami:

	a	b
$\leftrightarrow$ II	I	II
I	III	I
III	II	III

Zatímco v předchozích příkladech se víceméně dalo předem odhadnout, které stavy budou navzájem zaměnitelné, tady už to tak zřejmé nebylo. Podařilo se nám minimalizovat automat na polovinu původního počtu stavů a zůstal nám jen jeden koncový stav.

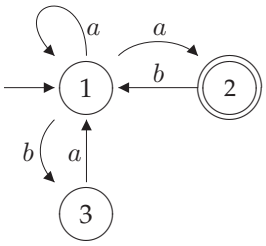
Úkoly

1. Minimalizujte následující konečný automat (nezapomeňte ho předem upravit – není totální):



	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), podle potřeby převeďte na totální,
   - minimalizujte ho.



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

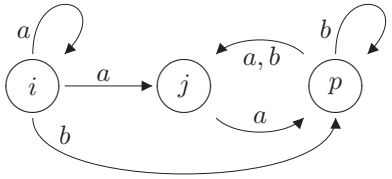
1.6 Vytvoření regulárního výrazu podle konečného automatu

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^0_{ij} a dokážeme je jednoduše zjistit přímo z automatu (v jakékoliv reprezentaci), například:

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



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

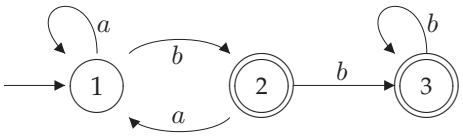
3. Indukcí budeme postupně cesty prodlužovat (spojovat). Vytváříme množiny R^k_{ij} – je to regulární výraz odpovídající cestě ze stavu i do stavu j vedoucí pouze přes stavy označené maximálně číslem k. Postupujeme podle vzorce:

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

   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

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^0_{ij} – regulární výrazy pro nejkratší cesty bez mezistavů.

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

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 $(a+\varepsilon)\cdot a^* = a^*$.

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

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.

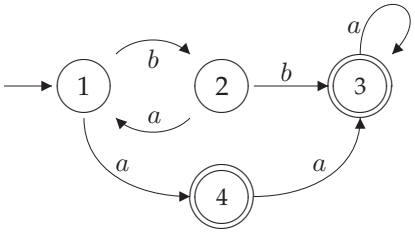
$R^3_{12} = a^*b(aa^*b)^*+a^*b(aa^*b)^*b\cdot(b+\varepsilon)^*\cdot\emptyset = a^*b(aa^*b)^*$
$R^3_{13} = a^*b(aa^*b)^*b+a^*b(aa^*b)^*b\cdot(b+\varepsilon)^*\cdot(b+\varepsilon) = a^*b(aa^*b)^*b+a^*b(aa^*b)^*b\cdot b^* =$
$\quad = a^*b(aa^*b)^*b\cdot(\varepsilon+b^*) = a^*b(aa^*b)^*bb^*$

Výsledkem je regulární výraz odpovídající jazyku rozpoznávanému daným automatem:

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

Příklad

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:

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

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

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

$$\begin{aligned} R_{11}^2 &= \varepsilon + b(ab)^*a = (ba)^* & R_{31}^2 &= \emptyset + \emptyset = \emptyset \\ R_{12}^2 &= b + b(ab)^*(\varepsilon + ab) = b(ab)^* & R_{32}^2 &= \emptyset \\ R_{13}^2 &= \emptyset + b(ab)^*b = b(ab)^*b & R_{33}^2 &= a + \varepsilon + \emptyset = a + \varepsilon \\ R_{14}^2 &= a + b(ab)^*aa = a + ba(ba)^*a = (ba)^*a & R_{34}^2 &= \emptyset \\ R_{21}^2 &= a + (\varepsilon + ab)(ab)^*a = (ab)^*a & R_{41}^2 &= \emptyset \\ R_{22}^2 &= (ab)^* & R_{42}^2 &= \emptyset \\ R_{23}^2 &= b + (\varepsilon + ab)(ab)^*b = (ab)^*b & R_{43}^2 &= a + \emptyset = a \\ R_{24}^2 &= aa + (\varepsilon + ab)(ab)^*aa = (ab)^*aa & R_{44}^2 &= \varepsilon + \emptyset = \varepsilon \end{aligned}$$

$$\begin{aligned} R_{11}^3 &= (ba)^* + \emptyset = (ba)^* & R_{21}^3 &= (ab)^*a + \emptyset = (ab)^*a \\ R_{12}^3 &= b(ab)^* + \emptyset = b(ab)^* & R_{22}^3 &= (ab)^* + \emptyset = (ab)^* \\ R_{13}^3 &= b(ab)^*b + b(ab)^*ba^*(a + \varepsilon) = b(ab)^*ba^* & R_{23}^3 &= (ab)^*b + (ab)^*ba^*(a + \varepsilon) = (ab)^*ba^* \\ R_{14}^3 &= (ba)^*a + \emptyset = (ba)^*a & R_{24}^3 &= (ab)^*aa + \emptyset = (ab)^*aa \end{aligned}$$

$R^3_{31} = \emptyset$
$R^3_{32} = \emptyset$
$R^3_{33} = a + \varepsilon + (a + \varepsilon)a^*(a + \varepsilon) = a^*$
$R^3_{34} = \emptyset$

$R^3_{41} = \emptyset$
$R^3_{42} = \emptyset$
$R^3_{43} = a + aa^*(a + \varepsilon) = aa^*$
$R^3_{44} = \varepsilon$

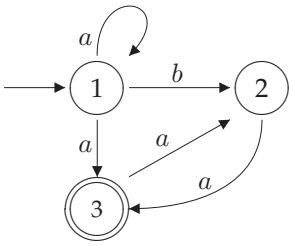
Ve čtvrtém kroku nás zajímají pouze cesty vedoucí z počátečního stavu do koncových stavů:
$R^4_{13} = b(ab)^*ba^* + (ba)^*a \cdot \varepsilon^* \cdot aa^* = b(ab)^*ba^* + (ba)^*aaa^* = (ba)^*bba^* + (ba)^*aaa^*$
$R^4_{14} = (ba)^*a + (ba)^*a \cdot \varepsilon^* \cdot \varepsilon = (ba)^*a$

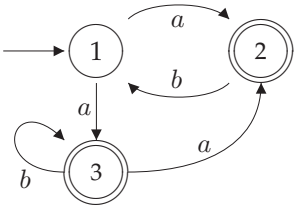
Regulární výraz odpovídající jazyku zadaného automatu je

$$R = R^4_{13} + R^4_{14} = (ba)^*bba^* + (ba)^*aaa^* + (ba)^*a = (ba)^*(bba^* + aaa^* + a)$$

Úkol

Zjistěte regulární výraz odpovídající jazyku následujících konečných automatů:





Poznámka:

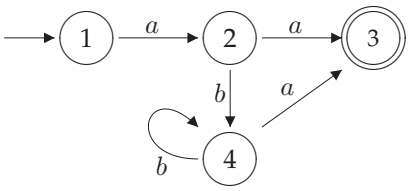
V předchozí sekci jsme pro minimalizaci automatu použili postup využívající poznatky z algebry. Jinou možností by byl následující postup:

- vezmeme deterministický konečný automat a vytvoříme k němu alternativní automaty takové, že jejich definice bude naprosto stejná, až na počáteční stavy: za počáteční stavy v těchto alternativních automatech budeme považovat postupně všechny stavy automatu (tedy získáme tolik alternativních automatů, kolik je stavů v množině Q),
- sestrojíme regulární výrazy odpovídající jazykům postupně všech alternativních automatů,
- pokud regulární výrazy pro některé stavy (coby počáteční v alternativním automatu) budou ekvivalentní, pak je prohlásíme za zaměnitelné a sloučíme.

Určitě je všem jasné, že tento postup je výrazně náročnější než ten využívající poznatky z algebry, což si ukážeme na příkladu.

Příklad

Minimalizujeme následující konečný automat – je to tentýž jako v příkladu na straně 19.



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

Potřebujeme zjistit regulární výrazy odpovídající cestám z jednotlivých stavů do koncových stavů. Máme čtyři stavy, dojdeme tedy ke čtyřem regulárním výrazům.

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

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

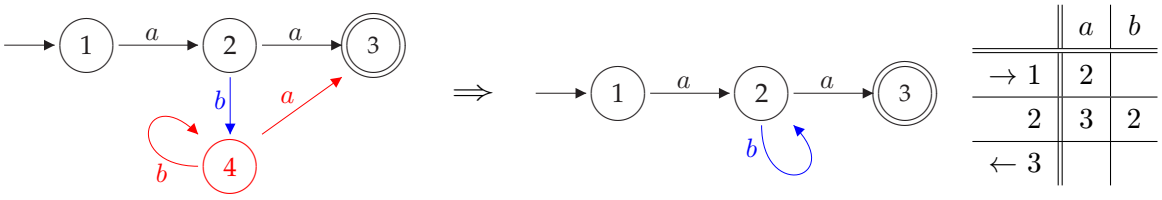
$R^2_{11} = \varepsilon$
$R^2_{12} = a$
$R^2_{13} = aa$
$R^2_{14} = ab$
$R^2_{21} = \emptyset$
$R^2_{22} = \varepsilon$
$R^2_{23} = a$
$R^2_{24} = b$
$R^2_{31} = \emptyset$
$R^2_{32} = \emptyset$
$R^2_{33} = \varepsilon$
$R^2_{34} = \emptyset$
$R^2_{41} = \emptyset$
$R^2_{42} = \emptyset$
$R^2_{43} =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

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

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
$\to 1$	2	
2	3	2
$\leftarrow 3$		

Výhodou je, že automat nemusí být totální, nevýhodou je složitá práce s regulárními výrazy, především jejich porovnávání není jednoduché.

Kapitola 2

Bezkontextové gramatiky

2.1 Úpravy bezkontextových gramatik

2.1.1 Převod na nezkracující bezkontextovou gramatiku

 Nezkracující bezkontextová gramatika buď vůbec neobsahuje ε-pravidla (pokud v jazyce generovaném gramatikou není prázdné slovo), anebo existuje jediné takové pravidlo, a to pro startovací symbol gramatiky, pak ale pak 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. Kdybychom však pouze simulovali použití ε-pravidel, mohli bychom se dostat do rekurze, které se však chceme vyhnout.

 Je dána gramatika $G = (N, T, P, S)$. Vytvoříme množinu N_ε všech neterminálů, které lze (třeba i po více než jednom kroku) přepsat na prázdné slovo ε.

Postup je iterační: množina $N_{\varepsilon,0}$ (báze) bude obsahovat právě ty neterminály, pro které existuje ε-pravidlo. 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.
$N_{\varepsilon,i} = N_{\varepsilon,i-1} \cup \{X \in N \ : \ X \to \alpha, \ \alpha \in N_{\varepsilon,i-1}\}$.

Vytvořenou množinu N_ε použijeme následovně:

- postupně zpracujeme všechna pravidla gramatiky kromě epsilonových, ta odstraníme (ignorujeme), a to dle následujících bodů,
- na pravé straně pravidla hledáme všechny výskyty symbolů patřících do N_ε,
- do výsledné množiny pravidel přidáme původní pravidlo a pak všechny jeho varianty, ve kterých je odstraněn jeden, dva, tři,... výskyty symbolů z N_ε (postupně všechny možné variace).

Takto vytvořenou množinu pravidel označme P'.

Pokud v jazyce $L(G)$ není slovo ε, jsme hotovi a výsledná gramatika je $G' = (N, T, P', S)$. Pokud však v $L(G)$ je prázdné slovo, musíme ho zařadit i do jazyka upravené gramatiky:

- přidáme nový neterminál S', $S' \notin N$, který se stane novým startovacím symbolem,
- přidáme dvě nová pravidla $S' \to S \mid \varepsilon$.

Výsledná gramatika pak je $G' = (N \cup \{S'\}, T, P' \cup \{S' \to S \mid \varepsilon\}, S')$.

Příklad

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:

$N_{\varepsilon,0} = \{B, C\}$ podle $B \to \varepsilon,\ C \to \varepsilon$
$N_{\varepsilon,1} = \{B, C, A\}$ podle $A \to BC$
$N_{\varepsilon,2} = \{B, C, A, D\}$ podle $D \to AAB$

V dalším kroku již nelze další neterminál přidat (ostatně všechny už v množině jsou), proto získáme $N_\varepsilon = N_{\varepsilon,2} = \{B, C, A, D\}$.

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.

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

Podívejme se na následující gramatiku. Na první pohled se může zdát, že si vystačíme s postupem použitým v předchozím příkladu. Ovšem ve skutečnosti do jazyka generovaného gramatikou patří prázdné slovo, tedy bude nutná dodatečná úprava, aby tuto vlastnost měla i upravená gramatika.

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

Opět rekurzívně vytvoříme množinu N_ε:

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

Sestrojíme množinu pravidel P' (nově přidaná pravidla jsou zbarvena modře):

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

Podle toho, že v množině N_ε je také startovací symbol gramatiky S, poznáme, že do jazyka gramatiky $L(G)$ patří i prázdné slovo. Proto je třeba přidat nový startovací symbol S' a pro něj dvě pravidla – první, kterým se napojíme na původní startovací symbol, a druhé epsilonové.
$G' = (\{S', S, A, B\},\ \{a, b\},\ P'',\ S')$ s pravidly
$S' \to S \mid \varepsilon$
$S \to AB \mid A \mid B \mid aS \mid a \mid bB \mid b$
$A \to baB \mid ba$
$B \to AA \mid A \mid b$

Úkol

K následujícím gramatikám vytvořte ekvivalentní nezkracující gramatiky.

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

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

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

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

$G_5 = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
$S \to aABa \mid bS \mid b$
$A \to aAA \mid bB \mid \varepsilon$
$B \to bB \mid aS \mid b$

$G_6 = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
$S \to aAA \mid bB \mid \varepsilon$
$A \to AaB \mid a \mid \varepsilon$
$B \to BbAA \mid a$

2.1.2 Redukce gramatiky

Účelem redukce gramatiky je snížení počtu symbolů a případně pravidel při zachování generovaného jazyka (tedy můžeme provádět pouze ekvivalentní úpravy). 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.

V obou případech rekurzívně (použijeme indukci) tvoříme množinu obsahující všechny symboly, které v gramatice mají zůstat. Formální zápis pro gramatiku $G = (N, T, P, S)$:

- Odstraňujeme nadbytečné neterminály:
  $E_0 = T$ (báze indukce – použijeme množinu terminálů)
  $E_i = E_{i-1} \cup \{A \in N\ :\ (A \to \alpha) \in P,\ \alpha \in E_{i-1}^*\}$
  (do následující množiny vložíme obsah předchozí množiny a přidáme všechny symboly, pro které

existuje pravidlo, kde na pravé straně jsou pouze symboly z množiny z předchozího kroku, *pozor, α může být i ε – prázdné slovo!*)

Končíme tehdy, když v daném kroku nepřidáme žádný symbol, $E_{k+1} = E_k = E_{def}$, kde k je nejmenší takový index, pro který daná rovnost platí. Pak zjistíme novou množinu neterminálů:
$N' = N \cap E_{def}$ (protože v množině E_{def} jsou také terminály).

- Odstraňujeme nedostupné symboly (terminální i neterminální):
  $S_0 = \{S\}$ (báze indukce, začínáme startovacím symbolem)
  $S_i = S_{i-1} \cup \{X \in (N \cup T) \ : \ (A \to \alpha X \beta) \in P, \ A \in S_{i-1}\}$
  (přidáme všechny symboly, které se nacházejí na pravé straně pravidel přepisujících již zařazené neterminály)

  Končíme opět tehdy, když nedochází ke změnám, $S_{r+1} = S_r = S_{def}$, kde r je nejmenší takový index, pro který daná rovnost platí. Pak zjistíme nové množiny terminálů a neterminálů:
  $N'' = N' \cap S_{def}$
  $T' = T \cap S_{def}$

Příklad

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).
$E_0 = \{a, b, c, d\}$
$E_1 = \{a, b, c, d, S, B, D\}$ (podle pravidel $S \to \varepsilon, \ B \to d, \ D \to \varepsilon$)
$E_2 = \{a, b, c, d, S, B, D\}$ (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.

$S_0 = \{S\}$
$S_1 = \{S, b, D\}$ (podle pravidel $S \to bS \mid bD$)
$S_2 = \{S, b, D, c, a\}$ (podle pravidel $D \to cDb \mid aS$)
$S_3 = S_2$

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

Redukujte tyto gramatiky:

$G_1 = (\{S, A, B, C, D\},\ \{a, b, c, d\},\ P,\ S)$
$S \to aBa \mid bA \mid c \mid ABc$
$A \to aA \mid dD \mid bDa$
$B \to aB \mid bA \mid bS$
$C \to aSc \mid cC \mid c$
$D \to dD \mid aAb$

$G_2 = (\{S, A, B, C, D, E\},\ \{0, 1, 2, 3\},\ P,\ S)$
$S \to AB0 \mid 2E \mid A1 \mid \varepsilon$
$A \to 0A0 \mid 1B \mid 1S$
$B \to 01B \mid 1D \mid EB$
$C \to 2A \mid 1S \mid 0$
$D \to D1 \mid 3B$
$E \to 0E1 \mid 1B$

$G_3 = (\{S, A, B, C, D\},\ \{a, b, c\},\ P,\ S)$
$S \to aD \mid bBD \mid c$
$A \to bbB \mid aA$
$B \to bB \mid aAb$
$C \to aD \mid b$
$D \to cS \mid aDD \mid \varepsilon$

$G_4 = (\{S, A, B, C, D\},\ \{a, b, c\},\ P,\ S)$
$S \to BaD \mid a$
$A \to aS \mid cc$
$B \to bSB \mid DC \mid \varepsilon$
$C \to aSDC \mid bSC$
$D \to aC \mid Da \mid BS$

2.1.3 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 analýze či programování.

Poznámka:

Případná ε-pravidla by nám mohla zajistit efekt jednoduchých pravidel dodatečně (například pokud máme pravidla $A \to BC,\ B \to \varepsilon$, pak je možná derivace $A \Rightarrow BC \Rightarrow C$, což je vlastně totéž, jako když použijeme pravidlo $A \to C$), je třeba předem převést gramatiku na nezkracující.

Pokud jednoduchá pravidla 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 indexu 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

Odstraníme jednoduchá pravidla v následující gramatice:
$G = (\{S, A, B, C\},\ \{a, b\},\ P,\ S)$
$S \to aBa \mid A$
$A \to aA \mid B$
$B \to bB \mid AB \mid C \mid \varepsilon$
$C \to bA \mid b$

Nejdřív převedeme gramatiku do tvaru nezkracující gramatiky:

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

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

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

$N_{S,0} = \{S\}$
$N_{S,1} = \{S, A\}$ (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 (konkrétně: v pravidle bude před šipkou ten symbol, který máme ve spodním indexu označení množiny, za šipkou budou pravé strany pravidel pro neterminály z dané množiny z původní gramatiky, přičemž původní jednoduchá pravidla ignorujeme).

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

Podíváme se na pár derivací v původní gramatice G a nové gramatice G':

G: $S \Rightarrow A \Rightarrow B \Rightarrow C \Rightarrow bA \Rightarrow bb$ $\qquad$ G: $S \Rightarrow aBa \Rightarrow aa$
G': $S' \Rightarrow bA \Rightarrow bb$ $\qquad$ G': $S' \Rightarrow aa$

Úkol

Odstraňte jednoduchá pravidla z následujících gramatik:

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

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

2.1.4 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 $A \Rightarrow^+ \alpha A$, kde 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 (ano, čtete dobře, píše se tady o programování podle gramatiky, bližší informace v předmětu Překladače). 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.

Postup je následující:

1. Předběžné úpravy: převedeme gramatiku do tvaru vlastní gramatiky, tj.
   - převedeme do tvaru nezkracující gramatiky,
   - odstraníme jednoduchá pravidla,
   - odstraníme nadbytečné a nedostupné symboly.
2. Levou rekurzi převedeme na pravou rekurzi.

V druhém bodu vycházíme z této myšlenky: předpokládejme, že máme sadu pravidel pro tentýž neterminál, označme
$A \to A\alpha_1 \mid A\alpha_2 \mid \ldots A\alpha_n \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_k$
kde řetězce $\beta_i,\ 1 \leq i \leq k$ začínají jiným symbolem než A

Je zřejmé, že tato pravidla v rekurzi vygenerují řetězec, na jehož začátku bude některý z podřetězců β_i a bude následovat libovolný počet podřetězců α_j, například:
$A \Rightarrow A\alpha_3 \Rightarrow A\alpha_5\alpha_3 \Rightarrow A\alpha_2\alpha_5\alpha_3 \Rightarrow \beta_8\alpha_2\alpha_5\alpha_3$

Je třeba tedy sestavit pravidla taková, aby byly generovány tytéž řetězce, ale bez levé rekurze. Původní sadu pravidel nahradíme jinou sadou:
$A \to \beta_1 B \mid \beta_2 B \mid \ldots \mid \beta_k B \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_k$
$B \to \alpha_1 B \mid \alpha_2 B \mid \ldots \mid \alpha_n B \mid \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$
Předchozí derivaci podle původních pravidel by odpovídala následující derivace:
$A \Rightarrow \beta_8 B \Rightarrow \beta_8\alpha_2 B \Rightarrow \beta_8\alpha_2\alpha_5 B \Rightarrow \beta_8\alpha_2\alpha_5\alpha_3$

Vidíme, že je generován tentýž řetězec, jen v opačném pořadí – zleva místo zprava, tedy levou rekurzi jsme nahradili pravou rekurzí.

Výše uvedený postup použijeme tehdy, když nám vadí ε-pravidla. Pokud nevadí, můžeme postupovat takto:
Původní sada pravidel:
$A \to A\alpha_1 \mid A\alpha_2 \mid \ldots A\alpha_n \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_k$
Nahradíme je touto sadou pravidel:
$A \to \beta_1 B \mid \beta_2 B \mid \ldots \mid \beta_k B$
$B \to \alpha_1 B \mid \alpha_2 B \mid \ldots \mid \alpha_n B \mid \varepsilon$
Odpovídající derivace by byla následující:
$A \Rightarrow \beta_8 B \Rightarrow \beta_8\alpha_2 B \Rightarrow \beta_8\alpha_2\alpha_5 B \Rightarrow \beta_8\alpha_2\alpha_5\alpha_3 B \Rightarrow \beta_8\alpha_2\alpha_5\alpha_3$

Derivace pak bude o jeden krok delší, ale na druhou stranu máme oproti výsledku předchozího postupu zhruba poloviční počet pravidel.

Poznámka:

Pozor – provedením výše naznačených úprav můžeme opět získat levě rekurzivní pravidla, tedy postup je třeba provádět tak dlouho, dokud se nezbavíme levě rekurzivních pravidel.

Jinou možností je úprava algoritmu tak, aby řešil levou (nebo pravou) rekurzi obecně, nikoliv jen *přímou* levou (pravou) rekurzi.

Příklad

Následující gramatiku zbavíme *přímé* levé rekurze.

Gramatika je nezkracující, neobsahuje jednoduchá pravidla, neobsahuje ani nadbytečné a nedostupné symboly, můžeme tedy přímo přikročit k odstraňování levě rekurzivních pravidel.
$G = (\{S, A, B\},\ \{a, b, c\},\ P,\ S)$
$S \to aAb \mid b$
$A \to Aa \mid bB \mid ab$
$B \to Ba \mid Bb \mid ca$

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

Pravidla $B \to Ba \mid Bb \mid ca$ nahradíme pravidly
$B \to caB' \mid ca$
$B' \to aB' \mid bB' \mid a \mid b$

Vytvořili jsme tedy ekvivalentní gramatiku bez přímé levé rekurze:
$G' = (\{S, A, A', B, B'\},\ \{a, b, c\},\ P',\ S)$
$S \to aAb \mid b$
$A \to bBA' \mid abA' \mid bB \mid ab$
$A' \to aA' \mid a$
$B \to caB' \mid ca$
$B' \to aB' \mid bB' \mid a \mid b$

Jak bude gramatika vypadat, pokud použijeme druhý postup (s ε-pravidly)?
$G'' = (\{S, A, A', B, B'\},\ \{a, b, c\},\ P'',\ S)$
$S \to aAb \mid b$
$A \to bBA' \mid abA'$
$A' \to aA' \mid \varepsilon$
$B \to caB'$
$B' \to aB' \mid bB' \mid \varepsilon$

Příklad

Odstraníme 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:
$G''' = (\{S, A, A', B, B'\},\ \{a, b, c\},\ P''',\ S)$
$S \to bAa \mid c \mid Ba \mid a \mid b \mid Abb$
$A \to BaA' \mid aA' \mid bA' \mid Ba \mid a \mid b$
$A' \to bbA' \mid bb$
$B \to aBbB' \mid abB' \mid aB' \mid |bcB' \mid caB' \mid aBb \mid ab \mid bc \mid ca$
$B' \to bcB' \mid bc$

Alternativně druhým uvedeným postupem:
$G'''' = (\{S, A, A', B, B'\},\ \{a, b, c\},\ P'''',\ S)$
$S \to bAa \mid c \mid Ba \mid a \mid b \mid Abb$
$A \to BaA' \mid aA' \mid bA'$
$A' \to bbA' \mid \varepsilon$
$B \to aBbB' \mid abB' \mid aB' \mid bcB' \mid caB'$
$B' \to bcB' \mid \varepsilon$

Příklad

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:

$G' = (\{S, A, A', B, B'\},\ \{a, b, c\},\ P',\ S)$
$S \to aAb \mid AB$
$A \to A'c \mid A'Sc \mid c \mid Sc$
$A' \to A'ab \mid ab$
$B \to B'a \mid a$
$B' \to B'bb \mid B'c \mid bb \mid c$

Nebo při použití ε-pravidel:

$G'' = (\{S, A, A', B, B'\},\ \{a, b, c\},\ P'',\ S)$
$S \to aAb \mid AB$
$A \to A'c \mid A'Sc$
$A' \to A'ab \mid \varepsilon$
$B \to B'a$
$B' \to B'bb \mid B'c \mid \varepsilon$


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
     $A_i \to A_j abB$
     $A_j \to bDA_i a \mid CA_j a$
     První z uvedenených pravidel nahradíme pravidly $A_i \to bdA_i aabB \mid CA_j aabB$. 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

Odstraníme levou rekurzi včetně nepřímé 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:
  $B \to aABaA'aB' \mid bBaA'aB' \mid aA'aB' \mid aABaaB' \mid bBaaB' \mid aaB' \mid cB' \mid$
  $aABaA'a \mid bBaA'a \mid aA'a \mid aABaa \mid bBaa \mid aa \mid c$
  $B' \to bB' \mid b$

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

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

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

Úkoly

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

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

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

   $G_3 = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
   $S \to AB \mid a$
   $A \to AaB \mid Sb \mid ba$
   $B \to BB \mid b \mid Ab \mid \varepsilon$

   $G_4 = (\{E, T, F\},\ \{i, +, *, (,)\},\ P,\ S)$
   $E \to E + T \mid T$
   $T \to T * F \mid F$
   $F \to (E) \mid i$

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

   $G_1 = (\{S, A, B\},\ \{0, 1\},\ P,\ S)$

   $S \to 11A \mid 01B \mid \varepsilon$
   $A \to 11A \mid B01 \mid 10$
   $B \to 01B \mid 00AB \mid 1A1 \mid 01$

   $G_2 = (\{S, A, B\},\ \{a, b\},\ P,\ S)$

   $S \to aAb \mid abB \mid ASA$
   $A \to bA \mid B \mid \varepsilon$
   $B \to bbB \mid abAB \mid abb$

2.2 Normální formy bezkontextových gramatik

Normální formy (čehokoliv) slouží k usnadnění porovnávání a případných dalších úprav, normované gramatiky se také využívají pro zjednodušení některých důkazů. V případě bezkontextových gramatik lze využít Chomského normální formu (CNF) a Greibachové normální formu (GNF).

2.2.1 Chomského normální forma

 Gramatika $G = (N, T, P, S)$ v Chomského normální formě (značíme CNF) má všechna pravidla v jednom z následujících tvarů:

- $A \to BC$, tedy na pravé straně dva neterminály,
- $A \to a$, tedy na pravé straně jeden terminál, a nebo
- $S \to \varepsilon$ – 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), 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 ve tvaru $A \to B_1B_2B_3 \dots B_n,\ n > 2$, vytvoříme množinu pravidel

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

   Původní pravou stranu pravidla vlastně generujeme po symbolech ve směru zleva.

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

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

$S \to N_aX_1 \mid N_aX_3 \mid c$	$B \to N_bB \mid AS \mid N_aX_4 \mid N_aX_6 \mid c \mid a$	$A \to AN_b \mid b$
$X_1 \to BX_2$	$X_4 \to BX_5$	$N_a \to a$
$X_2 \to AN_c$	$X_5 \to AN_c$	$N_b \to b$
$X_3 \to BN_c$	$X_6 \to BN_c$	$N_c \to c$

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

Následující gramatiky převeďte do Chomského normální formy:

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

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

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

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

$G_5 = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
$S \to aAbB \mid \varepsilon$
$A \to aA \mid a \mid Bba$
$B \to b \mid aSb \mid A$

$G_6 = (\{S, A, B\},\ \{0, 1\},\ P,\ S)$
$S \to 0A1 \mid 1B0 \mid \varepsilon$
$A \to 00AB \mid 1$
$B \to 1S \mid 0$

2.2.2 Greibachové normální forma

Gramatika je v Greibachové normální formě (značíme GNF), 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 pravé strany všech pravidel, kterými 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

Následující gramatiku převedeme do Greibachové 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.
$G' = (\{S, A, B, C\},\ \{a, b, c\},\ P',\ S)$
$S \to aBba \mid baA \mid ba \mid \varepsilon$
$A \to BaAB \mid BaB \mid bb$
$B \to CBab \mid abc$
$C \to bCb \mid a$

Levá rekurze se v této gramatice nevyskytuje. Při převodu do Greibachové 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.
$G_{GNF} = (\{S, A, B, C, N_a, N_b, N_c\},\ \{a, b, c\},\ P'',\ S)$
$S \to aBN_bN_a \mid bN_aA \mid bN_a \mid \varepsilon$
$A \to bCN_bBN_aN_bN_aAB \mid aBN_aN_bN_aAB \mid aN_bN_cN_aAB \mid bCN_bBN_aN_bN_aB \mid$
$\mid aBN_aN_bN_aB \mid aN_bN_cN_aB \mid bN_b$
$B \to bCN_bBN_aN_b \mid aBN_aN_b \mid aN_bN_c$
$C \to bCN_b \mid a$
$N_a \to a$
$N_b \to b$
$N_c \to c$

Příklad

Následující gramatiku převedeme do Greibachové 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 37. 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:
$G' = (\{S, A, A', B, B'\},\ \{a, b\},\ P',\ S)$
$S \to aA \mid AB \mid b$
$A \to aABaA' \mid bBaA' \mid aA' \mid aABa \mid bBa \mid a$

$A' \to BBaA' \mid BBa$
$B \to aB' \mid a$
$B' \to bB' \mid b$

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

Převeďte následující gramatiky do Greibachové normální formy.

$G_1 = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
$S \to aAB \mid AB \mid \varepsilon$
$A \to BbA \mid ab$
$B \to Ba \mid b$

$G_2 = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
$S \to BAaB \mid aBA \mid \varepsilon$
$A \to aaA \mid SB \mid a$
$B \to AbA \mid b$

$G_3 = (\{S, A, B, C\},\ \{a, b, c\},\ P,\ S)$
$S \to CAa \mid \varepsilon$
$A \to aCb \mid c$
$B \to cB \mid cS \mid b$
$C \to caC \mid A \mid b$

$G_4 = (\{E, F\},\ \{i, +, (,)\},\ P,\ S)$
$E \to E + F \mid F$
$F \to (E) \mid i$

2.3 Uzávěrové vlastnosti bezkontextových jazyků

Třída bezkontextových jazyků je uzavřena vzhledem k operacím sjednocení, zřetězení, iterace, pozitivní iterace, reverze a průniku s regulárním jazykem. Většinou se u bezkontextových jazyků tyto operace dají využít v souvislosti s gramatikami.

V následujícím příkladu jsou dány dvě gramatiky. Na nich si ukážeme použití operací sjednocení, zřetězení, iterace, pozitivní iterace a reverze jazyků.

Příklad

Jsou dány následující bezkontextové gramatiky:

$G_1 = (\{R, U, T\},\ \{a, b\},\ P_1,\ R)$
$R \to aUb \mid UT \mid \varepsilon$
$U \to aaU \mid TR \mid a$
$T \to bTb \mid \varepsilon$

$G_2 = (\{M, N, P\},\ \{a, b, c\},\ P_2,\ M)$
$M \to PM \mid bc$
$N \to aNa \mid \varepsilon$
$P \to MNb \mid c$

Ukázky derivací slov patřících do jazyků $L(G_1)$ a $L(G_2)$:

G_1: $R \Rightarrow UT \Rightarrow aT \Rightarrow abTb \Rightarrow abb$
G_2: $M \Rightarrow PM \Rightarrow MNbM \Rightarrow bcNbM \Rightarrow bcbM \Rightarrow bcbbc$

Provedeme operaci sjednocení, tedy vytvoříme gramatiku G_S generující jazyk $L(G_1) \cup L(G_2)$. Pravidla z obou gramatik přejmeme a použijeme nový neterminál S následovně:

$G_S = (\{S, R, U, T, M, N, P\},\ \{a, b, c\},\ P_S,\ S)$
$S \to R \mid M$
$R \to aUb \mid UT \mid \varepsilon$
$U \to aaU \mid TR \mid a$
$T \to bTb \mid \varepsilon$
$M \to PM \mid bc$
$N \to aNa \mid \varepsilon$
$P \to MNb \mid c$

V jazyce $L(G)$ by měla být všechna slova z jazyků $L(G_1)$ a $L(G_2)$. Například slova abb a $bcbbc$ by byla generována takto:

G_S: $S \Rightarrow R \Rightarrow UT \Rightarrow aT \Rightarrow abTb \Rightarrow abb$
G_S: $S \Rightarrow M \Rightarrow PM \Rightarrow MNbM \Rightarrow bcNbM \Rightarrow bcbM \Rightarrow bcbbc$

Hned v prvním kroku jsme rozhodli, jestli chceme generovat slovo patřící do prvního nebo druhého původního jazyka. Důležitým předpokladem je, aby množiny neterminálů původních gramatik byly disjunktní, což však platí i pro následující probírané operace.

Provedeme operaci zřetězení – vytvoříme gramatiku G_Z generující jazyk $L(G_1) \cdot L(G_2)$. Pravidla z obou gramatik přejmeme a použijeme nový neterminál Z následovně:

$G_Z = (\{Z, R, U, T, M, N, P\},\ \{a, b, c\},\ P_Z,\ Z)$
$Z \to RM$
$R \to aUb \mid UT \mid \varepsilon$
$U \to aaU \mid TR \mid a$
$T \to bTb \mid \varepsilon$
$M \to PM \mid bc$
$N \to aNa \mid \varepsilon$
$P \to MNb \mid c$

Když zřetězíme výše generovaná slova abb a $bcbbc$, získáme slovo $abbbcbbc$, které v gramatice G_Z vygenerujeme takto:

G_Z: $Z \Rightarrow RM \Rightarrow UTM \Rightarrow aTM \Rightarrow abTbM \Rightarrow abbM \Rightarrow$
$\Rightarrow abbPM \Rightarrow abbMNbM \Rightarrow abbbcNbM \Rightarrow abbbcbM \Rightarrow abbbcbbc$

Další operací je iterace. Vytvoříme gramatiku G_I generující jazyk $L(G_1)^*$. Pravidla z první gramatiky přejmeme a použijeme nový neterminál I následovně:

$G_I = (\{I, R, U, T\},\ \{a, b\},\ P_I,\ I)$

$I \to RI \mid \varepsilon$

$R \to aUb \mid UT \mid \varepsilon$

$U \to aaU \mid TR \mid a$

$T \to bTb \mid \varepsilon$

Začátek derivace v gramatice G_I by mohl vypadat například takto:

G_I: $\quad RI \Rightarrow RRI \Rightarrow RRRI \Rightarrow \ldots \Rightarrow RRR\cdots RI \Rightarrow RRR\cdots R$

Následně z výskytů neterminálu R generujeme řetězce, které jsou slovy z jazyka $L(G_1)$, v reálu tedy získáme slovo z jazyka $L(G_1)^*$ (prázdné slovo vygenerujeme v jednokrokové derivaci $I \Rightarrow \varepsilon$).

Pozitivní iterace je podobná. Vytvoříme gramatiku G_O generující jazyk $L(G_1)^+$. Pravidla z první gramatiky přejmeme a použijeme nový neterminál O následovně:

$G_O = (\{O, R, U, T\},\ \{a, b\},\ P_O,\ O)$

$O \to RO \mid R$

$R \to aUb \mid UT \mid \varepsilon$

$U \to aaU \mid TR \mid a$

$T \to bTb \mid \varepsilon$

S pravidly se zachází stejně jako u iterace, jen pro startovací symbol nemáme ε-pravidlo. Pokud v jazyce $L(G_1)$ není prázdné slovo, nebude ani v jazyce $L(G_O)$.

Zatímco u předchozích operací jsme vlastně jen předsunuli nová pravidla na začátku derivace a všechna původní pravidla jsme nechali v původním stavu, u reverze musíme zasáhnout do pravých stran pravidel (ale na druhou stranu nepotřebujeme žádný nový neterminál). Vytvoříme gramatiku G_R generující jazyk $L(G_1)^R$. Pravé strany všech pravidel obrátíme (převrátíme pořadí symbolů v řetězcích pravých stran).

$G_R = (\{R, U, T\},\ \{a, b\},\ P_R,\ R)$

$R \to bUa \mid TU \mid \varepsilon$

$U \to Uaa \mid RT \mid a$

$T \to bTb \mid \varepsilon$

Srovnejte následující derivace:

G_1: $\quad R \Rightarrow UT \Rightarrow aT \Rightarrow abTb \Rightarrow abb$

G_R: $\quad R \Rightarrow TU \Rightarrow Ta \Rightarrow bTba \Rightarrow bba$

Úkol

Jsou dány gramatiky G_1 a G_2:

$G_1 = (\{A, B\},\ \{a, b\},\ P_1,\ A)$

$A \to aaA \mid AB \mid \varepsilon$

$B \to bBa \mid A$

$G_2 = (\{C, D\},\ \{a, b\},\ P_2,\ C)$

$C \to aC \mid aD \mid aa$

$D \to bDC \mid \varepsilon$

Vytvořte gramatiky pro sjednocení, zřetězení, iteraci, pozitivní iteraci a reverzi jazyků $L(G_1)$ a $L(G_2)$.

2.4 Pumping lemma pro bezkontextové jazyky

S Pumping lemmatem jsme se seznámili už u regulárních jazyků. U bezkontextových jazyků narůstá složitost pravidel gramatiky a proto je složitější i tato věta.

Když pomocí Pumping lemma dokazujeme, že jazyk L není bezkontextový, postupujeme takto:

1. vybereme vhodné slovo $w \in L$, a to tak, aby bylo „dostatečně dlouhé“ – mělo by tedy být určeno výrazem, ve kterém je index („proměnná“) potenciálně jdoucí do nekonečna,
2. vybereme vhodné rozdělení $w = x \cdot y \cdot z \cdot u \cdot v$ tak, aby
   - $|y \cdot z \cdot u| \leq q$, kde q je některé konečné číslo, to znamená, že v celé prostřední části slova můžeme jako indexy použít pouze konečná čísla (popřípadě konstanty, u kterých počítáme, že jsou relativně malými čísly),
   - $|y \cdot u| > 0$ (v těchto dvou částech dohromady musí být alespoň jeden symbol),
3. zvolíme číslo $k \geq 0$ (většinou stačí vybrat $k = 0$ nebo $k = 2$) a vytvoříme slovo $w_k = x \cdot y^k \cdot z \cdot u^k \cdot v$
4. pokud $w_k \notin L$, vracíme se k bodu 2 a vyzkoušíme další rozdělení, pokud $w_k \in L$, vracíme se k bodu 1 a hledáme jiné vhodné slovo,
5. končíme tehdy, když se v bodu 2 nedaří najít další rozdělení.

Účelem je pro vybrané slovo otestovat všechna možná rozdělení a pro každé dojít k závěru, že $w_k \notin L$, pak dokážeme, že jazyk L není bezkontextový.

Příklad

Dokážeme, že jazyk $L = \{a^n b^{2n} a^n \ : \ n \geq 0\}$ není bezkontextový.

Vybereme si slovo $w = a^i b^{2i} a^i$ a předpokládáme, že i je dostatečně vysoké (může růst do nekonečna). Dále si určíme možná rozdělení tohoto slova – $w = x \cdot y \cdot z \cdot u \cdot v$:

1) $y \cdot z \cdot u$ obsahuje pouze symboly $'a'$ z první části slova,
2) $y \cdot z \cdot u$ v sobě zahrnuje hranici mezi symboly $'a'$ z první části slova a symboly $'b'$ z druhé části slova,
3) $y \cdot z \cdot u$ obsahuje pouze symboly $'b'$ z druhé části slova,
4) $y \cdot z \cdot u$ v sobě zahrnuje hranici mezi symboly $'b'$ z druhé části slova a symboly $'a'$ z třetí části slova,
5) $y \cdot z \cdot u$ obsahuje pouze symboly $'a'$ z třetí části slova.

Další rozdělení již nejsou možná, protože musí být splněny podmínky omezení $|y \cdot z \cdot u|$ shora a nenulové délky řetězce $y \cdot u$ (proto například do $y \cdot z \cdot u$ nemohou patřit symboly ze všech tří částí slova – tento řetězec by obsahoval celý podřetězec b^{2i}, který ale nelze shora omezit).

Prověříme všechny možnosti:

ad. 1) $x \cdot y \cdot z \cdot u \cdot v = (a^r) \cdot (a^s) \cdot \left(a^t\right) \cdot \left(a^h\right) \cdot \left(a^{i-(r+s+t+h)} b^{2i} a^i\right)$, kde $s + t + h \leq q$, $s + h > 0$

$w_n = x \cdot y^k \cdot z \cdot u^k \cdot v = a^r \cdot a^{k \cdot s} \cdot a^t \cdot a^{k \cdot h} a^{i-(r+s+t+h)} b^{2i} a^i = a^{k \cdot s + k \cdot h + i - s - h} b^{2i} a^i$

pro $k = 0$: $w_0 = a^{i-s-h} b^{2i} a^i \notin L$ (protože $s + h > 0$)

ad. 2) $(x) \cdot (y \cdot z \cdot u) \cdot (v) = (a^{m_1}) \cdot (a^{m_2} b^{m_3}) \cdot \left(b^{m_4} a^i\right)$

sestrojíme $x \cdot y^k \cdot z \cdot u^k \cdot v$; v prostředních třech částech sice mohou být symboly $'a'$ a $'b'$ uspořádány více způsoby, ale vždy se při pumpování změní buď počet $'a'$ v první části slova nebo počet $'b'$ v druhé části slova nebo obojí, tedy výsledek nebude souhlasit s třetí částí slova.

ad. 3) $x \cdot y \cdot z \cdot u \cdot v = \left(a^i b^r\right) \cdot \left(b^s\right) \cdot \left(b^t\right) \cdot \left(b^h\right) \cdot \left(b^{2i-(r+s+t+h)} a^i\right)$, kde $s + t + h \leq q$, $s + h > 0$

$w_n = x \cdot y^k \cdot z \cdot u^k \cdot v = a^i b^r b^{k \cdot s} b^t b^{k \cdot h} b^{2i-(r+s+t+h)} a^i = a^i b^{k \cdot s + k \cdot h + 2i - s - h} a^i$

pro $k = 0$: $w_0 = a^i b^{2i-s-h} a^i \notin L$

ad. 4) dokazujeme podobně jako bod 2)

ad. 5) dokazujeme podobně jako bod 1)

Pro všechna možná rozdělení nám vyšlo, že pro alespoň jednu hodnotu k slovo w_k nepatří do jazyka L, proto jazyk L není bezkontextový.

Všechny možnosti rozdělení si můžeme přehledněji zapsat do tabulky:

ad.	x	y	z	u	v	podmínky	w_k, w_0
1)	a^r	a^s	a^t	a^h	$a^{i-r-s-t-h} b^{2i} a^i$	$s + h > 0$ $s + t + h \leq q$	$a^{k \cdot s + k \cdot h + i - s - h} b^{2i} a^i$ $a^{i-s-h} b^{2i} a^i \notin L$
2)	$a^{i-r-s-t}$	a^r	a^s	$a^t b^h$	$b^{2i-h} a^i$	$r + t + h > 0$ $r + s + t + h \leq q$	$a^{i-r-t+kr} (a^t b^h)^k b^{2i-h} a^i$ $a^{i-r-t} b^{2i-h} a^i \notin L$
	a^{i-r-s}	a^r	$a^s b^t$	b^h	$b^{2i-t-h} a^i$	$r + h > 0$ $r + s + t + h \leq q$	$a^{i-r+k \cdot r} b^{k \cdot h + 2i - h} a^i$ $a^{i-r} b^{2i-h} a^i \notin L$
	a^{i-r}	$a^r b^s$	b^t	b^h	$b^{2i-s-t-h} a^i$	$r + s + h > 0$ $r + s + t + h \leq q$	$a^{i-r} (a^r b^s)^k b^{k \cdot h + 2i - s - h} a^i$ $a^{i-r} b^{2i-s-h} a^i \notin L$
3)	$a^i b^r$	b^s	b^t	b^h	$b^{2i-r-s-t-h} a^i$	$s + h > 0$ $s + t + h \leq q$	$a^i b^{k \cdot s + k \cdot h + 2i - s - h} a^i$ $a^i b^{2i-s-h} a^i \notin L$
4)	Dokazuje se podobně jako bod 2)						
5)	Dokazuje se podobně jako bod 1)						

U obdobného příkladu pro Pumping lemma pro regulární jazyky jsme si obvykle vystačili s jedním (patřičně zobecněným) řádkem. Zde jich je víc, protože je taky víc možností, jak slovo rozdělit. Ovšem princip je stejný – při určení slova používáme „dostatečně velký“ index (zde například i, který považujeme za konstantu, a obvykle počítáme s tím, že platí $i > q$ (tudíž se tento index nemůže použít v částech y, z, u, protože $|x \cdot y \cdot u| \leq q$). Počty jednotlivých symbolů v částech pak označujeme jinými indexy, které jsou pro nás taky konstantami, ale tentokrát pro ně nemáme žádné spodní omezení (takže potenciálně „malá“ čísla). Naopak k pro pumpování je pro nás proměnná.

Příklad

Dokážeme, že jazyk $L = \left\{ a^{n^2} \ : \ n \geq 0 \right\}$ není bezkontextový.

Opět zvolíme nějaké slovo $w = a^{p^2} \in L$ s „dostatečně velkým“ indexem p a rozdělíme je následovně:
$a^{p^2} = a^{x_1} a^{x_2} a^{x_3} a^{x_4} a^{x_5}$

Pokud z těchto dvou reprezentací téhož slova (na dvou stranách rovnice) vezmeme pouze délky, získáme již „číselnou“ rovnici
$p^2 = x_1 + x_2 + x_3 + x_4 + x_5$.

Podle podmínek Pumping lemmatu musí zároveň platit nerovnice

- $x_2 + x_4 > 0$ podle podmínky $|y \cdot u| > 0$
- $x_2 + x_3 + x_4 \leq q$ podle podmínky $|y \cdot z \cdot u| \leq q$

Po pumpování získáme $w_k = a^{x_1} a^{k \cdot x_2} a^{x_3} a^{k \cdot x_4} a^{x_5}$.

Dále postupujeme důkazem sporem. Předpokládejme, že jazyk L je bezkontextový. Pak by slovo w_k muselo patřit do jazyka L, protože to předpokládají podmínky v Pumping lemmatu, tedy muselo by platit:
$x_1 + k \cdot x_2 + x_3 + k \cdot x_4 + x_5 \ = \ m^2$, kde m je některé nezáporné číslo, jehož hodnota je závislá na hodnotě k (čím vyšší k, tedy čím víc pumpujeme, tím vyšší bude také m). Je třeba si uvědomit, jak je to s konstantami a proměnnými: čísla $p, x_1, \ldots, x_5$ jsou konstantami, protože jsme volili jedno slovo a pevně jsme stanovili jeho rozdělení. Oproti tomu k, m jsou proměnné, protože za k dosazujeme různá celá nezáporná čísla, a m by měla být proměnná závislá na k. Pro přehlednost naši rovnici upravíme:
$k \cdot (x_2 + x_4) + (x_1 + x_3 + x_5) \ = \ m^2$

Získali jsme rovnici, v jejíchž obou částech oddělených rovnítkem jsou funkce. Zatímco pravá část rovnice roste s mocninnou řadou, pravá lineárně (je to lineární funkce), a tedy mnohem pomaleji. Uvědomme si, že na obou stranách rovnice musí být výsledkem přirozené číslo (oborem hodnot obou funkcí jsou přirozená čísla), a pro jakoukoliv hodnotu k bychom měli najít odpovídající hodnotu m. Jenže to není možné – neexistuje dvojice lineární a mocninné funkce, jejichž grafy by se překrývaly. Dospěli jsme ke sporu, proto jazyk L není bezkontextový.

Příklad

Dokážeme, že jazyk $L = \{ww \ : \ w \in \{a, b\}^*\}$ není bezkontextový. Tento jazyk se skládá ze slov, jejichž obě poloviny jsou stejné.

Pro důkaz si vybereme slovo $w = a^i b^i a^i b^i$, přičemž i je dostatečně velké číslo (počítejme $i > q$), a dokážeme, že pro toto slovo neexistuje žádné rozdělení, které by umožňovalo „pumpování“ dle Pumping lemmatu.

Zamysleme se nad tím, kam při rozdělení na části $x \cdot y \cdot z \cdot u \cdot v$ může „padnout“ trojice $y \cdot z \cdot u$. Víme, že její délka je shora omezena hranicí q, proto sem nebudeme umísťovat index i. Jsou tyto možnosti:

1) zahrnuje pouze symboly $'a'$ z první poloviny slova,
2) je na rozmezí mezi symboly $'a'$ a $'b'$ v první polovině slova,
3) zahrnuje pouze symboly $'b'$ z první poloviny slova,
4) je na rozmezí mezi polovinami slova, zahrnuje tedy symboly $'b'$ a $'a'$ na tomto rozmezí,
5) nachází se v druhé polovině slova – bez újmy na obecnosti můžeme tuto část důkazu vynechat, byla by obdobná jako v předchozích bodech.

Podíváme se na jednotlivé možnosti.

ad. 1) Pokud $y \cdot z \cdot u$ zabírá pouze symboly $'a'$ z první poloviny slova (samozřejmě ne všechny), pak by při pumpování došlo k nesouladu mezi první a druhou polovinou slova (první polovina by se měnila, druhá nikoliv), proto $w_k \ \notin L$.

ad. 2) Tentýž případ, měnil by se počet a struktura znaků pouze v první polovině slova. Poloviny slova by byly shodné pouze pro $k = 1$, ale Pumping lemma vyžaduje, aby byly shodné $\forall k \geq 0$.

ad. 3) Stejný případ.

ad. 4) V první polovině slova by rostl počet symbolů $'b'$, v druhé polovině slova zase počet symbolů $'a'$. Opět by se nerovnaly a tedy $w_k \ \notin L$.

Úkoly

1. Dokažte pomocí Pumping lemma, že jazyk $L_1 = \{a^n b^n c^n \ : \ n \geq 1\}$ není bezkontextový.
2. Dokažte pomocí Pumping lemma, že jazyk $L_2 = \left\{a^{2^n} \ : \ n \geq 0\right\}$ není bezkontextový.
3. Dokažte pomocí Pumping lemma, že jazyk $L_3 = \{wcw \ : \ w \in \{a, b\}^*\}$ není bezkontextový.
4. Dokažte, že jazyk $L_4 = \{a^n b^m a^n \ : \ 0 \leq m \leq n\}$ není bezkontextový. Použijte podobný postup jako je v příkladu na straně 48, a také nerovnosti v zadání jazyka.
5. Ukažte, že pro jazyk $L_2 = \left\{a^n b^{2n} a^n b^m \ : \ m, n \geq 0\right\}$ podmínka z Pumping lemmatu platí, třebaže nejde o bezkontextový jazyk. Proč?

Kapitola 3

Zásobníkový automat

Se zásobníkovými automaty jsme se seznámili v minulém semestru, v tomto semestru se podíváme na některé jejich typické vlastnosti a zaměříme se na způsoby jeho vytváření.

3.1 Připomenutí – jak pracuje zásobníkový automat

Zásobníkový automat pracuje takto:

1. vyjme symbol na vrcholu zásobníku,
2. může/nemusí přečíst jeden symbol ze vstupní pásky, pokud přečte, posune se o políčko dál,
3. dále se rozhoduje podle
   - svého vnitřního stavu,
   - symbolu, který vyndal ze zásobníku,
   - pokud četl ze vstupní pásky, pak i podle přečteného symbolu,
4. akce automatu spočívá v
   - přechodu do některého dalšího stavu
   - a v uložení řetězce znaků do zásobníku.

Rozeznáváme tyto základní typy zásobníkových automatů:

1. zásobníkový automat končící přechodem do koncového stavu,
2. zásobníkový automat končící prázdným zásobníkem.

Existuje také jejich kombinace – zásobníkový automat končící přechodem do koncového stavu při prázdném zásobníku.

Zásobníkový automat končící přechodem do koncového stavu značíme $\mathcal{A}_F$. Jeho koncová konfigurace má tento tvar:

$$(q_f, \varepsilon, \gamma),\ q_f \in F,\ \gamma \in \Gamma^*$$

Rozpoznávaný jazyk je

$$L(\mathcal{A}_F) = \{w \in \Sigma^* \ :\ (q_0, w, Z_0) \vdash {}^*(q_f, \varepsilon, \gamma),\ q_f \in F,\ \gamma \in \Gamma^*\}$$

To znamená, že abychom mohli skončit, musíme přečíst celý vstup a dostat se do některého koncového stavu. Množina koncových stavů zde nebývá prázdná (kdyby byla, automat by rozpoznával prázdný jazyk).

Zásobníkový automat končící s prázdným zásobníkem značíme $\mathcal{A}_\emptyset$. Koncová konfigurace vypadá následovně:

$$(q, \varepsilon, \varepsilon),\ q \in Q$$

Rozpoznávaný jazyk je

$$L(\mathcal{A}_\emptyset) = \{w \in \Sigma^* \ :\ (q_0, w, Z_0) \vdash {}^*(q, \varepsilon, \varepsilon),\ q \in Q\}$$

Abychom mohli skončit, musíme přečíst celý vstup a vyprázdnit zásobník. Koncové stavy nepotřebujeme, proto je množina koncových stavů obvykle prázdná.

Zásobníkový automat končící v koncovém stavu při prázdném zásobníku značíme $\mathcal{A}_{F,\emptyset}$. Jeho koncová konfigurace je

$$(q_f, \varepsilon, \varepsilon),\ q_f \in F$$

Rozpoznávaný jazyk je

$$L(\mathcal{A}_{F,\emptyset}) = \{w \in \Sigma^* \ :\ (q_0, w, Z_0) \vdash {}^*(q_f, \varepsilon, \varepsilon),\ q_f \in F\}$$

Je třeba splnit podmínky obou předchozích typů zároveň – abychom mohli skončit výpočet (úspěšně), musí být přečten celý vstup, vyprázdněn zásobník a navíc musí být automat v koncovém stavu.

Poznámka:

Všechny tři typy zásobníkových automatů končí samozřejmě výpočet i tehdy, když nejsou v žádné koncové konfiguraci, ale do žádné další se nemohou dostat (přechodová funkce nedává možnost reagovat v daném stavu s daným obsahem zásobníku a vstupní pásky). V tomto případě však končíme s tím, že zpracovávané slovo nepatří do jazyka rozpoznávaného automatem.

Příklad

Sestrojíme zásobníkový automat (dále ZA) končící s prázdným zásobníkem rozpoznávající jazyk $L = \{wcw^R \ :\ w \in \{a, b\}^*\}$

Automat bude pracovat takto:

- v první fázi (stav q_0) čte obsah vstupu (první polovina slova) a ukládá do zásobníku (co v každém kroku vyjmeme, vrátí do zásobníku zároveň se symbolem ze vstupu, tedy ukládá dva symboly),
- díky principu zásobníku (čteme v opačném pořadí, než jak byly symboly uloženy) je ukládaná první polovina slova zároveň zrcadlově převrácena,
- když na vstupu narazíme na c (hranice, polovina slova), přejdeme do stavu q_1 a tím změníme způsob práce automatu,
- když jsme ve stavu q_1, nic do zásobníku neukládáme, symbol, který v každém kroku vyjmeme, porovnáme s tím, co je na vstupu – když souhlasí, můžeme pokračovat (tj. v každém kroku se posuneme na vstupu a zároveň ubereme symbol ze zásobníku).

V plné specifikaci uvedeme stavy, abecedu, zásobníkovou abecedu, počáteční stav, symbol konce zásobníku, δ-funkci a množinu koncových stavů (zde prázdnou), dále upřesníme předpis δ-funkce:
$\mathcal{A}_\emptyset = (\{q_0, q_1\},\ \{a, b, c\},\ \{a, b, Z_0\},\ q_0,\ Z_0,\ \delta,\ \emptyset)$

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$	na začátku výpočtu, slovo začíná a
$\delta(q_0, b, Z_0) = (q_0, bZ_0)$	na začátku výpočtu, slovo začíná b
$\delta(q_0, a, X) = (q_0, aX),\ X \in \{a, b\}$	zatím jen načítáme a ukládáme do zásobníku
$\delta(q_0, b, X) = (q_0, bX),\ X \in \{a, b\}$	
$\delta(q_0, c, X) = (q_1, X),\ X \in \{a, b\}$	jsme na hranici
$\delta(q_1, a, a) = (q_1, \varepsilon)$	shoda a v obou polovinách slova
$\delta(q_1, b, b) = (q_1, \varepsilon)$	shoda b v obou polovinách slova
$\delta(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$	skončili jsme na vstupu i v zásobníku
$\delta(q_0, \varepsilon, Z_0) = (q_1, \varepsilon)$	potřebujeme přijmout i prázdné slovo

Ukázka výpočtu automatu na slovo $abcba$:

$(q_0, abcba, Z_0) \vdash (q_0, bcba, aZ_0) \vdash$	q_0 : přenášíme do zásobníku obsah vstupu
$\vdash (q_0, cba, baZ_0) \vdash$	hraniční bod, přejdeme do módu q_1
$\vdash (q_1, ba, baZ_0) \vdash (q_1, a, aZ_0) \vdash$	q_1 : jen vybíráme ze zásobníku a porovnáváme
$\vdash (q_1, \varepsilon, Z_0) \vdash (q_1, \varepsilon, \varepsilon)$	konec

Prázdné slovo na vstupu zpracujeme takto:
$(q_0, \varepsilon, Z_0) \vdash (q_1, \varepsilon, \varepsilon)$

Slova nepatřící do jazyka:
$(q_0, abca, Z_0) \vdash (q_0, bca, aZ_0) \vdash (q_0, ca, baZ_0) \vdash (q_1, a, baZ_0) \vdash$ chyba
$(q_0, acaaa, Z_0) \vdash (q_0, caaa, aZ_0) \vdash (q_1, aaa, aZ_0) \vdash (q_1, aa, Z_0) \vdash (q_1, aa, \varepsilon) \vdash$ chyba
$(q_0, bc, Z_0) \vdash (q_0, c, bZ_0) \vdash (q_1, \varepsilon, bZ_0) \vdash$ chyba

Příklad

Zásobníkový automat končící v koncovém stavu (a vlastně i „hybridní“ typ) bychom z automatu z předchozího příkladu vytvořili takto:

- přidáme nový počáteční stav, nový koncový stav a nový zásobníkový symbol,
- první krok výpočtu bude spočívat v tom, že se napojíme na původní automat, jehož výpočet pak budeme „simulovat“ (tj. jeho přechodovou funkci také využijeme),
- ovšem pod jeho symbolem konce zásobníku budeme mít vsunutý svůj, který nám pomůže detekovat konec výpočtu simulovaného automatu, po čemž přejdeme do koncového stavu.

Původní automat je tedy $\mathcal{A}_\emptyset = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, \emptyset)$, kde $Q = \{q_0, q_1\},\ \Sigma = \{a, b, c\},\ \Gamma = \{a, b, Z_0\}$.
Sestrojíme nový automat $\mathcal{A}_F = \left(Q \cup \{q_0^F, q_f^F\}, \Sigma, \Gamma \cup \{Z_0^F\}, q_0^F, Z_0^F, \delta^F, \{q_f^F\}\right)$ takto:

- celou původní přechodovou funkci přejmeme: $\delta^F(q, x, Z) = \delta(q, x, Z),\ q \in Q,\ x \in \Sigma,\ Z \in \Gamma$,
- pro první krok bude $\delta^F(q_0^F, \varepsilon, Z_0^F) = (q_0, Z_0 Z_0^F)$, takže první dvě konfigurace ve výpočtu jsou $(q_0^F, w, Z_0^F) \vdash (q_0, w, Z_0 Z_0^F) \vdash \ldots$
  od druhého kroku se napojujeme na výpočet původního automatu a používáme předpisy přejaté přechodové funkce,

- když výpočet simulovaného automatu končí (tj. jeho zásobník by byl prázdný), v našem automatu bude v zásobníku „pomocná zarážka“ Z_0^F, která byla od samého začátku podsunuta pod symbolem Z_0, podle toho poznáme, že máme udělat poslední krok pomocí předpisu
  $\delta^F(q, \varepsilon, Z_0^F) = (q_f^F, \varepsilon)$ (pro všechny stavy původního automatu $q \in Q$),
  čímž přejdeme do koncového stavu a nádavkem opravdu vyprázdníme zásobník.

V našem příkladu:

$\mathcal{A}_F = \left(Q \cup \{q_0^F, q_f^F\}, \Sigma, \Gamma \cup \{Z_0^F\}, q_0^F, Z_0^F, \delta^F, \{q_f^F\}\right)$

$\delta^F(q_0, X, Z_0) = (q_0, aZ_0),\ X \in \{a, b\}$
$\delta^F(q_0, a, X) = (q_0, aX),\ X \in \{a, b\}$
$\delta^F(q_0, b, X) = (q_0, bX),\ X \in \{a, b\}$
$\delta^F(q_0, c, X) = (q_1, X),\ X \in \{a, b\}$
$\delta^F(q_1, a, a) = (q_1, \varepsilon)$
$\delta^F(q_1, b, b) = (q_1, \varepsilon)$
$\delta^F(q_1, \varepsilon, Z_0) = (q_1, \varepsilon)$
$\delta^F(q_0, \varepsilon, Z_0) = (q_1, \varepsilon)$

$\delta^F(q_0^F, \varepsilon, Z_0^F) = (q_0, Z_0 Z_0^F)$
$\delta^F(q_0, \varepsilon, Z_0^F) = (q_f^F, \varepsilon)$
$\delta^F(q_1, \varepsilon, Z_0^F) = (q_f^F, \varepsilon)$

Srovnejte výpočet téhož slova v původním a novém automatu:

$\mathcal{A}_\emptyset$: $(q_0, aca, Z_0) \vdash (q_0, ca, aZ_0) \vdash (q_1, a, aZ_0) \vdash (q_1, \varepsilon, Z_0) \vdash (q_1, \varepsilon, \varepsilon)$

$\mathcal{A}_F$: $(q_0^F, aca, Z_0^F) \vdash (q_0, aca, Z_0 Z_0^F) \vdash (q_0, ca, aZ_0 Z_0^F) \vdash (q_1, a, aZ_0 Z_0^F) \vdash (q_1, \varepsilon, Z_0 Z_0^F) \vdash$
$\vdash (q_1, \varepsilon, Z_0^F) \vdash (q_f, \varepsilon, \varepsilon)$

Poznámka:

Zásobníková abeceda se nemusí nijak krýt se vstupní abecedou automatu. Zatím jsme do zásobníku vkládali tentýž symbol, který jsme přečetli na vstupu, ale ve skutečnosti tyto abecedy mohou být navzájem nezávislé, jak uvidíme v následujícím příkladu.

Příklad

Sestrojíme zásobníkový automat končící v koncovém stavu rozpoznávající jazyk

$L = \{a^n b^{2n} \ : \ n \geq 1\}$

Nejdřív si promyslíme, jak budeme v jednotlivých stavech zacházet se zásobníkem:

- ve stavu q_0 budeme načítat vstup a za každý symbol a ze stupu uložíme dva symboly x na zásobník,
- pokud ve stavu q_0 najdeme na vstupu symbol b, přejdeme do stavu q_1 a ze zásobníku vyndáme jeden symbol x,
- ve stavu q_1 budeme načítat vstup a srovnávat se zásobníkem (za každý symbol b ze vstupu vyndáme jeden symbol x ze zásobníku,
- pokud ve stavu q_1 narazíme na dno zásobníku, přejdeme do koncového stavu q_f.

Především musíme zajistit, aby nám *seděly počty*. Ke každému symbolu a ze vstupu patří dva symboly x na zásobníku (o to se staráme ve stavu q_0), a ke každému symbolu b na vstupu přísluší jeden symbol

x ze zásobníku. Díky tomu ke každému symbolu a z první části vstupu přiřazujeme dva symboly b z druhé části vstupu.

Náš zásobníkový automat je definován takto:

$\mathcal{A}_{\mathcal{F}} = (\{q_0, q_1, q_f\},\ \{a, b\},\ \{x, Z_0\},\ \delta,\ q_0,\ Z_0,\ \{q_f\})$

$\delta(q_0, a, Z_0) = (q_0, xxZ_0)$	první symbol ze vstupu; vyndaný symbol Z_0 vrátíme do zásobníku
$\delta(q_0, a, x) = (q_0, xxx)$	za každé a na vstupu přidáme xx do zásobníku, plus vrátíme vyndané x
$\delta(q_0, b, x) = (q_1, \varepsilon)$	přecházíme do druhé části slova
$\delta(q_1, b, x) = (q_1, \varepsilon)$	za každé b na vstupu vyndáme jedno x ze zásobníku
$\delta(q_1, \varepsilon, Z_0) = (q_f, \varepsilon, Z_0)$	konec, přechod do koncového stavu

Ukážeme si zpracování některého slova jazyka:

$(q_0, aabbbb, Z_0) \vdash (q_0, abbbb, xxZ_0) \vdash (q_0, bbbb, xxxxZ_0) \vdash (q_1, bbb, xxxZ_0) \vdash (q_1, bb, xxZ_0) \vdash$
$\vdash (q_1, b, xZ_0) \vdash (q_1, \varepsilon, Z_0) \vdash (q_f, \varepsilon, Z_0)$

Slova nepatřící do jazyka:

$(q_0, ab, Z_0) \vdash (q_0, b, xxZ_0) \vdash (q_1, \varepsilon, xZ_0) \vdash$ chyba

$(q_0, \varepsilon, Z_0) \vdash$ chyba

(pokud není nic na vstupu, dá se použít jen ε-přechod, jenže ten pro stav q_0 nemáme)

Příklad

Kdybychom trochu upravili jazyk z předchozího příkladu (zařadili do něj prázdné slovo):

$L = \{a^n b^{2n} \ :\ n \geq 0\}$

Zásobníkový automat by byl skoro stejný, jen bychom museli navíc přidat jeden předpis do přechodové funkce: $\delta(q_0, \varepsilon, Z_0) = (q_f, Z_0)$

Pak by zpracování prázdného slova bylo možné:

$(q_0, \varepsilon, Z_0) \vdash (q_f, \varepsilon, Z_0)$

Poznámka:

Na „krajní podmínky“ si dávejte pozor. Vždy, když navrhnete automat (jakýkoliv), vyzkoušejte, jak reaguje zejména na nejkratší slova, která by měl přijímat, nebo naopak na taková krátká slova, která by přijímat neměl.

Příklad

V příkladu na straně 55 jsme sestrojili zásobníkový automat rozpoznávající koncovým stavem pro jazyk $L = \{a^n b^{2n} \ :\ n \geq 1\}$.

Tento automat převedeme na zásobníkový automat rozpoznávající prázdným zásobníkem (plus koncovým stavem).

- přidáme nový počáteční stav, nový „ukončující“ stav (ten bude zároveň koncovým stavem) a nový zásobníkový symbol,

- první krok výpočtu bude spočívat v tom, že se napojíme na původní automat, jehož výpočet pak budeme „simulovat“,
- pod jeho symbolem konce zásobníku budeme mít vsunutý svůj, který nám pomůže detekovat konec výpočtu simulovaného automatu, po čemž vymažeme obsah zásobníku a ukončíme výpočet.

Původní automat je tedy $\mathcal{A}_F = (Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$, kde $Q = \{q_0, q_1, q_f\},\ \Sigma = \{a, b\},\ \Gamma = \{x, Z_0\}$.
Sestrojíme nový automat $\mathcal{A}_\emptyset = \left(Q \cup \{q_0^\emptyset, d^\emptyset\}, \Sigma, \Gamma \cup \{Z_0^\emptyset\}, q_0^\emptyset, Z_0^\emptyset, \delta^\emptyset, \{d^\emptyset\}\right)$ takto:

- celou původní přechodovou funkci přejmeme: $\delta^\emptyset(q, x, Z) = \delta(q, m, Z),\ q \in Q,\ m \in \Sigma,\ Z \in \Gamma$,
- pro první krok bude $\delta^\emptyset(q_0^\emptyset, \varepsilon, Z_0^\emptyset) = (q_0, Z_0 Z_0^\emptyset)$, takže první dvě konfigurace ve výpočtu jsou $(q_0^\emptyset, w, Z_0^\emptyset) \vdash (q_0, w, Z_0 Z_0^\emptyset) \vdash \ldots$
  od druhého kroku se napojujeme na výpočet původního automatu a používáme předpisy přejaté přechodové funkce,
- když výpočet simulovaného automatu končí (tj. jsme ve stavu q_f), přejdeme do „mazacího“ stavu $d^\emptyset$ a postupně vyndáme všechny zbývající symboly ze zásobníku, čímž umožníme ukončení výpočtu:
  $\delta^\emptyset(q_f, \varepsilon, Z) = (d^\emptyset, \varepsilon)$ (pro všechny stavy původního automatu $q_f \in F$ a $Z \in \Gamma$),
  $\delta^\emptyset(d, \varepsilon, Z) = (d^\emptyset, \varepsilon)$ (pro všechny symboly $Z \in \Gamma \cup \{Z_0^\emptyset\}$).

V našem příkladu:

$\mathcal{A}_\emptyset = \left(Q \cup \{q_0^\emptyset, d^\emptyset\}, \Sigma, \Gamma \cup \{Z_0^\emptyset\}, q_0^\emptyset, Z_0^\emptyset, \delta^\emptyset, \{d^\emptyset\}\right)$

$\delta^\emptyset(q_0, a, Z_0) = (q_0, xxZ_0)$
$\delta^\emptyset(q_0, a, x) = (q_0, xxx)$
$\delta^\emptyset(q_0, b, x) = (q_1, \varepsilon)$
$\delta^\emptyset(q_1, b, x) = (q_1, \varepsilon)$
$\delta^\emptyset(q_1, \varepsilon, Z_0) = (q_f, \varepsilon, Z_0)$
$\delta^\emptyset(q_0^\emptyset, \varepsilon, Z_0^\emptyset) = (q_0, Z_0 Z_0^\emptyset)$
$\delta^\emptyset(q_f, \varepsilon, x) = (d^\emptyset, \varepsilon)$
$\delta^\emptyset(q_f, \varepsilon, Z_0) = (d^\emptyset, \varepsilon)$
$\delta^\emptyset(d^\emptyset, \varepsilon, x) = (d^\emptyset, \varepsilon)$
$\delta^\emptyset(d^\emptyset, \varepsilon, Z_0) = (d^\emptyset, \varepsilon)$

Srovnejte výpočet téhož slova v původním a novém automatu:

$\mathcal{A}_F$: $(q_0, abb, Z_0) \vdash (q_0, bb, xxZ_0) \vdash (q_1, b, xZ_0) \vdash (q_1, \varepsilon, Z_0) \vdash (q_f, \varepsilon, Z_0)$

$\mathcal{A}_\emptyset$: $(q_0^\emptyset, abb, Z_0^\emptyset) \vdash (q_0, abb, Z_0 Z_0^\emptyset) \vdash (q_0, bb, xxZ_0 Z_0^\emptyset) \vdash (q_1, b, xZ_0 Z_0^\emptyset) \vdash (q_1, \varepsilon, Z_0 Z_0^\emptyset) \vdash$
$\vdash (q_f, \varepsilon, Z_0 Z_0^\emptyset) \vdash (d^\emptyset, \varepsilon, Z_0^\emptyset) \vdash (d^\emptyset, \varepsilon, \varepsilon)$

Úkoly

1. Zamyslete se nad tím, jak by vypadal zásobníkový automat pro jazyk $L = \{a^n b^k c^n \ :\ n, k \geq 1\}$. Prostřední část bychom zpracovávali prakticky podobně jako u konečného automatu – bez zásobníku.
2. Sestrojte zásobníkový automat končící s prázdným zásobníkem pro jazyk $L = \{a^n b^n cc \ :\ n \geq 0\}$. Pak podle postupu ukázaného na příkladu proveďte změnu na zásobníkový automat končící v koncovém stavu.
3. Podle postupů v předchozích příkladech sestrojte zásobníkový automat končící v koncovém stavu pro jazyk $L = \{ww^R \ :\ w \in \{a, b\}^*\}$. Pak podle postupu ukázaného na příkladu proveďte změnu na zásobníkový automat končící s prázdným zásobníkem.

3.2 Zásobníkový automat podle bezkontextové gramatiky

Mezi zásobníkovými automaty a bezkontextovými gramatikami existuje podobný vztah jako mezi konečnými automaty a regulárními gramatikami. Existuje algoritmus, jak podle bezkontextové gramatiky sestrojit ekvivalentní zásobníkový automat (ten si za chvíli ukážeme) a naopak jak podle zásobníkového automatu sestrojit bezkontextovou gramatiku (ten je složitější, necháme si ho na další semestr).

Postup bude založen na úplně jiném principu než u regulárních jazyků. Cokoliv se bude dít, to se bude dít na zásobníku. Budeme potřebovat jen jediný stav (resp. stav pro nás nebude představovat žádnou relevantní informaci, ten jeden máme jen proto, že nějaký stav být musí). Podle zadané gramatiky sestrojíme zásobníkový automat končící prázdným zásobníkem.

Postupujeme takto:

- jediný stav q, je zároveň počáteční,
- abecedu automatu vezmeme z množiny terminálních symbolů,
- zásobníkovou abecedu utvoříme z množin terminálních a neterminálních symbolů (vše se děje na zásobníku, tj. musíme sem zařadit všechny „stavební kameny“),
- δ-funkci sestrojíme především podle pravidel, jak uvidíme o něco dále,
- jako symbol konce zásobníku použijeme startovací symbol gramatiky,
- množina koncových stavů bude prázdná (končíme prázdným zásobníkem).

Zbývá určit přechodovou funkci. Její předpis se bude skládat ze dvou částí:

1. první část sestrojíme podle pravidel gramatiky, použijeme tehdy, když je na vrcholu zásobníku neterminál:
   $A \to \alpha \qquad \Longrightarrow \qquad \delta(q, \varepsilon, A) \ni (q, \alpha)$
   znamená:
   pokud je na vrcholu zásobníku neterminál A, nebudeme si všímat vstupu, vyjmeme A ze zásobníku a místo něj tam dáme α (pravou stranu pravidla)
2. druhá část se použije tehdy, když bude na vrcholu zásobníku terminální symbol:
   $\delta(q, a, a) = (q, \varepsilon)$
   znamená:
   pokud je na vrcholu zásobníku terminál a a na vstupu bude tentýž terminál, vyjmeme a ze zásobníku a nic tam už dávat nebudeme.

Příklad

Podle zadané gramatiky sestrojíme zásobníkový automat:
$G = (\{S, A, B\},\ \{a, b, c\},\ P,\ S)$
$S \to abSba \mid A$
$A \to cAc \mid aB$
$B \to aB \mid \varepsilon$

Budeme chtít automat končící s prázdným zásobníkem, ten je v těchto případech vhodnější. Použijeme jediný stav q, zásobníková abeceda bude obsahovat všechny terminály a neterminály. Výsledný automat bude následující:
$\mathcal{A} = (\{q\},\ \{a, b, c\},\ \{S, A, B, a, b, c\},\ \delta,\ q,\ S,\ \emptyset)$

První část – podle pravidel:
$\delta(q, \varepsilon, S) = \{(q, abSba),\ (q, A)\}$
$\delta(q, \varepsilon, A) = \{(q, cAc),\ (q, aB)\}$
$\delta(q, \varepsilon, B) = \{(q, aB),\ (q, \varepsilon)\}$

Druhá část – podle terminálů:
$\delta(q, a, a) = \{(q, \varepsilon)\}$
$\delta(q, b, b) = \{(q, \varepsilon)\}$
$\delta(q, c, c) = \{(q, \varepsilon)\}$

V gramatice odvodíme slovo a totéž slovo rozpoznáme ve vytvořeném automatu:
$S \Rightarrow abSba \Rightarrow abAba \Rightarrow abaBba \Rightarrow abaaBba \Rightarrow abaaba$

Ekvivalentní zpracování (téhož) slova v automatu:
$(q, abaaba, S) \vdash (q, abaaba, abSba) \vdash (q, baaba, bSba) \vdash (q, aaba, Sba) \vdash (q, aaba, Aba) \vdash$
$\vdash (q, aaba, aBba) \vdash (q, aba, Bba) \vdash (q, aba, aBba) \vdash (q, ba, Bba) \vdash (q, ba, ba) \vdash (q, a, a) \vdash (q, \varepsilon, \varepsilon)$

Poznámka:

Co se vlastně děje v takto zkonstruovaném automatu? Všimněte si obsahu zásobníku, srovnejte ho s odvozením slova v gramatice. Když si odmyslíme manipulaci s terminály v zásobníku (zleva je „umazáváme“), je jasné, že vlastně provádíme simulaci. V zásobníku simulujeme odvození slova, a pokud se podaří v simulaci dojít v zásobníku ke slovu, které jsme měli na vstupu (přesněji celé ho v zásobníku po odvození „zlikvidovat“), můžeme tvrdit, že slovo ze vstupu lze vygenerovat gramatikou, podle které byl automat sestrojen.

Úkoly

1. Podle odvození (derivace) na konci posledního příkladu sestrojte derivační strom. V tomto derivačním stromě sledujte, co se děje v souvislosti s rozpoznáváním slova v ekvivalentním automatu. Dá se říct, že automat prochází určitým způsobem tento derivační strom?
2. Sestrojte zásobníkové automaty ekvivalentní s těmito gramatikami:

$G_1 = (\{S\},\ \{a, b\},\ P,\ S)$
$S \to aSa \mid bSb \mid c$

$G_2 = (\{S, A, B\},\ \{a, b\},\ P,\ S)$
$S \to bAB \mid aBA \mid \varepsilon$
$A \to abAab \mid \varepsilon$
$B \to baBba \mid \varepsilon$

Kapitola 4

Jazyky typu 0

Jazyky typu 0 v Chomského hierarchii nám popisují vše, co je vypočitatelné, algoritmizovatelné. Problém je vypočitatelný (algoritmizovatelný), právě když existuje jazyk typu 0, který jej popisuje. Protože jazyky typu 0 jsou rozpoznávány Turingovými stroji, můžeme říct, že problém je vypočitatelný právě tehdy a jen tehdy, pokud lze sestrojit Turingův stroj, který jej počítá.

4.1 Turingův stroj

4.1.1 Co je to Turingův stroj – pojmy a značení

Z přednášek víme, že Turingův stroj je abstraktní výpočetní model. Je určen takto:

✎ $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F, \{-1, 0, 1\})$, kde

- Q je neprázdná konečná *množina stavů*,
- Σ je neprázdná konečná *vstupní abeceda*,
- Γ je neprázdná konečná *pásková (pracovní) abeceda* a platí $\Sigma \subset \Gamma$,
- δ je *přechodová funkce*, která musí být vypočitatelná,
- F je neprázdná množina *koncových stavů*, platí $F \subseteq Q$,
- $\{-1, 0, 1\}$ je určení možných *pohybů čtecí a zápisové hlavy na pásce* (vlevo, stát, vpravo), také může být například $\{L, S, R\}$.

Vstupní abeceda obsahuje ty symboly, ze kterých se může skládat vstupní slovo, jazyk rozpoznávaný Turingovým strojem je právě nad touto abecedou. Je zřejmé, že pásková abeceda obsahuje i takové symboly, které nejsou ve vstupní abecedě. Je to především speciální páskový symbol označující buňku, ve které není nic zapsáno, obvykle se značí B (ze slova Blank – „prázdný“) nebo $\sqcup$. Dále tam mohou být symboly určující začátek a konec obsazené části pásky (pokud jsou používány), obvykle symboly \$ nebo #.

Přechodová funkce má formu $\delta(p, a) = (q, b, M)$, kde

- $p, q \in Q$ je výchozí a cílový stav přechodu,
- $a, b \in \Gamma$ jsou symboly páskové abecedy, předpis určuje, že přečtený symbol a má být přepsán symbolem b,
- $M \in \{-1, 0, 1\}$ určuje pohyb hlavy – určujeme, zda se má pokračovat vlevo, zůstat na místě nebo se posunout o pole doprava.

Příklad

Například konfigurace $(abbca, q, daab)$ znamená, že se stroj nachází ve stavu q, na pásce je slovo $abbcadaab$ a čtecí a zápisová hlava ukazuje na šestý symbol slova – d.

Konfigurace je (α, q, β), kde $\alpha \cdot \beta$ je slovo na pásce, $\alpha, \beta \in \Gamma^*$, výpočet je ve stavu $q \in Q$ a čtecí/zápisová hlava ukazuje na první symbol řetězce β. Také lze zapsat jako $\alpha q \beta$.

Počáteční konfigurace je (ε, q_0, w_0), kde q_0 je počáteční stav a $w_0 \in \Sigma^*$ je slovo na vstupu. Koncová konfigurace je (w_1, q_f, w_2), kde $q_f \in F,\ w_1, w_2 \in \Gamma^*$. Slovo $w_1 \cdot w_2$ lze brát jako výstupní řetězec, pokud je nějaký výstup požadován. Turingův stroj totiž nejen určuje, zda dané slovo patří do jazyka jím rozpoznávaného, ale může provádět také nejrůznější výpočty včetně složitých matematických operací – vždyť pro cokoliv vypočitatelného lze vytvořit Turingův stroj.

Přechod mezi konfiguracemi (přesněji – jde o relaci, vzpomeňte si, co to znamená v řeči matematiky) je definován pomocí přechodové funkce, podobně jako u jiných typů automatů, jen je trochu komplikovanější, protože existuje více možností posunu čtecí/zápisové hlavy. Relace přechodu mezi konfiguracemi je určena takto:

$$(\alpha, q_i, a\beta) \vdash (\alpha b, q_j, \beta) \quad \Leftrightarrow \quad \delta(q_i, a) = (q_j, b, 1) \tag{4.1}$$

$$(\alpha, q_i, a\beta) \vdash (\alpha, q_j, b\beta) \quad \Leftrightarrow \quad \delta(q_i, a) = (q_j, b, 0) \tag{4.2}$$

$$(\alpha c, q_i, a\beta) \vdash (\alpha, q_j, cb\beta) \quad \Leftrightarrow \quad \delta(q_i, a) = (q_j, b, -1) \tag{4.3}$$

V prvním případě se čtecí a zápisová hlava posunuje doprava, v druhém zůstává na místě (tj. v dalším kroku bude číst totéž políčko jako v tomto) a v třetím se posunuje doleva.

4.1.2 Navrhujeme Turingův stroj pro daný jazyk

Při návrhu Turingova stroje postupujeme následovně:

- zjistíme si, které části slova mají být „synchronizovány“ – pokud je jejich délka nebo typ symbolu v nějakém vztahu, a také jakým způsobem,
- určíme stavy a odpovídající reakce, promyslíme si „strategii“ zpracování slova včetně přechodu do koncových stavů,
- sestavíme δ-funkci,
- otestujeme pro několik typických slov rozpoznávaného jazyka, především pro nejkratší slova.

Příklad

Sestrojíme Turingův stroj rozpoznávající jazyk $L = \{a^n b^n c^n \ :\ n \geq 0\}$

$\mathcal{M} = (\{q_0, q_P, q_A, q_B, q_C, q_f, q_{accept}\},\ \{a, b, c\},\ \{a, \bar{a}, b, \bar{b}, c, \bar{c}, \sqcup, \$\},\ \delta,\ \{q_{accept}\},\ \{-1, 0, 1\})$

Budeme postupovat takto:

1. označíme první symbol a na pásce (tj. přepíšeme symbolem $\bar{a}$),
2. najdeme první b, označíme ho,
3. pak najdeme první c, taktéž označíme,
4. přejdeme na začátek (postupujeme doleva, dokud nenajdeme nejbližší označené $\bar{a}$),
5. posuneme se o políčko dále na první neoznačené a,

6. pokud takové existuje, pak návrat k bodu 1, jinak bod 7,
7. po označení všech symbolů a a k nim příslušejících b a c ještě zkontrolujeme, zda neexistují některé neoznačené symboly b a c a ukončíme výpočet.

Cyklus naznačený v bodech 1 až 6 bude probíhat postupně přes všechny symboly a ze vstupu, tedy n-krát.

Jednotlivé stavy fungují jako „běhová informace“ (sdělují nám, ve které fázi cyklu se výpočet nachází), znamenají:

- q_A – označili jsme a, přeskakujeme symboly $a, \bar{b}$, hledáme první neoznačené b,
- q_B – označili jsme b, přeskakujeme symboly $b, \bar{c}$, hledáme první neoznačené c,
- q_C – označili jsme c, vracíme se na začátek k poslednímu označenému $\bar{a}$, při pohybu doleva přeskakujeme všechny symboly $\bar{c}, b, \bar{b}, a$.

Definujeme přechodovou funkci:

$\delta(q_0, \$) = (q_{accept}, 0)$ (přijali jsme prázdné slovo)

$\delta(q_0, a) = (q_A, \bar{a}, 1)$
$\delta(q_P, a) = (q_A, \bar{a}, 1)$
$\delta(q_A, a) = (q_A, a, 1)$
$\delta(q_A, \bar{b}) = (q_A, \bar{b}, 1)$
$\delta(q_A, b) = (q_B, \bar{b}, 1)$

$\delta(q_B, b) = (q_B, b, 1)$
$\delta(q_B, \bar{c}) = (q_B, \bar{c}, 1)$
$\delta(q_B, c) = (q_C, \bar{c}, -1)$
$\delta(q_C, X) = (q_C, X, -1)$,
$X \in \{a, b, \bar{b}, \bar{c}\}$

$\delta(q_C, \bar{a}) = (q_P, \bar{a}, 1)$
$\delta(q_0, \bar{b}) = (q_f, \bar{b}, 1)$
$\delta(q_f, \bar{b}) = (q_f, \bar{b}, 1)$
$\delta(q_f, \bar{c}) = (q_f, \bar{c}, 1)$
$\delta(q_f, \$) = (q_{accept}, \$, 0)$

Ukázka zpracování slova abc:

$(\varepsilon, q_0, abc) \vdash (\bar{a}, q_A, bc) \vdash (\bar{a}\bar{b}, q_B, c) \vdash (\bar{a}, q_C, \bar{b}\bar{c}) \vdash (\varepsilon, q_C, \bar{a}\bar{b}\bar{c}) \vdash (\bar{a}, q_0, \bar{b}\bar{c}) \vdash (\bar{a}\bar{b}, q_f, \bar{c}) \vdash$
$\vdash (\bar{a}\bar{b}\bar{c}, q_f, \varepsilon) \vdash (\bar{a}\bar{b}\bar{c}, q_{accept}, \varepsilon)$

Jestliže zaznamenáme i hraniční symboly (někdy to může být přehlednější), zpracování bude vypadat takto:

$(\$, q_0, abc\$) \vdash (\$\bar{a}, q_A, bc\$) \vdash (\$\bar{a}\bar{b}, q_B, c\$) \vdash (\$\bar{a}, q_C, \bar{b}\bar{c}\$) \vdash (\$, q_C, \bar{a}\bar{b}\bar{c}\$) \vdash (\$\bar{a}, q_0, \bar{b}\bar{c}\$) \vdash$
$\vdash (\$\bar{a}\bar{b}, q_f, \bar{c}\$) \vdash (\$\bar{a}\bar{b}\bar{c}, q_f, \$) \vdash (\$\bar{a}\bar{b}\bar{c}, q_{accept}, \$)$

Ukázka zpracování slova $aabbcc$:

$(\varepsilon, q_0, aabbcc) \vdash (\bar{a}, q_A, abbcc) \vdash (\bar{a}a, q_A, bbcc) \vdash (\bar{a}a\bar{b}, q_B, bcc) \vdash (\bar{a}a\bar{b}b, q_B, cc) \vdash$
$\vdash (\bar{a}a\bar{b}, q_C, b\bar{c}c) \vdash (\bar{a}a, q_C, \bar{b}b\bar{c}c) \vdash (\bar{a}, q_C, a\bar{b}b\bar{c}c) \vdash (\varepsilon, q_C, \bar{a}a\bar{b}b\bar{c}c) \vdash (\bar{a}, q_0, a\bar{b}b\bar{c}c) \vdash$
$\vdash (\bar{a}\bar{a}, q_A, \bar{b}b\bar{c}c) \vdash (\bar{a}\bar{a}\bar{b}, q_A, b\bar{c}c) \vdash (\bar{a}\bar{a}\bar{b}\bar{b}, q_B, \bar{c}c) \vdash (\bar{a}\bar{a}\bar{b}\bar{b}\bar{c}, q_B, c) \vdash (\bar{a}\bar{a}\bar{b}\bar{b}, q_C, \bar{c}\bar{c}) \vdash$
$\vdash (\bar{a}\bar{a}\bar{b}, q_C, \bar{b}\bar{c}\bar{c}) \vdash (\bar{a}\bar{a}, q_C, \bar{b}\bar{b}\bar{c}\bar{c}) \vdash (\bar{a}, q_C, \bar{a}\bar{b}\bar{b}\bar{c}\bar{c}) \vdash (\bar{a}\bar{a}, q_0, \bar{b}\bar{b}\bar{c}\bar{c}) \vdash (\bar{a}\bar{a}\bar{b}, q_f, \bar{b}\bar{c}\bar{c}) \vdash$
$\vdash (\bar{a}\bar{a}\bar{b}\bar{b}, q_f, \bar{c}\bar{c}) \vdash (\bar{a}\bar{a}\bar{b}\bar{b}\bar{c}, q_f, \bar{c}) \vdash (\bar{a}\bar{a}\bar{b}\bar{b}\bar{c}\bar{c}, q_f, \varepsilon) \vdash (\bar{a}\bar{a}\bar{b}\bar{b}\bar{c}\bar{c}, q_{accept}, \varepsilon)$

Ukázka zpracování slova ac:

$(\varepsilon, q_0, ac) \vdash (\bar{a}, q_A, c)$ chyba $\Rightarrow$ slovo ac není přijato.

Ukázka zpracování slova ε:

$(\varepsilon, q_0, \varepsilon) \vdash (\varepsilon, q_{accept}, \varepsilon)$

Příklad

Vytvoříme Turingův stroj rozpoznávající jazyk $L = \{wcw \ : \ w \in \{a, b\}^*\}$

Opět si nejdřív promyslíme, jak by měl stroj postupovat. Předpokládejme, že na vstupu je slovo *abbcabb*. Nejdřív označíme symbol v první polovině slova, pak se přesuneme do druhé části slova a najdeme párový symbol. Potom je třeba se přesunout zpět do první části slova, označit další symbol, atd. Může to vypadat například takto:

začátek zpracování:	$abb\ c\ abb$
označíme první symbol:	$xbb\ c\ abb$
označíme k němu párový symbol:	$xbb\ c\ xbb$
podobně po druhém kroku:	$xxb\ c\ xxb$
po třetím kroku:	$xxx\ c\ xxx$

V předchozím příkladu jsme počítali s tím, že slovo na vstupu je ohraničeno symboly \$. Nyní si ukážeme postup, ve kterém slovo není těmito symboly ohraničeno, ale za posledním symbolem slova jsou symboly $\sqcup$. Když se nad tím zamyslíme, obě řešení jsou ekvivalentní, protože $\sqcup$ můžeme také chápat jako hraniční symboly a stejně s nimi zacházet.

Použijeme tyto stavy:

- q_0 – označíme symbol v první části slova, na který ukazuje čtecí/zápisová hlava, přejdeme do stavu q_A nebo q_B podle toho, zda jsme označili symbol a nebo b, posun vpravo,
- q_A – byl označen symbol a, posunujeme se v cyklu doprava a hledáme polovinu slova (symbol c, tedy přeskakujeme všechny symboly a a b, které jsou před c),
- q_B – totéž pro b,
- q_{AC} – pokračovatel stavu q_A, přecházíme do něj, pokud ve stavu q_A najdeme symbol c, hledáme párový symbol (symbol a v druhé části slova, tedy přeskakujeme vše, co bylo označeno – symboly x),
- q_{BC} – podobně ze stavu q_B,
- q_{ZC} – v druhé části slova jsme ve stavech q_{AC} nebo q_{BC} našli párový symbol, označili, teď se v cyklu posunujeme zpět doleva a hledáme polovinu slova (symbol c, přeskakujeme vše označené – symboly x),
- q_Z – ve stavu q_{ZC} jsme našli polovinu slova, nyní budeme pokračovat doleva a hledat poslední označený symbol, přeskakujeme symboly a a b, až najdeme nejpravější x, změníme stav na q_0, posuneme se o krok doprava (tj. čtecí/zápisová hlava bude ukazovat na první neoznačený symbol nebo v případě všech označených na c) a v cyklu se vracíme na začátek tohoto postupu,
- q_f – do tohoto stavu přecházíme, pokud jsme ve stavu q_0 místo neoznačeného symbolu a nebo b našli přímo symbol c, tj. první část slova je celá označená; nyní musíme v cyklu směrem doprava projít zbytek slova, zda tam nezůstaly neoznačené symboly,
- q_{accept} – do tohoto stavu přejdeme, pokud ve stavu q_f najdeme konec slova (symbol $\sqcup$), což znamená, že je označena i celá druhá část.

Turingův stroj bude definován následovně:
$\mathcal{M} = (\{q_0, q_A, q_{AC}, q_B, q_{BC}, q_Z, q_{ZC}, q_f, q_{accept}\},\ \{a, b, c\},\ \{a, b, c, x, \sqcup\},\ \delta,\ \{q_{accept}\},\ \{-1, 0, 1\})$

Nalezeno a:
$\delta(q_0, a) = (q_A, x, 1)$
$\delta(q_A, Y) = (q_A, Y, 1), \;\; Y \in \{a, b\}$
$\delta(q_A, c) = (q_{AC}, c, 1)$
$\delta(q_{AC}, x) = (q_{AC}, x, 1)$
$\delta(q_{AC}, a) = (q_{ZC}, x, -1)$

Nalezeno b:
$\delta(q_0, b) = (q_B, x, 1)$
$\delta(q_B, Y) = (q_B, Y, 1), \;\; Y \in \{a, b\}$
$\delta(q_B, c) = (q_{BC}, c, 1)$
$\delta(q_{BC}, x) = (q_{BC}, x, 1)$
$\delta(q_{BC}, b) = (q_{ZC}, x, -1)$

Návrat doleva:
$\delta(q_{ZC}, x) = (q_{ZC}, x, -1)$
$\delta(q_{ZC}, c) = (q_Z, c, -1)$
$\delta(q_Z, Y) = (q_Z, Y, -1), \;\; Y \in \{a, b\}$
$\delta(q_Z, x) = (q_0, x, 1)$

Ukončení s kontrolou:
$\delta(q_0, c) = (q_f, c, 1)$
$\delta(q_f, x) = (q_f, x, 1)$
$\delta(q_f, \sqcup) = (q_{accept}, \sqcup, 0)$

Ukázka zpracování slova aca:
$(\varepsilon, q_0, aca) \vdash (x, q_A, ca) \vdash (xc, q_{AC}, a) \vdash (x, q_{ZC}, cx) \vdash (\varepsilon, q_Z, xcx) \vdash (x, q_0, cx) \vdash (xc, q_f, x) \vdash$
$\vdash (xcx, q_f, \sqcup) \vdash (xcx, q_{accept}, \sqcup)$

Ukázka zpracování slova c (to je nejkratší slovo jazyka):
$(\varepsilon, q_0, c) \vdash (c, q_f, \sqcup) \vdash (c, q_{accept}, \sqcup)$

Poznámka:

Zatímco konečné a zásobníkové automaty primárně slouží spíše jen k rozpoznávání slov jazyka (určujeme, zda slovo patří nebo nepatří do jazyka), Turingův stroj může být i nástrojem transformace – slovo na vstupu zpracuje na řetězec, který pak bude výstupem (výsledkem výpočtu). Může zpracovávat textové řetězce nebo třeba provádět matematické výpočty.

Příklad

Sestrojíme Turingův stroj, který pro zadaný vstup vytvoří jeho reverzi (tj. pro slovo w na vstupu bude na výstupu w^R). Na vstupu mohou být jakákoliv slova nad abecedou $\{a, b\}$.

Předpokládejme, že na vstupu je slovo $aabab$. Nejdřív na konec slova přidáme symbol #, kterým označíme rozhraní mezi původním a novým slovem, a pak začneme kopírovat původní slovo na pásku za #. Původní načítáme zprava (od konce, přitom přepisujeme symboly, abychom poznali, co už je zkopírováno), nové slovo naopak tvoříme zleva, aby bylo revertováno (převráceno):

začátek zpracování:	$aabab$
označíme hranici:	$aabab\ \#$
označíme první symbol:	$aabax\ \#$
zapíšeme k němu párový symbol:	$aabax\ \#\ b$
podobně po druhém kroku:	$aabxx\ \#\ ba$
po třetím kroku:	$aaxxx\ \#\ bab$
po čtvrtém kroku:	$axxxx\ \#\ baba$
po pátém kroku:	$xxxxx\ \#\ babaa$
po ukončení (proveden úklid):	$babaa$

Stavy:

- q_0 – projdeme celé slovo, na konec přidáme #, pak „couvnutí“ před # a přechod do stavu q_Z,
- q_Z – v cyklu postupujeme doleva a přeskakujeme označené symboly, zastavíme se na nejbližším neoznačeném symbolu, pak přechod do q_A nebo q_B,
- q_A – právě jsme označili a, v cyklu postupujeme doprava a přeskakujeme označené symboly, zastavíme se až na symbolu #, přechod do q_{AH},
- q_B – podobně jako q_A, ale pro symbol b,
- q_{AH} – v cyklu postupujeme doprava a přeskakujeme všechny zapsané symboly z pravé části slova, hledáme konec, pak zapíšeme a a přecházíme do stavu q_{ZH},
- q_{BH} – podobně jako q_{AH}, ale pro symbol b,
- q_{ZH} – v cyklu postupujeme doleva až k symbolu #, přeskakujeme všechny zapsané symboly a a b, pak přejdeme do stavu q_Z,
- q_f – úklid po kopírování, smažeme původní slovo a dodaný hraniční symbol #,
- q_{accept} – koncový stav.

$\mathcal{M} = (\{q_0, q_A, q_{AH}, q_B, q_{BH}, q_Z, q_{ZH}, q_f, q_{accept}\},\ \{a,b,c\},\ \{a,b,x,\sqcup,\#\},\ \delta,\ \{q_{accept}\},\ \{-1,0,1\})$

Začátek – příprava:
$\delta(q_0, Y) = (q_0, Y, 1),\ \ Y \in \{a,b\}$
$\delta(q_0, \sqcup) = (q_Z, \#, -1)$

Ukončení s úklidem:
$\delta(q_Z, \sqcup) = (q_f, \sqcup, 1)$
$\delta(q_f, x) = (q_f, \sqcup, 1)$
$\delta(q_f, \#) = (q_{accept}, \sqcup, 0)$

Posun zpět – doleva:
$\delta(q_Z, x) = (q_Z, x, -1)$
$\delta(q_Z, a) = (q_A, x, 1)$

$\delta(q_Z, b) = (q_B, x, 1)$
$\delta(q_{ZH}, Y) = (q_{ZH}, Y, -1)$
$\delta(q_{ZH}, \#) = (q_Z, \#, -1)$

Našli jsme a:
$\delta(q_A, x) = (q_A, x, 1)$
$\delta(q_A, \#) = (q_{AH}, \#, 1)$
$\delta(q_{AH}, Y) = (q_{AH}, Y, 1)$
$\delta(q_{AH}, \sqcup) = (q_Z H, a, -1)$

Našli jsme b:
$\delta(q_B, x) = (q_B, x, 1)$
$\delta(q_B, \#) = (q_{BH}, \#, 1)$
$\delta(q_{BH}, Y) = (q_{BH}, Y, 1)$
$\delta(q_{BH}, \sqcup) = (q_Z H, b, -1)$

Poznámka:

Všimněme si rozdílu mezi činností Turingova stroje a dříve definovaných automatů:

- Turingův stroj nejen čte vstupní pásku, může ji i zapisovat,
- čtecí a zápisová hlava se může (nemusí) pohybovat různými směry, nejen doprava,
- nepotřebujeme zásobník, ale přesto můžeme uchovávat i jinou informaci než označení stavu, ve kterém právě jsme – kamkoliv na pásku si můžeme cokoliv poznamenat a později tuto informaci využít,
- na Turingově stroji lze provádět i výpočty.

Turingovými stroji se budeme podrobněji zabývat v navazujícím předmětu Teorie vyčíslitelnosti a složitosti.

Úkoly

1. Sestrojte Turingovy stroje rozpoznávající jazyky
   - $L_1 = \{a^n b^n \ : \ n \geq 0\}$
   - $L_2 = \{a^n b^{2n} a^n \ : \ n \geq 1\}$
   - $L_3 = \{wcwcw \ : \ w \in \{a,b\}^*\}$
   - $L_4 = \{wcw^R cw \ : \ w \in \{a,b\}^*\}$
2. Navrhněte (alespoň rámcově, popisem činnosti) Turingův stroj, který zdvojí svůj vstup (tj. například pokud je na vstupu slovo *abb*, na výstupu bude slovo *abbabb*).
3. Navrhněte Turingův stroj, který bude mít na vstupu číslo v desítkové soustavě a jeho výstupem bude toto číslo vynásobené dvěma.

4.2 Gramatiky typu 0

4.2.1 Návrh gramatiky typu 0

Gramatika typu 0 je taková gramatika, jejíž pravidla jsou ve tvaru

$$\alpha \ \rightarrow \ \beta, \quad \alpha \in (N \cup T)^* N (N \cup T)^*, \ \beta \in (N \cup T)^*$$

Jde tedy o obecné gramatiky, kde v předpisu máme jediný požadavek – na levé straně pravidla musí být řetězec terminálních a neterminálních symbolů obsahující alespoň jeden neterminál.

Příklad

Následující gramatika generuje jazyk $L = \left\{a^{2^n} \ : \ n \geq 1\right\}$.
$G = (\{S, A, B, C, D, E\}, \ \{a\}, \ P, \ S)$
$S \rightarrow ACaB$
$Ca \rightarrow aaC$
$CB \rightarrow DB$
$CB \rightarrow E$
$aD \rightarrow Da$
$AD \rightarrow AC$
$aE \rightarrow Ea$
$AE \rightarrow \varepsilon$

Odvodíme nejkratší slovo generované gramatikou:
$S \Rightarrow ACaB \Rightarrow AaaCB \Rightarrow AaaE \Rightarrow AaEa \Rightarrow AEaa \Rightarrow aa$

Poznámka:

Všimněte si „stěhování“ symbolu E doleva pomocí pravidla $AE \rightarrow EA$. Takový postup není v bezkontextových gramatikách možný, naopak v gramatikách typu 0 a 1 je běžný, „posouvaný“ symbol plní podobnou funkci jako stav v automatech.

Příklad

Sestrojíme gramatiku generující jazyk $L = \{a^n b^n c^n \ : \ n \geq 1\}$.

Na začátku vytvoříme neterminálovou „kostru“. Použijeme neterminály H_1 a H_2 jako hranice částí slova obsahujících různé symboly, dále neterminály A a B jako „běžce“ určující právě zpracovávanou část slova (přidáváme buď zároveň symboly a a b, a nebo symbol c), dále neterminál Z pro návrat, kdy se připravujeme na generování další trojice symbolů a, b, c, a konečně neterminál E zajišťující správné ukončení generování slova a odstranění hraničních symbolů.

Postup si můžeme nastínit na sekvenci větných forem, které by podle pravidel gramatiky byly generovány – viz tabulku 4.1.

začátek generování další trojice a, b, c	$aaaA\ H_1\ \ bbb\ \ H_2\ ccc$
vygenerujeme a, b, posun zpracování doprava	$aaaa\ H_1\ bBbbb\ H_2\ ccc$
posun zpracování doprava	$aaaa\ H_1\ bbBbb\ H_2\ ccc$
postupně „přeskočíme“ všechna b ...	$aaaa\ H_1\ bbbbB\ H_2\ ccc$
vygenerujeme c za symbolem H_2, posun doleva	$aaaa\ H_1\ bbbbZ\ H_2\ cccc$
posun doleva, přeskočíme b	$aaaa\ H_1\ bbbZb\ H_2\ cccc$
pořád doleva ...	$aaaa\ H_1\ Zbbbb\ H_2\ cccc$
navážeme na začátek	$aaaaA\ H_1\ \ bbbb\ \ H_2\ cccc$

Tabulka 4.1: Postup generování slova v gramatice typu 0

Budeme postupovat takto:

- Nejdřív vygenerujeme hraniční symboly a neterminál A.
  $S \to AH_1H_2$
- Vygenerujeme a a b, přesun doprava.
  $AH_1 \to aH_1bB$
- Posouvání symbolu B doprava:
  $Bb \to bB$
- Konec rekurze z předchozího bodu, vygenerujeme c, posun zpět doleva.
  $BH_2 \to ZH_2c$
- Posun symbolu Z doleva:
  $bZ \to Zb$
- Konec rekurze z předchozího bodu, navážeme na první bod.
  $H_1Z \to AH_1$
- Zajistíme správné ukončení generování terminálního slova.
  $AH_1 \to abE$
  $Eb \to bE$
  $EH_2 \to c$

Celá gramatika:
$G = (\{S, A, B, H_1, H_2, Z, E\},\ \{a, b, c\},\ P,\ S)$
$S \to AH_1H_2 \mid abE$
$AH_1 \to aH_1bB$

$Bb \to bB$
$BH_2 \to ZH_2c$
$bZ \to Zb$
$H_1Z \to AH_1$
$Eb \to bE$
$EH_2 \to c$

Vygenerujeme slovo $a^2b^2c^2$, modře je vyznačena část slova přepisovaná v dalším kroku některým pravidlem:
$S \Rightarrow AH_1H_2 \Rightarrow aH_1bBH_2 \Rightarrow aH_1bZH_2c \Rightarrow aH_1ZbH_2c \Rightarrow aAH_1bH_2c \Rightarrow aabEbH_2c \Rightarrow$
$\Rightarrow aabbEH_2c \Rightarrow aabbcc$

Úkoly

1. V první gramatice této sekce (str. 66) vygenerujte slovo $a^{2^2} = a^4$.
2. Ke gramatice z téhož příkladu přidejte ještě jedno pravidlo tak, aby generovala jazyk $L = \{a^{2^n} \ : \ n \geq 0\}$.
3. V gramatice z posledního příkladu (str. 67) vygenerujte slovo $a^3b^3c^3$.
4. Gramatiku z téhož příkladu upravte tak, aby generovala jazyk
   - $L_1 = \{a^n b^n c^n \ : \ n \geq 0\}$
   - $L_2 = \{a^{2n} b^n c^{2n} \ : \ n \geq 1\}$

4.2.2 Kurodova normální forma

Obecný tvar pravidla může být trochu problém především v důkazech. Proto v případě, že gramatiku 0 chceme využít pro některý účel, ji převedeme do normální formy. Pro gramatiky typu 0 existuje *Kurodova normální forma* (anglicky Kuroda Normal Form).

Gramatika typu 0 je v *Kurodově normální formě (KNF) pro jazyky typu 0*, jestliže pro všechna její pravidla $\alpha \to \beta$ platí
$|\alpha| \leq 2, \ |\beta| \leq 2$
(a samozřejmě musí splňovat předpis pro pravidla gramatiky typu 0, tedy alespoň jeden neterminál na levé straně pravidla). Obvykle se předpokládá, že všechna pravidla gramatiky $G = (N, T, P, S)$ v KNF splňují jeden z těchto tvarů:

- $AB \to CD, \quad A, B, C, D \in N$
- $A \to CD, \quad A, C, D \in N$
- $A \to a, \quad A \in N, \ a \in T$
- $A \to \varepsilon, \quad A \in N$

Všimněte si, že se připouští i jeden typ zkracujícího pravidla (poslední typ), protože gramatiky typu 0 obecně nejsou nezkracující (to jsou až gramatiky typu 1).

Vidíme, že předpis druhého a třetího typu pravidla je podobný předpisu pro Chomského normální formu. Pokud se jedná o pravidlo bezkontextového typu (i taková mohou být v gramatice typu 0), pak pro takové pravidlo můžeme použít algoritmus pro převod do Chomského normální formy.

Postup převodu gramatiky typu 0 do KNF:

1. Odstraníme jednoduchá pravidla.
2. Všechny terminály $a \in T$ (na levé i pravé straně pravidla) nahradíme „pomocnými“ neterminály N_a.
3. Pro všechny neterminály vytvořené v předchozím bodu přidáme pravidlo $N_a \to a$.
4. Pravidla $A \to B_1B_2 \dots B_n,\ n > 2$ nahradíme pravidly

$$\begin{aligned} A &\to B_1X_1 \\ X_1 &\to B_2X_2 \\ &\vdots \\ X_{n-2} &\to B_{n-1}B_n \end{aligned}$$

   Tedy pravidla bezkontextového typu zpracováváme stejně jako pro Chomského NF.
5. Pravidla $A_1A_2 \dots A_m \to B_1B_2 \dots B_m,\ m > 2$ (obě strany pravidla stejně dlouhé) nahradíme pravidly

$$\begin{aligned} A_1A_2 &\to B_1X_1 \\ X_1A_3 &\to B_2X_2 \\ &\vdots \\ X_{m-2}A_m &\to B_{m-1}B_m \end{aligned}$$

6. Nezkracující pravidla $A_1A_2 \dots A_m \to B_1B_2 \dots B_n,\ 2 < m \leq n$ nahradíme pravidly

$$\begin{aligned} A_1A_2 &\to B_1X_1 \\ X_1A_3 &\to B_2X_2 \\ &\vdots \\ X_{m-2}A_m &\to B_{m-1}X_{m-1} \end{aligned} \qquad \begin{aligned} X_{m-1} &\to B_mX_m \\ X_m &\to B_{m+1}X_{m+1} \\ &\vdots \\ X_{n-2} &\to B_{n-1}B_n \end{aligned}$$

7. Zkracující pravidla $A_1A_2 \dots A_m \to B_1B_2 \dots B_n,\ n < m$ nahradíme pravidly

$$\begin{aligned} A_1A_2 &\to B_1X_1 \\ X_1A_3 &\to B_2X_2 \\ &\vdots \\ X_{n-1}A_{n+1} &\to B_nX_n \end{aligned} \qquad \begin{aligned} X_nA_{n+2} &\to X_{n+1} \\ &\vdots \\ X_{m-3}A_{m-1} &\to X_{m-2} \\ X_{m-2}A_m &\to \varepsilon \end{aligned}$$

Příklad

Postup si ukážeme na gramatice

$S \to AaBC$	①
$C \to cBAa \mid bS$	②,③
$AaBc \to d$	④
$BA \to abcd$	⑤

V pravidlech ④ a ⑤ provedeme náhradu terminálů $a \in T$ novými neterminály N_a. Pravidlo ③, nyní $C \to N_b S$, nemusíme dále zpracovávat, už odpovídá KNF.

Nejdřív zpracujeme pravidla bezkontextového typu ① a ②.

$S \to AX_1$
$X_1 \to N_a X_2$
$X_2 \to BC$
$C \to N_c X_3$
$X_3 \to BX_4$
$X_4 \to AN_a$

Zbývají pravidla ③ a ④ – jedno je zkracující a druhé nezkracující:

$BA \to N_a X_5$
$N_a X_5 \to N_b X_6$
$X_6 \to N_c N_d$
$AN_a \to N_d X_7$
$X_7 B \to X_8$
$X_8 N_c \to \varepsilon$

Celá gramatika po převodu do KNF:

$S \to AX_1$
$X_1 \to N_a X_2$
$X_2 \to BC$
$C \to N_c X_3$
$X_3 \to BX_4$
$X_4 \to AN_a$
$BA \to N_a X_5$
$N_a X_5 \to N_b X_6$
$X_6 \to N_c N_d$
$AN_a \to N_d X_7$
$X_7 B \to X_8$
$X_8 N_c \to \varepsilon$
$N_a \to a$
$N_b \to b$
$N_c \to c$
$N_d \to d$

Úkoly

1. Zpracujte následující pravidla tak, aby odpovídala požadavkům na KNF a přitom generovala ekvivalentní řetězec:
   - $A \to BCabD$
   - $BA \to acbA$
   - $CaD \to b$
2. Převeďte do KNF gramatiky z příkladů na stranách 66 a 67.
3. Převeďte do tvaru KNF následující gramatiku:

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

   $S \to aAB \mid BA \mid C$
   $aAa \to bCba \mid B$
   $AC \to d$
   $B \to aB \mid CC \mid b$
   $C \to cS \mid d$