Sbírka řešených příkladů z teorie automatů a formálních jazyků

Michal Chytil

UNIVERZITA KARLOVA

Sbírka řešených příkladů z teorie automatů a formálních jazyků

Michal Chytil

Praha 1987

Katedra kybernetiky a informatiky
matematicko-fyzikální fakulty Univerzity Karlovy

Vedoucí katedry: prof. RNDr. Milan Vlach, DrSc.

OBSAH

Ú V O D

Tato sbírka příkladů je určena především posluchačům oboru "Teoretická kybernetika, matematická informatika a teorie systémů" k procvičování látky probírané na přednášce "Teorie automatů a formálních jazyků", resp. na přednášce "Teorie automatů a vyčíslitelnosti", která má začít ve školním roce 1988/89. Poslouží i posluchačům učitelských kombinací "Matematika - výpočetní technika" a "Fyzika - výpočetní technika" ve 3. - 5. semestru k předmětu "Teoretické základy informatiky".

Většina příkladů je zaměřena na ovládnutí základních pojmů a algoritmů i když některé příklady vyžadují hlubší zamyšlení. Další (nefešené) příklady může čtenář nalézt např. ve slovenském překladu knihy [1] . Tématicky příklady odpovídají zhruba kapitolám 1 - 5 knihy [2] . Jedná se většinou o příklady, které jsem po několik let používal při cvičeních k přednášce z teorie automatů a formálních jazyků. Proto některá zadání najdete ve skriptech [3] .

S povděkem přijmu upozornění na chyby, které jsem snad přehlédl, i na další m ná zlepšení sbírky.

Mé upřímné poděkování patří Kateřině Hlaváčkové za asistenci při sestavová- ní této sbírky i za její pozorné a pečlivé přepsání na stroji.

Michal Chytil

29. 5. 1987

1. KONEČNÉ AUTOMATY A REGULÁRNÍ JAZYKY

Reprezentace konečných automatů

1.1

Konečný automat zadaný tabulkou

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

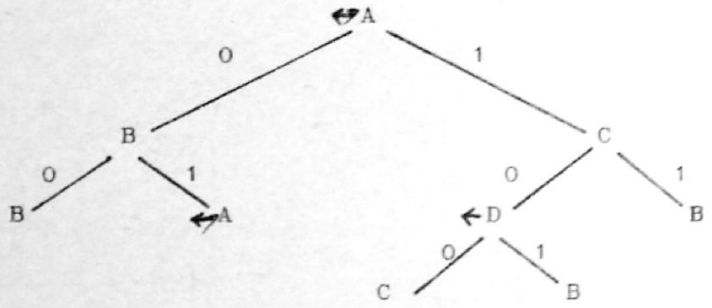
zadejte

a) grafem,

b) stromem.

1.2

Konečný automat zadaný stromem



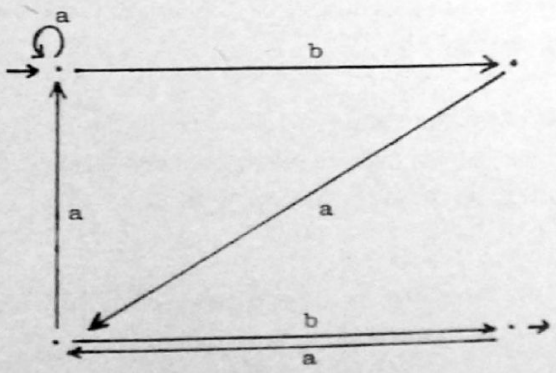
zadejte

a) tabulkou,

b) grafem.

1.3

Graf následujícího parciálního automatu doplňte na graf automatu totálního.



1.4 Určete posloupnost stavů, do kterých se dostane automat

	0	1
→ q_0	q_1	q_0
q_1	q_2	q_0
← q_2	q_2	q_2

při zpracování vstupního slova
a) 0110111001,
b) 101100111.

1.5 Jazyk rozpoznávaný automatem z příkladu 1.4 je tvořen právě slovy v abecedě $\{0,1\}$ obsahujícími podslovo 00. Najděte podobnou slovní charakterizaci pro jazyky rozpoznatelné následujícími automaty.

a)

	0	1
↔ q_0	q_1	q_0
q_1	q_0	q_1

b)

	0	1
→ q_0	q_1	q_0
← q_1	q_2	q_1
← q_2	q_0	q_2

c)

	a	b
→ q_0	q_0	q_1
q_1	q_0	q_2
← q_2	q_0	q_2

d)

	0	1
→ q_0	q_1	q_0
← q_1	q_2	q_0
← q_2	q_2	q_0

e)

	0	1
→ A	B	A
B	A	C
← C	B	C

f)

	a	b
← 1	1	2
→ 2	1	3
← 3	2	3

1.6 Pro každý z následujících jazyků sestrojte automat, který jej rozpoznává. Vstupní abeceda je ve všech případech $\{0,1\}$. Jazyky jsou tvořeny právě všemi slovy splňujícími uvedené podmínky.

a) Počet jedniček, které slovo obsahuje, je dělitelný dvěma nebo třemi.
b) Slovo je zakončeno dvěma nebo více symboly 1 předcházenými aspoň jednou 0.
c) Slovo končí symbolem 1 a má délku $3k+1$ pro nějaké $k \geq 0$.

1.7 Automat přijímá všechna slova v jisté abecedě, jejichž délka je dělitelná pěti. Kolik musí mít minimálně stavů?

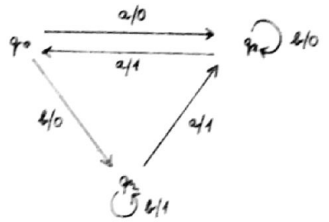
1.8 Autonomním automatem nazveme každý automat s jednoprvkovou abecedou. Popište třídu jazyků rozpoznatelných autonomními automaty.

1.9 Mějme libovolný konečný automat se 4 stavy. Zvolme nějaký jeho vstupní symbol x. Ukažte, že slovo složené ze 17 symbolů x převede automat do téhož stavu jako vstupní slovo xxxxx.

1.10 Dokažte, že následující jazyky nejsou rozpoznatelné konečným automatem.
a) $L_1 = \{0^m 1^n 0^m ; 0 \leq m,n\}$,
b) $L_2 = \{ww ; w \in \{0,1\}^*\}$,
c) $L_3 = \{w(w)^R ; w \in \{0,1\}^*\}$ (Symbolem $(w)^R$ označujeme slovo w psané pozpátku.)
d) $L_4 = \{0^p ; p$ je prvočíslo$\}$.

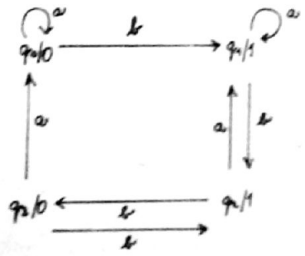
Mealyho a Mooreovy sekvenční stroje

1.11 Jakou výstupní posloupnost vydá Mealyho sekvenční stroj



v odpověď na vstupní posloupnost
a) abbabba,
b) bbbba,
začne-li činnost ve stavu q_1 ?

1.12 Jakou výstupní posloupnost vydá Mooreův sekvenční stroj



v odpověď na vstupní posloupnost
a) aabbaab,
b) babb,
začne-li činnost ve stavu q_2 ?

1.13

Definice: Nechť $M_1=(Q_1,\Sigma,\Delta_1,\delta_1,\lambda_1)$, $M_2=(Q_2,\Sigma,\Delta_2,\delta_2,\lambda_2)$ jsou dva Mealyho stroje se stejnou vstupní abecedou.

1) Nechť $q_1\in Q_1$, $q_2\in Q_2$. Stavy q_1,q_2 nazveme ekvivalentní (značíme $q_1\sim q_2$), jestliže pro libovolné $u\in\Sigma^*$, $x\in\Sigma$, platí: $\lambda_1(\delta_1^*(q_1,u),x)=\lambda_2(\delta_2^*(q_2,u),x)$.

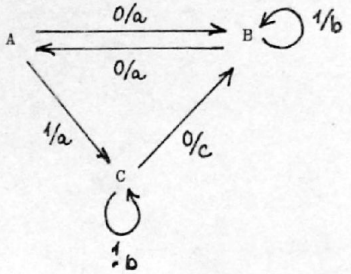
2) Stroj M_1 lze vnořit do M_2 (značíme $M_1\leq M_2$), jestliže ke každému $q_1\in Q_1$ existuje s ním ekvivalentní $q_2\in Q_2$.

3) Stroje M_1,M_2 jsou ekvivalentní ($M_1\sim M_2$), jestliže $M_1\leq M_2$ a $M_2\leq M_1$.

Předchozí definice se vztahuje i na Mooreovy stroje, neboť každý Mooreův stroj $M=(Q,\Sigma,\Delta,\delta,\mu)$ lze chápat jako Mealyho stroj, jehož výstupní funkce λ je dána vztahem $\lambda(q,x)=\mu(\delta(q,x))$, pro libovolné $q\in Q$, $x\in\Sigma$.

(1.14) Ukažte, že ke každému Mealyho stroji lze sestrojit s ním ekvivalentní Mooreův stroj.

(1.15) K Mealyho stroji



sestrojte ekvivalentní Mooreův stroj.

(1.16) Sestrojte Mealyho stroje s popsaným chováním. Vstupní i výstupní abeceda bude v obou případech $\{0,1\}$.

a) Odpovědí na libovolné první dva symboly je 0. Dále se každý n-tý výstupní symbol rovná (n-2)-mu vstupnímu symbolu.

b) Výstup 1 je vydán pokaždé, když na vstupu je člen skupiny jedniček, která je předcházena skupinou aspoň dvou nul. Ve všech ostatních případech vydá 0.

(1.17) (Sčítačka) Sestrojte Mealyho stroj, který realizuje sčítání dvojic binárních čísel. Čísla jsou zadávána od nejnižších k nejvyšším řádům. Ve stejném pořadí jsou vydávány číslice výsledku. V každém okamžiku je zadávána dvojice binárních číslic příslušejících ve sčítaných číslech témuž řádu. Např. binární čísla 01011, 10001 jsou sčítána takto:

časový okamžik	1	2	3	4	5
vstup	1 1	1 0	0 0	1 0	0 1
výstup	0	0	1	1	1

Vstupní abeceda je tedy $\{00,01,10,11\}$, výstupní abeceda je $\{0,1\}$.

(1.18) Sestrojte Mooreův stroj realizující sčítačku zadanou v předchozím příkladu.

(1.19) Sestrojte konečný automat, který rozpoznává jazyk $L=\{$binární zápis čísla n; n je dělitelné $5\}$ tak, že binární zápisy čísel čte odleva, tj. od nejvyšších řádů.

(1.20) Sestrojte Mealyho stroj, který realizuje celočíselné dělení binárních čísel pěti, tzn. binární zápis čísla n transformuje na binární zápis celé části čísla $n/5$ (předpokládáme, že n je nezáporné). Stroj čte číslo od nejvyšších řádů.

(1.21) Sestrojte Mealyho stroj, který realizuje násobení binárního čísla třemi. Číslo je čteno od nejnižších řádů.

(1.22) Mealyho stroj se třemi vnitřními stavy odpovídá na vstupní posloupnost 00001000100010 výstupní posloupností 01010000101001. Najděte takový stroj. Zkuste určit, zda může existovat nějaký takový stroj se dvěma stavy.

(1.23) (Úloha o synchronizaci řady střelců.)

Úkolem je sestrojit tři typy sekvenčních strojů s vlastnostmi, které dále popíšeme. Tyto stroje budou srovnány do řady a propojeny navzájem tak, každý z nich je spojen pouze se svými sousedy. Všechny stroje v řadě jsou téhož typu (vojáci), s výjimkou strojů na obou koncích. Stroj na levém konci řady je druhého typu (generál), stroj na pravém konci řady je třetího typu. Stroje jsou synchronizovány, tzn. ke změnám stavu dochází u všech ve stejný okamžik. Následující stav každého stroje závisí pouze na jeho současném stavu a stavu jeho sousedů. Na začátku jsou všechny stroje v klidovém stavu. Úkolem je navrhnout stroje tak, aby

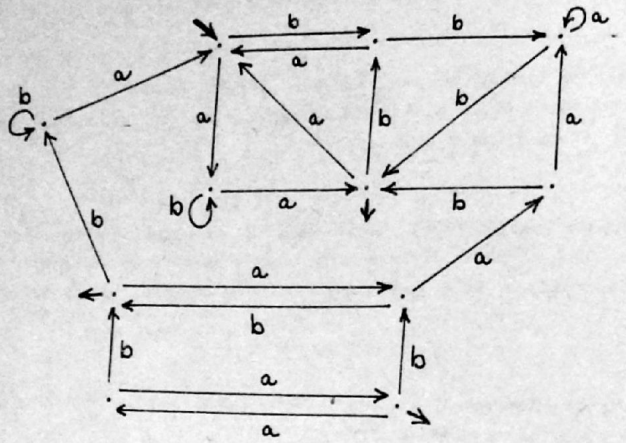
1) v prvním taktu generál změnil stav (na stav "vystřelte, až budete připraveni") a dále už svůj stav neměnil,

2) ostatní stroje měly jeden význačný stav ("střelba") různý od klidového stavu. Do tohoto stavu mají stroje vstoupit všechny zároveň v jediný okamžik. Před tím do něho nemá vstoupit žádný z nich.

Obtížnost úlohy spočívá v tom, že je třeba sestrojit stroje tak, aby fungovaly popsaným způsobem bez ohledu na to, jak dlouhý řetězec z nich sestavíme. Je tedy například vyloučeno, aby kterýkoliv stroj mohl "spočítat" délku řady.

Redukce automatů

1.24 Z následujícího diagramu automatu vyškrtněte nedosažitelné stavy.



Podobně jako konečné automaty lze redukovat i Mealyho sekvenční stroje. I zde postupně konstruujeme posloupnost ekvivalencí $\overset{1}{\sim}$, $\overset{2}{\sim}$, $\overset{3}{\sim}$, ... Přitom $p \overset{i}{\sim} q$, právě když pro každou vstupní posloupnost délky nejvýše i vydá automat stejnou výstupní posloupnost, ať začne činnost ve stavu p nebo q .

1.25 Zredukujte Mealyho sekvenční stroj

	0	1
q_0	$q_1/0$	$q_2/0$
q_1	$q_3/1$	$q_4/0$
q_2	$q_0/0$	$q_1/1$
q_3	$q_4/0$	$q_5/1$
q_4	$q_5/0$	$q_5/0$
q_5	$q_2/1$	$q_4/0$

Redukci lze přehledně provádět tak, že k tabulce stroje připisujeme zprava sloupce reprezentující jednotlivé ekvivalence $\overset{1}{\sim}$. Každá třída ekvivalence je vždy reprezentována nejmenším stavem (ve zvoleném uspořádání), který k ní náleží. Následující ekvivalence $\overset{i+1}{\sim}$ je vždy vytvořena z $\overset{i}{\sim}$ postupným zjemňováním na základě jednotlivých vstupních symbolů.

Naznačený postup podrobně rozpracujte a porovnejte s uvedeným řešením této úlohy.

Pomocí vyplněné tabulky zjistěte

a) Jak dlouhá posloupnost je potřeba pro rozlišení stavů q_1 , q_5 ?
Najděte takovou minimální posloupnost.

b) Jak dlouhá posloupnost je potřeba k rozlišení stavů q_0 , q_4 ?
Najděte takovou minimální posloupnost.

1.26 Zredukujte Mealyho stroj

	0	1
A	B/1	C/0
B	B/1	A/0
C	A/0	B/0
D	E/1	A/0
E	D/1	E/0
F	B/1	C/0

Pomocí vyplněné tabulky zjistěte

a) Jak dlouhá posloupnost je potřeba na rozlišení stavů D, E ?
Najděte takovou minimální posloupnost.

b) Jak dlouhá posloupnost je potřeba k rozlišení stavů B, D ?
Najděte takovou minimální posloupnost.

c) Jak dlouhá posloupnost je potřeba k rozlišení stavů A, F ?

Analogickým způsobem jako u Mealyho strojů lze provést redukci Mooreových strojů. U nich má smysl začít redukci (podobně jako u konečných automatů) konstrukcí ekvivalence $\overset{0}{\sim}$ dané značkovací funkcí stroje.

1.27 Pomocí následující tabulky redukce Mooreova stroje určete délky nejkratších posloupností, které rozliší dvojice stavů

a) q_2, q_6 c) q_2, q_3
b) q_0, q_6 d) q_1, q_5 . Sestrojte takové posloupnosti.

	0	1	výst.	0	0	1	0	1	0	1
q0	q2	q1	0	0	0	0	0	0	0	0
q1	q4	q2	0	0	0	0	1	1	1	1
q2	q0	q6	0	0	0	0	0	0	0	2
q3	q6	q5	0	0	0	3	3	3	3	3
q4	q5	q0	0	0	4	4	4	4	4	4
q5	q4	q3	1	5	5	5	5	5	5	5
q6	q1	q6	0	0	0	0	0	0	6	6

V příkladech 1.25 - 1.27 jsme tabulku redukce vytvářeli v jednotlivých fázích. Během každé fáze se vyplní jedna sada sloupců odpovídajících jednotlivým prvkům vstupní abecedy. Na konci i-té je tak zkonstruovaná ekvivalence $\overset{i}{\sim}$, která je používána při konstrukci sloupců tabulky v následující fázi. Takto vytvořená tabulka umožňuje okamžitě zjistit, jak dlouhé slovo je třeba k rozlišení dvou daných stavů apod.

Jde-li nám pouze o konstrukci reduktu, je možno použít úspornější tabulky, při které se při konstrukci každého sloupce tabulky vychází z rozkladu daného bezprostředně předcházejícím sloupcem a nikoliv sloupcem z konce předchozí fáze. Tímto způsobem se mohou rychleji rozlišit vzájemně neekvivalentní stavy, ekvivalence $\overset{i}{\sim}$ však v takové tabulce nemusí být zachyceny.

1.28

Sestrojte redukt Mooreova stroje (úspornější tabulkou).

	0	1	výst.
q_0	q_1	q_2	0
q_1	q_0	q_3	0
q_2	q_4	q_1	0
q_3	q_0	q_1	0
q_4	q_2	q_0	1
q_5	q_4	q_3	0

Nakreslete graf výchozího stroje i jeho reduktu.

1.29

Sestrojte redukt Mooreova stroje (úspornější tabulkou)

	0	1	výst.
q_0	q_1	q_2	1
q_1	q_3	q_0	0
q_2	q_4	q_5	0
q_3	q_0	q_2	0
q_4	q_2	q_5	0
q_5	q_0	q_3	0

1.30

Sestrojte tabulku redukce následujících dvou Mooreových strojů (úspornější verzi) a najděte všechny vzájemně ekvivalentní stavy.

A_1:

	0	1	výst.
q_0	q_9	q_1	0
q_1	q_1	q_2	1
q_2	q_3	q_1	1
q_3	q_2	q_4	0
q_4	q_2	q_3	0

A_2:

	0	1	výst.
q_5	q_5	q_6	0
q_6	q_7	q_5	0
q_7	q_7	q_9	1
q_8	q_9	q_8	0
q_9	q_8	q_7	1

1.31

Vytvořte tabulku redukce následujících dvou Mooreových strojů (úspornější verzi) a najděte všechny vzájemně ekvivalentní stavy.

A_1:

	0	1	výst.
q_0	q_0	q_1	1
q_1	q_2	q_3	1
q_2	q_3	q_4	0
q_3	q_1	q_0	0
q_4	q_3	q_2	0

A_2:

	0	1	výst.
q_5	q_6	q_7	0
q_6	q_8	q_5	1
q_7	q_9	q_6	1
q_8	q_5	q_8	0
q_9	q_7	q_6	0

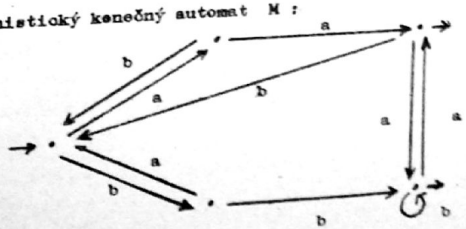
Nedeterministické konečné automaty

1.32

Sestrojte nedeterministický konečný automat, který přijímá právě všechna slova z abecedy $\{a, b, o\}$ začínající nebo končící řetězcem baac nebo obsahující na kterémkoli místě řetězec aboa .

1.33

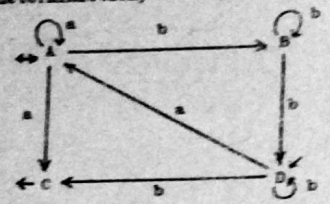
Mějme deterministický konečný automat M :



Sestrojte nedeterministický konečný automat, který rozpoznává zrcadlový obraz $(L(M))^R$ jazyka $L(M)$, tzn. jazyk tvořený právě všemi řetězci jazyka $L(M)$ psanými pozpátku.

(1.34)

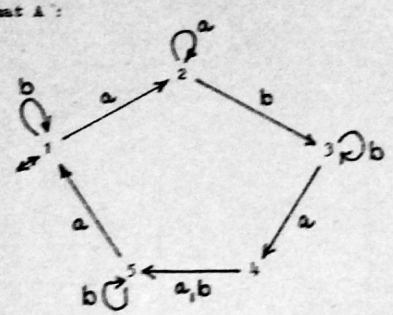
Mějme nedeterministický konečný automat M :



1) Nakreslete graf nedeterministického konečného automatu rozpoznávajícího jazyk $(L(M))^R$.
2) K automatu získanému v předchozím bodu sestrojte ekvivalentní redukovaný deterministický konečný automat.

(1.35)

Mějme automat A :



1) Sestrojte automat rozpoznávající jazyk
$L_1 = \{uv ; u, v \in \{a,b\}^+ \& (uav \in L(A) \text{ nebo } ubv \in L(A))\}$,
tzn. jazyk utvořený z řetězců jazyka $L(A)$, ze kterých "vypadne jedno písmeno".

2) Sestrojte automat rozpoznávající jazyk
$L_2 = \{uv ; u, v \in \{a,b\}^+ \& uav \in L(A)\}$,
tzn. jazyk utvořený z řetězců jazyka $L(A)$, ze kterých "vypadne právě jedno a".

3) Sestrojte automat rozpoznávající jazyk
$L_3 = \{uav ; uv \in L(A)\}$,
tzn. jazyk utvořený ze slov jazyka $L(A)$, do kterých byl "vložen jeden symbol a".

4) Jak by vypadaly automaty rozpoznávající jazyky $L_1 \cup L(A)$, $L_2 \cup L(A)$, $L_3 \cup L(A)$?

(1.36)

Navrhněte algoritmus, který umožní pro každý nedeterministický automat A rozhodnout, zda $L(A) \neq \emptyset$.

(1.37)

Navrhněte algoritmus, který umožní pro každý nedeterministický automat A nad abecedou X rozhodnout, zda $L(A) \neq X^*$.

Synchronizační posloupnosti

Jestliže v daném automatu A existuje stav q_f a existuje pro něj řetězec w vstupních symbolů, který má tu vlastnost, že automat A převede z libovolného stavu vždy do stavu q_f, pak řetězec w nazýváme synchronizační posloupností.

Nachází-li se několik exemplářů automatu A v různých (nám neznámých) stavech, potom lze všechny automaty převést do téhož stavu (synchronizovat) tak, že jim na vstup současně přivedeme tutéž synchronizační posloupnost.

Podobně lze k normální funkci vrátit zařízení, které se na základě nějaké vnější poruchy "dostalo z fáze".

Zjistit, zda daný automat má synchronizační posloupnost, lze jednoduchým užitím podmnožinové konstrukce. Z následujícího příkladu je vidět, že ke každému konečnému automatu A lze dokonce sestrojit konečný automat rozpoznávající právě všechny synchronizační posloupnosti automatu A .

(1.38)

Určete synchronizační posloupnosti Mooreova automatu M zadaného tabulkou:

	0	1	
A	B	C	0
B	B	D	1
C	A	E	0
D	C	E	0
E	C	C	1

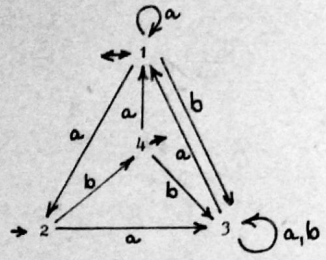
Sestrojte konečný automat reprezentující zmíněnou množinu posloupností.

(1.39)

Nechť A je konečný automat s množinou stavů Q, vstupní abecedou Σ a přechodovou funkcí δ. Sestrojte konečný automat přijímající právě všechny synchronizační posloupnosti automatu A .

Implementace nedeterministického automatu

1.40 Využijte principu podmnožinové konstrukce k návrhu implementace nedeterministického konečného automatu:



v jazyce Pascal.

1.41 V jazyce Pascal implementujte automat $A_n = (Q_n, X, f_n, I_n, F_n)$, kde jednotlivé parametry jsou definovány takto:

$Q_n = \{0,1,2,\ldots,n-1\}$,

$X = \{a, b\}$,

$I_n = F_n = \{0\}$,

$f_n(i,a) = \{i+1\}$ pro všechna i, $0 \leq i < n-1$,

$f_n(n-1,a) = \{0\}$,

$f_n(i,b) = \{0,i\}$ pro všechna i, $0 < i \leq n-1$,

$f_n(0,b) = \emptyset$.

Zvolte např. $n = 200$.

1.42 (Identifikace klíčových slov.) Sestrojte Mooreův automat se vstupní abecedou $\{a, b\}$, který v každém okamž ku určuje ta z klíčových slov

A = baa,
B = aa,
C = aba,

kterými končí právě přečtený úsek slova, tzn. vydává vždy jistý symbol s, kde $s \subseteq \{A, B, C\}$.

Navrhněte implementaci tohoto automatu.

(Návod. Začněte návrhem nedeterministického automatu, z něhož získáte Mooreův automat modifikací podmnožinové konstrukce. Srovnejte implementaci na základě nedeterministického automatu s implementací na základě výsledného Mooreova automatu.)

Uzávěrové vlastnosti

1.43 Nechť L_1, L_2, L_3, L jsou libovolné jazyky v abecedě Σ, h libovolný homomorfismus, w libovolný řetězec a δ libovolná substituce. Dokažte následující vztahy:

a) $L_1(L_2 \cup L_3) = L_1L_2 \cup L_1L_3$
b) $L_1(L_2 \cap L_3) = L_1L_2 \cap L_1L_3$
c) $(L^*)^* = L^*$
d) $(L_1 \cup L_2)^* = L_1^*(L_2L_1^*)^*$
e) $h(L_1 \cup L_2) = h(L_1) \cup h(L_2)$
f) $h(L_1 . L_2) = h(L_1) . h(L_2)$
g) $\partial_w(L_1 \cup L_2) = \partial_w(L_1) \cup \partial_w(L_2)$
h) $\partial_w(\Sigma^* - L) = \Sigma^* - \partial_w L$
i) $(L_1 . L_2)^R = L_2^R . L_1^R$
j) $\delta(L_1L_2) = \delta(L_1) . \delta(L_2)$.

1.44 Nechť L, L_1, L_2 jsou libovolné jazyky a h libovolný homomorfismus. Dokažte, že následující vztahy obecně nemusí platit.

a) $L_1L_2 = L_2L_1$
b) $(L_1 \cap L_2)^* = L_1^* \cap L_2^*$
c) $(L_1 \cup L_2)^* = L_1^* \cup L_2^*$
d) $h(L_1 \cap L_2) = h(L_1) \cap h(L_2)$
e) $h(h(L)) = h(L)$
f) $(L_1L_2)^*L_1 = L_1^*(L_2L_1)^*$
g) $L_2(L_2 \setminus L_1) = L_1$

1.45 Dokažte, že pro libovolné jazyky L, L_1, L_2 v abecedě Σ platí:

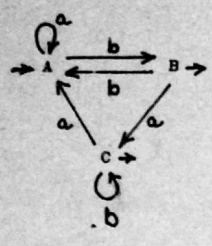
a) $L_1, L_2 \in \mathcal{F} \Rightarrow L_1 - L_2 \in \mathcal{F}$.
b) $L \in \mathcal{F} \Rightarrow L^+ \in \mathcal{F}$.
c) $L_1 \in \mathcal{F}$ a L_2 se liší od L_1 v nejvýše konečně mnoha slovech $\Rightarrow L_2 \in \mathcal{F}$.
d) Jestliže h je homomorfismus, potom $L \in \mathcal{F} \Rightarrow h^{-1}(L) \in \mathcal{F}$.

1.46 Využijte uzávěrových vlastností k návrhu algoritmu rozhodujícího pro každé dva konečné automaty A_1, A_2, zda jazyky $L(A_1), L(A_2)$ jsou nesrovnatelné inkluzí.

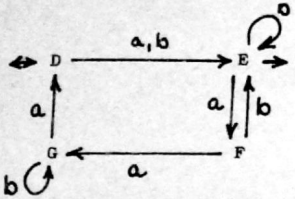
1.47

Nějme dva deterministické konečné automaty

M1:



M2:



Nakreslete grafy automatů rozpoznávajících jazyky

a) $L(M_1) \cap L(M_2)$,

b) $L(M_1) \cup L(M_2)$,

c) $L(M_1) - L(M_2)$.

1.48

Sestrojte automat rozpoznávající jazyk L v abecedě {0,1} tvořený slovy, která obsahují sudý počet bloků jedniček (tzn. maximálních souvislých úseků jedniček) a každý takovýto úsek je liché délky. (Např. 000,01011100,1110100101 jsou řetězce patřící do L, zatímco 01110, 0100110 do L nepatří .)

1.49

Sestrojte redukovaný konečný automat (v normálním tvaru) rozpoznávající jazyk $\{a,ab,ba\}^*$.

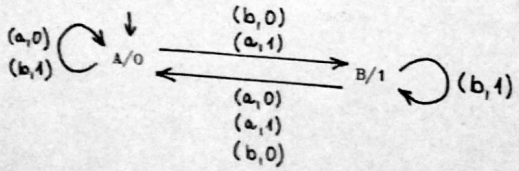
1.50

Sestrojte nejmenší Mealyho automat, který pro dvě binární čísla x,y počítá binární číslo 3(x+y) . (Návod: využijte automatů z př. 1.17,1.21 .)

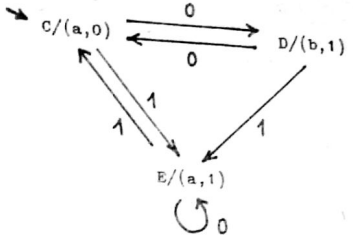
1.51

Nějme dvě abecedy X = {a,b} , Y = {0,1} a dva iniciální Mooreovy automaty

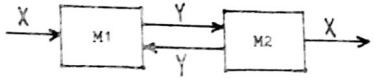
M1:



M2:



Předpokládejme, že automaty jsou propojeny podle následujícího obrázku



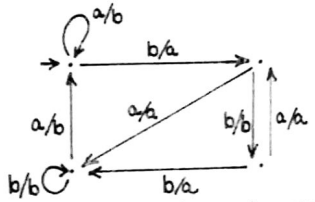
a začínají oba činnost ve svých počátečních stavech.

1) Popište zobrazení $X^* \to X^*$, které tento systém realizuje.

2) Existuje pro uvedený systém synchronizační posloupnost?

1.52

Mějme iniciální Mealyho automat A daný přechodovým grafem



Předpokládáme, že automat zahajuje činnost vždy v počátečním stavu.

Proberme několik možných typů pozorování činnosti automatu A .

1) Každý výpočet automatu A lze popsat jako posloupnost dvojic

(*) $(x_1,y_1)(x_2,y_2)\ldots(x_n,y_n)$,

kde xi (resp. yi) je vstupní (resp. výstupní) symbol automatu v i-tém kroku výpočtu.

Sestrojte konečný automat, který přijímá právě všechny posloupnosti tvaru (*).

2) Obdobně jako v předchozím případě, pouze vstupní i výstupní signály jsou pozorovány jednotlivě, nikoli ve dvojicích. Přitom vstup vždy předchází příslušný výstup a každý výstup předchází vstup v následujícím kroku, tzn. je pozorována posloupnost tvaru

(**) $x_1\ y_1\ x_2\ y_2\ \ldots\ x_n\ y_n$,

kde xi (resp. yi) je vstupní (resp. výstupní) symbol v i-tém kroku.

Sestrojte (parciální) konečný automat, který přijímá právě všechny posloupnosti tvaru (**).

3) Je pozorován pouze výstup automatu A. Tak jsou získány posloupnosti tvaru

$$y1\ y2\ \dots\ yn\ .$$

Sestrojte konečný automat přijímající právě takovéto výstupní posloupnosti automatu A.

4) Podobně jako v případě 2 s tím rozdílem, že nelze rozlišit pořadí vstupního a výstupního signálu v jednotlivých okamžicích, tzn. pozorované posloupnosti jsou tvaru

$$x1\ y1\ x2\ y2\ \dots\ xn\ yn\ ,$$

kde pro každé $1 \leq i \leq n$ je buď xi vstupní a yi výstupní nebo xi výstupní a yi vstupní

symbol v i-tém okamžiku.
Sestrojte konečný automat, který přijímá právě takovéto posloupnosti.

5) Podobně jako v předchozím případě s tím rozdílem, že libovolný vstupní nebo výstupní signál může být nečitelný, tzn. místo libovolného (třeba i všech) signálu xi nebo yi se může objevit signál n.
a) Sestrojte automat přijímající právě takovéto posloupnosti.
b) Může mít automat pro případ 5 méně stavů než pro případ 4?

(1.53)

Nechť $M_1,\dots,M_n$ jsou sekvenční stroje. Tvoří množina všech posloupností, které jsou synchronizačními posloupnostmi pro každé A_i, jazyk rozpoznatelný konečným automatem ?

(1.54)

Nechť Σ_1, Σ_2 jsou dvě konečné abecedy a $h_1: (\Sigma_1 \cup \Sigma_2)^* \longrightarrow \Sigma_1^*$, $h_2: (\Sigma_1 \cup \Sigma_2)^* \longrightarrow \Sigma_2^*$ definované takto : pro libovolné $x \in \Sigma_1 \cup \Sigma_2$ je

$$h_1(x) = \begin{cases} x, & \text{jestliže } x \in \Sigma_1 \\ \varepsilon, & \text{jesliže } x \notin \Sigma_1 \end{cases}, \quad h_2(x) = \begin{cases} x, & \text{jestliže } x \in \Sigma_2 \\ \varepsilon, & \text{jestliže } x \notin \Sigma_2 . \end{cases}$$

Pro každé dva jazyky $L_1 \subseteq \Sigma_1^*$, $L_2 \subseteq \Sigma_2^*$ definujeme <u>střídavý produkt jazyků L_1, L_2</u> jako jazyk $L_1 \amalg L_2 = \{w \in (\Sigma_1 \cup \Sigma_2)^* ; h_1(w) \in L_1 \,\&\, h_2(w) \in L_2\}$.
Dokažte, že pro každé dva jazyky L_1, L_2 rozpoznatelné konečnými automaty je i jejich střídavý produkt rozpoznatelný konečným automatem.

(1.55)

Navrhněte takový jazyk L rozpoznatelný konečným automatem, aby platilo

$$L = L \setminus L = L / L .$$

(1.56)

Zjistěte, zda existuje netriviální sériová a paralelní dekompozice Mealyho stroje $A = (Q, \Sigma, \Delta, \delta, \lambda)$ daného tabulkou

	a	b	a	b
1	4	2	C	D
2	3	1	D	A
3	2	4	B	C
4	1	3	A	B

$Q = \{1,2,3,4\}$
$\Sigma = \{a,b\}$
$\Delta = \{A,B,C,D\}$
$\delta: Q \times \Sigma \longrightarrow Q$
$\lambda: Q \times \Sigma \longrightarrow \Delta$

a případné tytodekompozice sestrojte.

(1.57)

Pro následující sekvenční stroje zadané tabulkami přechodových funkcí zjistěte, zda mají netriviální sériovou či paralelní dekompozici.

1)

δ	a	b
1	4	2
2	4	3
3	1	2
4	1	3

$Q = \{1,2,3,4\}$
$\Sigma = \{a,b\}$

2)

δ	a	b	c
1	3	4	2
2	5	1	3
3	2	6	5
4	6	1	6
5	1	6	1
6	4	3	2

$Q = \{1,2,3,4,5,6\}$
$\Sigma = \{a,b,c\}$

3)

δ	a	b	c	d	e
1	2	3	6	7	1
2	1	4	5	8	1
3	4	1	8	7	3
4	3	2	7	8	3
5	6	7	2	3	2
6	5	8	1	4	2
7	8	5	4	3	4
8	7	6	3	4	4

$Q = \{1,\dots,8\}$
$\Sigma = \{a,\dots,e\}$

Regulární jazyky, regulární rovnice

1.58 Popište pomocí regulárních výrazů následující jazyky v abecedě $\{a,b\}$.

a) Jazyk všech slov obsahujících podslovo aaba.

b) Jazyk všech slov začínajících předponou abb nebo končících příponou bbaa.

c) Jazyk všech slov obsahujících počet výskytů symbolu a dělitelný třemi.

d) Jazyk těch slov, která začínají i končí stejnou dvojicí symbolů.

e) Jazyk právě těch slov, která neobsahují podslovo aa.

1.59 Rozhodněte, zda platí:

a) $(011 + (10)^*1 + 0)^* \equiv 011(011 + (10)^*1 + 0)^*$

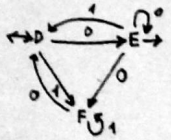
b) $((1 + 0)^*100(1 + 0)^*)^* \equiv ((1 + 0)100(1 + 0)^*100)^*$

1.60 Sestrojte zobecněný nedeterministický automat rozpoznávající jazyk určený regulárním výrazem $((01^*0 + 101)^*100 + (11)^*0)^*01$.

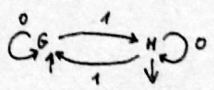
1.61 Následující tři stavové diagramy nedeterministických automatů popisují regulární jazyky L_1, L_2, L_3.

L_1:

L_2:



L_3:



Definujme regulární substituci δ předpisem $\delta(a) = L_2$, $\delta(b) = L_3$. Najděte regulární výraz popisující jazyk $\delta(L_1)$.

Meze použitelnosti konečných automatů

1.62 Odhadněte počet stavů redukovaného konečného automatu, který rozpoznává jazyk $L = \{w \in \{a,b\}^*,\ |w| = 6$ a w je symetrické slovo, tzn. $w = w^R\}$.

1.63 Budiž $n \geq 1$ přirozené číslo. Odhadněte počet stavů redukovaného konečného automatu, který rozpoznává jazyk

$L = \{w \in \{a,b\}^*,\ |w| = 2n$ a w je symetrické slovo$\}$.

1.64 Odhadněte počet stavů nedeterministického konečného automatu, který rozpoznává jazyk $L = \{w \in \{a,b\}^*,\ |w| = 6$ a w je symetrické slovo$\}$.

1.65 Budiž $n \geq 1$ přirozené číslo. Odhadněte počet stavů nedeterministického konečného automatu, který rozpoznává jazyk

$L = \{w \in \{a,b\}^*,\ |w| = 2n$ a w je symetrické slovo$\}$.

1.66 Nechť $n \geq 1$ je přirozené číslo. Odhadněte počet stavů redukovaného konečného automatu rozpoznávajícího jazyk daný regulárním výrazem $(a + b)^*a(a + b)^{n-1}$.

1.67 Nechť $n \geq 1$ je přirozené číslo. Odhadněte počet stavů nedeterministického konečného automatu rozpoznávajícího jazyk daný regulárním výrazem $(a + b)^*a(a + b)^{n-1}$.

1.68 Dokažte, že jazyk $L = \{w \in \{a,b\}^*;$ w je symetrické slovo$\}$ není rozpoznatelný konečným automatem.

1.69 Dokažte, že jazyk $L = \{w \in \{0,1\}^*;$ w obsahuje stejný počet nul jako jedniček$\}$ není regulární.

1.70 Dokažte, že jazyk $L = RV(\{0,1\})$ všech regulárních výrazů nad abecedou $\{0,1\}$ není regulární jazyk.

2. POJEM GRAMATIKY, CHOMSKÉHO HIERARCHIE

Konstrukce gramatik

2.1 Dokažte, že gramatika $G = (\{S,B,C\},\{a,b,c\},S,P)$, kde P je množina pravidel:

$S \to aSBC$ $\quad bB \to bb$

$S \to abC$ $\quad bC \to bc$

$CB \to BC$ $\quad cC \to cc$

generuje jazyk $L = \{a^n b^n c^n ; n \geq 1\}$.

2.2 Navrhněte gramatiky, které generují tyto jazyky:

a) $L_1 = \{u \in \{a,b\}^* ;$ počet výskytů symbolu b v u je dělitelný třemi$\}$

b) $L_2 = \{u \in \{a,b\}^* ;$ u obsahuje podslovo baab$\}$

c) $L_3 = \{0^n 1^m 0^n ; n,m \geq 0\}$

d) $L_4 = \{w(w)^R ; w \in \{0,1\}^*\}$

e) $L_5 = \{ww ; w \in \{0,1\}^*\}$

f) $L_6 = \{a^{2^n} ; n \geq 1\}$

2.3 Popište jazyk L generovaný gramatikou (S je jediný neterminál)

$S \to bSS$

$S \to a$.

2.4 Ukažte, že gramatika G:

$S \to SS \mid aSb \mid bSa \mid \varepsilon$

generuje právě všechna slova obsahující stejný počet symbolů a a b.

2.5 K bezkontextové gramatice G:

$S \to AB \mid \varepsilon$

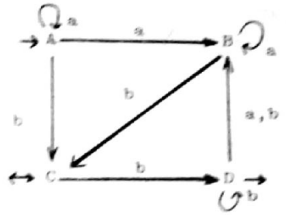
$A \to aAAb \mid BS \mid CA \mid \varepsilon$

$B \to BbA \mid CaC \mid \varepsilon$

$C \to aBB \mid bS$

sestrojte nevypouštějící bezkontextovou gramatiku G' takovou, že $L(G') = L(G) - \{\varepsilon\}$.

2.6 K nedeterministickému automatu



sestrojte s ním ekvivalentní regulární gramatiku.

2.7 K regulární gramatice

$S \to abS \mid bbaA \mid \varepsilon$

$A \to abA \mid bB$

$B \to acS \mid bC \mid \varepsilon$

$C \to aC \mid bA$

sestrojte s ní ekvivalentní nedeterministický konečný automat.

2.8 Sestrojte regulární gramatiky generující jazyky popsané následujícími regulárními výrazy.

a) $01110\ (01)^*\ (111)^*$

b) $(01^* + 101\ )^* 0^* 1$

c) $((01 + 101)^* + (1 + 00)^*)^*\ 01^* 0$

d) $(a+bc)\ (aa^* + (ab)^* c + a)^*$

e) $(((ab+c)^*\ a\ (bc)^* + b)^* (ab^* + c)^*)^*$

2.9 Popište, jak lze od regulárních gramatik generujících jazyky L_1, L_2 přejít k regulárním gramatikám generujícím jazyky

a) $L_1 . L_2$, b) $L_1 \cup L_2$, c) L_1^* .

2.10 Sestrojte bezkontextové gramatiky generující jazyky

a) $\{0^n 1 0^n ; n \geq 0\}$,

b) $\{0^n 1^m 0^n ; n,m \geq 1\}$,

c) $\{ww^R ; w \in \{0,1\}^*\}$,

d) $\{0^n 1^m ; 0 \leq n \leq m\}$,

e) $\{0^n 1^m ; 1 \leq n \leq m \leq 2n\}$,

f) $RV(\{0,1\})$ (množina regulárních výrazů nad abecedou $\{0,1\}$).

2.11 Gramatiku $G = (\Pi, \Sigma, S, P)$ nazýváme separovanou, jestliže pro každé pravidlo $\alpha \to \beta \in P$ platí: buď $\alpha, \beta \in \Pi^+$

nebo $\alpha \in \Pi$ a $\beta \in \Sigma \cup \{\varepsilon\}$.

Ukažte, že ke každé gramatice existuje s ní ekvivalentní separovaná gramatika.

2.12 Gramatika $G = (N, \Sigma, S, P)$ typu 0 se nazývá monotónní, jestliže pro každé pravidlo $\alpha \to \beta \in P$ platí, že $|\alpha| \leq |\beta|$. Ukažte, že ke každé monotónní gramatice je možno sestrojit s ní ekvivalentní kontextovou gramatiku.

2.13 (Řešení vyžaduje i znalost lemmatu o vkládání procvičovaného v kapitole 3.)
Zařaďte do Chomského hierarchie jazyky

a) $L_1 = \{w \in \{a,b\}^* ;$ w obsahuje maximálně tři výskyty symbolu a$\}$

b) $L_2 = \{a^i b^j ; 1 \leq i \leq j \leq 2i\}$

c) L_3 generovaný gramatikou $S \longrightarrow S + S \mid a$

d) $L_4 = \{a^i o b^{2^i} ; i \geq 1\}$

e) L_5 generovaný gramatikou

$S \longrightarrow ASB \mid \varepsilon$

$AB \longrightarrow BA$

$A \longrightarrow a$

$B \longrightarrow b$

f) $L_6 = \{0^i 1^j ; i \equiv j \pmod 3\}$

g) $L_7 = \{a^i b^j ; i-j \equiv 1 \pmod 3\}$.

3. BEZKONTEXTOVÉ JAZYKY

Redukce, derivační stromy, víceznačnost

3.1 Zredukujte následující bezkontextové gramatiky:

1) $S \longrightarrow aSb \mid aAbb \mid \varepsilon$
$A \longrightarrow aAB \mid bB$
$B \longrightarrow aAb \mid BB$
$C \longrightarrow CC \mid cS$

(Jako obvykle velká písmena označují neterminály, malé terminály, S je počáteční symbol.)

2) $S \longrightarrow aA \mid bB \mid aSa \mid bSb \mid \varepsilon$
$A \longrightarrow bCD \mid Dba$
$B \longrightarrow Bb \mid AC$
$C \longrightarrow aA \mid c$
$D \longrightarrow DE$
$E \longrightarrow \varepsilon$

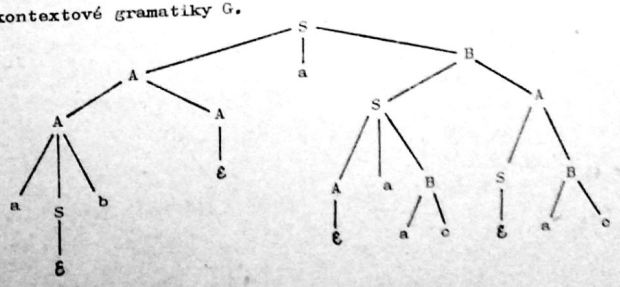
3) $S \longrightarrow A \mid B \mid 0SB \mid \varepsilon$
$A \longrightarrow 0A \mid C$
$B \longrightarrow 1B \mid 1$
$C \longrightarrow ASS$

3.2 Mějme gramatiku G danou pravidly

$S \longrightarrow aS \mid AB \mid CD$
$A \longrightarrow aDb \mid AD \mid BC$
$B \longrightarrow bSb \mid BB$
$C \longrightarrow BA \mid ASb$
$D \longrightarrow ABCD \mid \varepsilon$.

Zjistěte, zda $L(G) \neq \phi$.

3.3 Je dán derivační strom popisující strukturu řetězce w = abaaacac podle jisté bezkontextové gramatiky G.



1) Napište levé odvození řetězce w podle gramatiky G.
2) Napište pravé odvození řetězce w podle G.
3) Najděte rozklad $w = w_1w_2w_3$ tak, aby řetězec $w_1w_2^2w_3$ opět patřil do L(G).
4) Zkuste zjistit , zda G je víceznačná gramatika.

3.4
Gramatika $E \longrightarrow E+E \mid E*E \mid (E) \mid a$ generující aritmetické výrazy typu $(a+a+a)*a$, $a+a*a$, atd., je víceznačná. Zkuste popsat typy víceznačnosti, které se v ní vyskytují a navrhnout úpravy gramatiky, které víceznačnost odstraní. Jazyk generovaný gramatikou se přitom nesmí změnit.

3.5
Dokažte, že každý regulární jazyk je jednoznačný.

Zásobníkové automaty

3.6
Sestrojte nedeterministický zásobníkový automat, který prázdným zásobníkem rozpoznává jazyk generovaný gramatikou

$E \longrightarrow E+T \mid T$
$T \longrightarrow T*F \mid F$
$F \longrightarrow (E) \mid a$.

3.7
Sestrojte deterministický zásobníkový automat, který koncovým stavem rozpoznává jazyk generovaný gramatikou $S \longrightarrow SS \mid [S] \mid \varepsilon$.

3.8
Sestrojte deterministický zásobníkový automat rozpoznávající jazyk $L = \{w \in \{0,1\}^*;$ w obsahuje stejný počet nul a jedniček$\}$ koncovým stavem.

3.9
Sestrojte nedeterministický zásobníkový automat, který rozpoznává jazyk $L = \{uov ; u,v \in \{a,b\}^* \wedge u \neq v\}$.

3.10
K zásobníkovému automatu M se vstupní abecedou $\{a,b\}$, zásobníkovou abecedou $\{A,B\}$, počátečním zásobníkovým symbolem A, množinou stavů $\{q_0,q_1,q_2\}$, počátečním stavem q_0 a přechodovou funkcí definovanou takto :

$\delta(q_0,a,A) = \{(q_1,AA),(q_0,B)\}$,
$\delta(q_1,b,A) = \{(q_1,AA)\}$,
$\delta(q_0,\ell,B) = \{(q_1,A)\}$,
$\delta(q_1,\ell,A) = \{(q_2,\ell)\}$,
$\delta(q_2,a,A) = \{(q_2,A)\}$,
$\delta(q_2,b,A) = \{(q_2,\ell)\}$,

sestrojte gramatiku generující jazyk N(M) a tuto gramatiku zredukujte.

Chomského normální forma

3.11
Převeďte do Chomského normální formy gramatiku G danou následujícími pravidly (terminály jsou 0 a 1):

$S \longrightarrow A \mid 0SA \mid \varepsilon$
$A \longrightarrow 1A \mid 1 \mid B1$
$B \longrightarrow 0B \mid 0 \mid \varepsilon$.

3.12
Převeďte do Chomského normální formy gramatiky

1) $S \longrightarrow 0A10B11$
$A \longrightarrow 0A1 \mid \varepsilon$
$B \longrightarrow 0B11 \mid \varepsilon$,

2) $S \longrightarrow (E)$
$E \longrightarrow F+F \mid F*F$
$F \longrightarrow a \mid S$,

3) $S \longrightarrow abS \mid CaS \mid BaS \mid a$
$B \longrightarrow aCB \mid SC$
$C \longrightarrow BCb \mid SB$.

Odstranění levé rekurze

3.13
V gramatice

$A \longrightarrow A+B \mid B$
$B \longrightarrow B*C \mid C$
$C \longrightarrow (A) \mid a$

odstraňte levě rekurzivní pravidla, tzn pravidla tvaru $X \longrightarrow X\alpha$, kde X je neterminál a α neprázdný řetězec.

3.14
Nechť $G = (\Pi, \Sigma, S, P)$ je bezkontextová gramatika. Nechť $A \longrightarrow A\alpha_1, \ldots, A \longrightarrow A\alpha_k$ jsou všechna její A-pravidla, která mají na pravé straně A jako nejlevější symbol.
Nechť $A \longrightarrow \beta_1, \ldots, A \longrightarrow \beta_m$ jsou všechna ostatní A-pravidla gramatiky G.
Zkonstruujme na základě G gramatiku $G' = (\Pi \cup \{Z\}, \Sigma, S, P')$ tak, že přidáme nový neterminál Z a nahradíme všechna A-pravidla z G soustavou pravidel:

(1) $A \longrightarrow \beta_i$; $A \longrightarrow \beta_i Z$ pro $1 \leq i \leq m$

(2) $Z \longrightarrow \alpha_i$; $Z \longrightarrow \alpha_i Z$ pro $1 \leq i \leq r$.

Dokažte, že L(G) = L(G').

3.15 Využijte postupu z cvičení 3.14 a převeďte gramatiku

$A \longrightarrow BA$ $\quad C \longrightarrow AC$

$A \longrightarrow BC$ $\quad B \longrightarrow 0$

$B \longrightarrow AB$ $\quad C \longrightarrow 1$

do Greibachové normální formy.

Lemma o vkládání

3.16 Dokažte, že následující jazyky nejsou bezkontextové.

a) $L_1 = \{0^{i_1} j 0^i \; ; \; 0 \leq j \leq i\}$

b) $L_2 = \{0^{i_1} j 0^{i_1} j \; ; \; i,j \geq 0\}$

c) $L_3 = \{a^i b^j c^k \; ; \; 0 \leq i \leq j \leq k\}$

d) $L_4 = \{a^n \; ; \; n \geq 1\}$

3.17 Najděte algoritmus, který pro každý bezkontextový jazyk rozhodne, zda je konečný.

3.18 Dokažte toto lemma o vkládání pro lineární jazyky:
ke každému lineárnímu jazyku L existuje přirozené číslo p takové, že každý řetězec z L délky alespoň p lze psát ve tvaru z = uvwxy,
kde $|uvxy| \leq p$, aspoň jeden z řetězců v,x je neprázdný a pro každé $i \geq 0$ je $uv^i wx^i y \in L$.

3.19 Pomocí lemmatu ze cvičení 3.18 dokažte, že jazyk $\{0^{i_1} i 0^{j_1} j \; ; \; i,j \geq 0\}$ není lineární.

3.20 Dokažte toto zobecnění lemmatu o vkládání:
pro každý bezkontextový jazyk L existují přirozená čísla m,n s následujícími vlastnostmi:
Nechť $z = a_1 \ldots a_k \in L$ a $1 \leq i_1 < i_2 < \ldots < i_l \leq k$ (tzn. ve slově z označíme l písmen). Jestliže $l > n$, potom slovo z může být napsáno ve tvaru

z = uvwxy

tak, že

1) $uv^i wx^i y \in L$ pro všechna $i \geq 0$,
2) alespoň jedno ze slov v,x obsahuje označené písmeno (tzn. $v = a_p \ldots a_q$, $x = a_r \ldots a_s$ a pro některé i_j je buď $p \leq i_j \leq q$ nebo $r \leq i_j \leq q$) ,
3) ve slově vwx se vyskytuje nejvýše m označených písmen.

3.21 Dokažte, že jazyk $L = \{a^i b^j c^k \; ; \; i \neq j \;\&\; j \neq k \;\&\; i \neq k\}$ není bezkontextový.

Uzávěrové vlastnosti

3.22 Dokažte, že třída bezkontextových jazyků je uzavřena vůči operaci

a) sjednocení,
b) zřetězení,
c) iterace,
d) zrcadlového obrazu,
e) substituce,
f) homomorfismu.

3.23 Dokažte, že třída bezkontextových jazyků tvoří plnou abstraktní třídu jazyků.

3.24 Dokažte, že plná abstraktní třída jazyků je uzavřena vůči:

a) zřetězení jazyků,
b) operaci iterace,
c) regulární substituci,
d) sekvenčnímu zobrazení,
e) inverznímu sekvenčnímu zobrazení,
f) levému i pravému kvocientu s regulárním jazykem.

3.25 Ukažte, že třída bezkontextových jazyků není uzavřena vůči operaci střídavého produktu definovaného v 1.54 .

LL(1) - analýza

4.1 Mějme LL(1)-analyzátor

	a	(	)	+	*	e
A	BA´;1	BA;1				ε;3
A´			ε;3	+BA´;2		
B	CB´;4	CB´;4				
B´			ε;6	ε;6	*CB´;5	ε;6
C	a;8	(A);7				
a	krátit					
(		krátit				
)			krátit			
+				krátit		
*					krátit	
e						přijmout

pro gramatiku

(1) A ⟶ BA´
(2) A´⟶ +BA´
(3) A´⟶ ε
(4) B ⟶ CB´
(5) B´⟶ *CB´
(6) B´⟶ ε
(7) C ⟶ (A)
(8) C ⟶ a

Projděte výpočet tohoto analyzátoru nad
a) řetězcem a*(a+a),
b) řetězcem a(a+a),
c) řetězcem a+a .

4.2 K následující jednoduché LL(1)-gramatice sestrojte LL(1)-analyzátor.

(1) S ⟶ if E then S
(2) S ⟶ while E do S
(3) S ⟶ repeat S until E
(4) S ⟶ s
(5) E ⟶ c
(6) E ⟶ non c

Neterminály této gramatiky jsou S , E , terminály jsou if , then , while, do , non , repeat , until , s , c .

Připomeňme několik důležitých pojmů užívaných v teorii LL(1)-gramatik. Předpokládáme bezkontextovou gramatiku $G = (\Pi, \Sigma, S, P)$.

1) Pro libovolný řetězec $\alpha \in (\Pi \cup \Sigma)^*$
$FIRST(\alpha) = \{u \in \Sigma^*$, buď $|u| \leq 1$ a $\alpha \Rightarrow^* u$,
nebo $|u| = 1$ a $\alpha \Rightarrow^* uv$ pro nějaké $v \in \Sigma^+\}$.

2) Pro libovolný neterminál A
$FOLLOW(A) = \{u$; $u \in FIRST(\alpha)$ pro nějakou levou větnou formu $vA\alpha\}$.

3) Pro libovolné pravidlo $X \longrightarrow \alpha \in P$ definujeme řídicí množinu pravidla $X \longrightarrow \alpha$ takto:
$D(X \longrightarrow \alpha) = FIRST(\alpha\, FOLLOW(X))$.

4) Gramatika G je (silná) LL(1)-gramatika, jestliže pro každá její dvě pravidla
$X \longrightarrow \alpha$
$X \longrightarrow \beta$
taková, že $\alpha \neq \beta$ platí
$D(X \longrightarrow \alpha) \cap D(X \longrightarrow \beta) = \emptyset$.

4.3 Sestrojte LL(1)-analyzátor pro gramatiku

(1) S ⟶ if c then S
(2) S ⟶ repeat SS´until c
(3) S ⟶ begin SS´end
(4) S ⟶ s
(5) S´⟶ ;SS´
(6) S´⟶ ε .

4.4 Sestrojte (pokud existuje) LL(1)-analyzátor pro gramatiku

(1) S ⟶ aAb
(2) S ⟶ BAb
(3) A ⟶ aS
(4) A ⟶ ε
(5) B ⟶ bAb .

4.5

Vysvětlete, proč žádná redukovaná bezkontextová gramatika obsahující odvozovací pravidlo s levou rekurzí, tzn. pravidlo tvaru

$X \longrightarrow X\alpha$,

kde X je neterminál a α libovolný řetězec, není LL(1).

Proč nemůže být LL(1) ani gramatika, ve které je možné odvození tvaru

$X \Rightarrow^{+} X\alpha$?

4.6

Zjistěte, zda gramatika

(1) S ⟶ BA
(2) A ⟶ BS
(3) A ⟶ d
(4) B ⟶ aA
(5) B ⟶ bS
(6) B ⟶ c

je LL(1). Jestliže ano, sestrojte LL(1)-analyzátor.

Popišme postup, kterým pro libovolnou bezkontextovou gramatiku $G = (\mathbb{N}, \Sigma, S, P)$ zkonstruujeme funkci FOLLOW.

1. Sestrojme funkci FIRST.
2. Sestrojme graf relace $\rhd$ na množině $\mathbb{N}$ definované takto:
   pro libovolné neterminály $A \neq B$ je
   $A \rhd B \Leftrightarrow_{df}$ existuje pravidlo $A \to \alpha B \beta \in P$ takové, že $\beta \Rightarrow^{*} \varepsilon$.
   (Podmínka $\beta \Rightarrow^{*} \varepsilon$ se snadno ověří pomocí funkce FIRST, protože $\beta \Rightarrow^{*} \varepsilon \Leftrightarrow \varepsilon \in FIRST(\beta)$.)
3. Definujme reflexivní a tranzitivní uzávěr $\rhd^{*}$ relace $\rhd$.
   Na $\mathbb{N}$ definujme ekvivalenci $\sim$ předpisem
   $A \sim B \Leftrightarrow_{df} A \rhd^{*} B$ a zároveň $B \rhd^{*} A$.
4. Sestrojme částečné uspořádání $<$ množiny $\mathbb{N}/\sim$ indukované operací $\rhd^{*}$: pro $[A] \neq [B]$ je
   $[A] < [B]$ právě když $A \rhd^{*} B$.
5. Pro každý neterminál $A \in \mathbb{N} - \{S\}$ sestrojíme množinu
   $FOL(A) =_{df} \{x \in \Sigma$; existuje $X \to \alpha A \beta \in P$ takové, že $x \in FIRST(\beta)\}$.
   Pro počáteční neterminál S
   $FOL(S) =_{df} [\{x \in \Sigma$; existuje $X \to \alpha S \beta \in P$ takové, že $x \in FIRST(\beta)] \cup \{\varepsilon\}$.
6. Za pomoci relace $<$ a ekvivalence $\sim$ konstruujeme množiny FOLLOW podle předpisu:

$$FOLLOW(X) = \bigcup_{Y \rhd X} FOL(Y)$$

4.7

Ověřte korektnost uvedené konstrukce funkce FOLLOW.

4.8

Zkonstruujte funkce FIRST a FOLLOW pro gramatiku

A ⟶ AC | DaED
B ⟶ Cb | EcE | ε
C ⟶ BaED | CA
D ⟶ p | ED | ε
E ⟶ BpAaB | qAr

4.9

Pro gramatiku

(1) S ⟶ aABbCD
(2) S ⟶ ε
(3) A ⟶ ASd
(4) A ⟶ ε
(5) B ⟶ SAc
(6) B ⟶ fC
(7) B ⟶ ε
(8) C ⟶ Sg
(9) C ⟶ Ch
(10) C ⟶ ε
(11) D ⟶ aBD
(12) D ⟶ ε

a) Sestrojte funkce FIRST a FOLLOW.
b) Sestrojte řídicí množiny jejích pravidel.
c) Určete, zda je LL(1).

4.10

K následující gramatice sestrojte LL(1)-analyzátor, pokud existuje.

(1) S ⟶ IT
(2) I ⟶ if E then S
(3) T ⟶ else S
(4) T ⟶ ε
(5) S ⟶ while E do S
(6) S ⟶ begin SS′end
(7) S′⟶ ; SS′
(8) S′⟶ ε
(9) S ⟶ s
(10) E ⟶ cE′
(11) E′⟶ or cE′
(12) E′⟶ ε .

4.11

Pro gramatiku

(0) S ⟶ A
(1) A ⟶ A + B
(2) A ⟶ B
(3) B ⟶ B * C
(4) B ⟶ C
(5) C ⟶ (A)
(6) C ⟶ a

sestrojte funkce FIRST a FOLLOW

Na řešení této úlohy navazuje 4.17.

(4.12)

Prozkoumejte následující gramatiku popisující regulární výrazy nad abecedou {a, b} .

(1) E → TE′
(2) E′ → +E
(3) E′ → ε
(4) T → FT′
(5) T′ → T
(6) T′ → ε
(7) F → PF′
(8) F′ → *F
(9) F′ → ε
(10) P → (E)
(11) P → a
(12) P → b
(13) P → Λ

a) Pro všechny neterminály této gramatiky vypočtěte hodnoty funkcí FIRST a FOLLOW.
b) Ukažte, že gramatika je LL(1) a sestrojte LL(1)-analyzátor.

(4.13)

Dokažte, že každý regulární jazyk je LL(1).

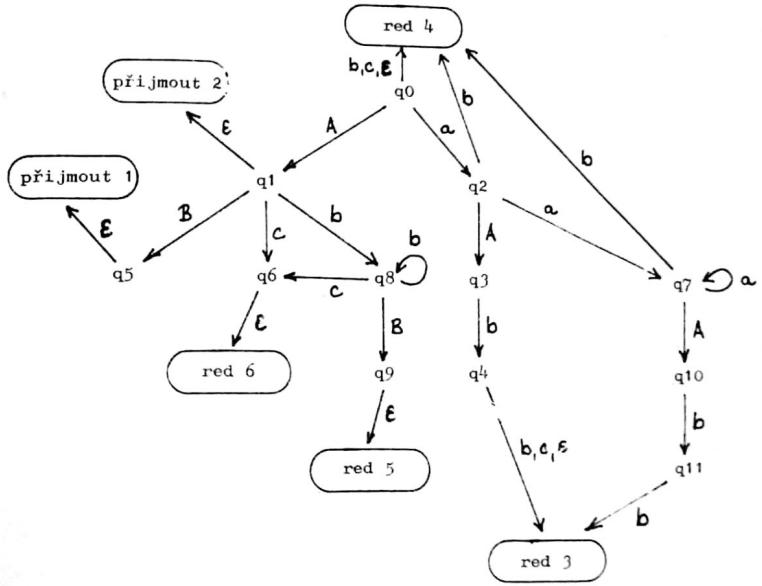
LR-analýza

(4.14)

Pro gramatiku

(1) S → AB
(2) S → A
(3) A → aAb
(4) A → ε
(5) B → bB
(6) B → c

mějme dán LR(1)-analyzátor:



Zapište výpočty tohoto analyzátoru pro řetězce
a) aabbbbc
b) aaabbb
c) aacbb

(4.15)

Sestrojte LR(0)-analyzátor pro gramatiku

(1) S′ → S
(2) S → (SS)
(3) S → ()

4.16

1) Sestrojte položkový automat pro gramatiku

(1) $S' \longrightarrow Sc$
(2) $S \longrightarrow SA$
(3) $S \longrightarrow \varepsilon$
(4) $A \longrightarrow aSb$.

2) Pokud je gramatika LR(0), sestrojte její LR(0)-analyzátor.

Připomeňme, že redukovaná bezkontextová gramatika $G = (\Pi, \Sigma, S, P)$ se nazývá SLR(1), jestliže

1. Počáteční neterminál S se nevyskytuje na pravé straně žádného pravidla z P.
2. Pro každý stav q položkového automatu gramatiky G platí:
   a) jestliže $A \longrightarrow \alpha.$, $B \longrightarrow \beta.$ jsou dvě různé úplné položky vyskytující se současně v q , potom
   $FOLLOW(A) \cap FOLLOW(B) = \emptyset$,
   b) jestliže se v q současně vyskytují položky tvaru $A \longrightarrow \alpha.$ a $B \longrightarrow \beta.x\gamma$, kde x je terminální symbol, potom $x \notin FOLLOW(A)$.

4.17

1) Ke gramatice

(0) $S \longrightarrow A$
(1) $A \longrightarrow A + B$
(2) $A \longrightarrow B$
(3) $B \longrightarrow B * C$
(4) $B \longrightarrow C$
(5) $C \longrightarrow (A)$
(6) $C \longrightarrow a$

sestrojte položkový automat a rozhodněte, zda gramatika je LR(0).

2) Využijte výsledku př. 4.11 a bodu 1) a zkuste sestrojit SLR(1)-analyzátor.

4.18

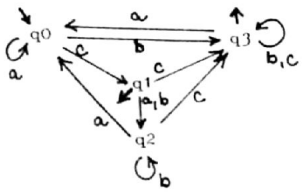
Zjistěte, zda následující gramatika je LR(0), SLR(1) nebo LR(1) a sestrojte příslušný analyzátor.

(0) $Z \longrightarrow A$
(1) $A \longrightarrow aBcB$
(2) $A \longrightarrow B$
(3) $A \longrightarrow D$
(4) $B \longrightarrow b$
(5) $B \longrightarrow Ff$
(6) $D \longrightarrow dE$
(7) $E \longrightarrow FcA$
(8) $E \longrightarrow FcE$
(9) $F \longrightarrow b$.

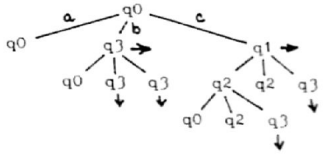
ŘEŠENÍ

1.1

a)



b)

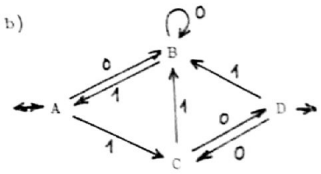


1.2

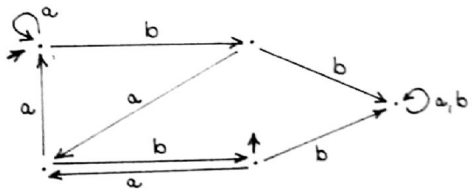
a)

	0	1
↔ A	B	C
B	B	A
C	D	B
← D	C	B

b)



1.3



1.4

a) $q_0q_1q_0q_0q_1q_0q_0q_0q_1q_2q_2$
b) $q_0q_0q_1q_0q_0q_1q_2q_2q_2q_2$

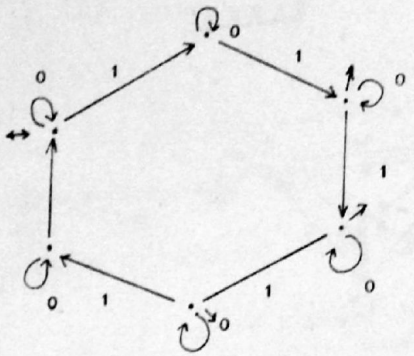
1.5

a) Slova v abecedě $\{0,1\}$ obsahující právě sudý počet nul.
b) Označme $0(w)$, resp. $1(w)$ počet výskytů nul, resp. jedniček ve slově w. Potom daný automat přijímá právě slova slova $w \in \{0,1\}^*$, pro něž $0(w) \not\equiv 0 \pmod 3$.
c) Slova v abecedě $\{a,b\}$ končící skupinou aspoň dvou b.
d) Slova nekončící dvěma nebo více nulami.
e) Slova zakončená neprázdnou skupinou jedniček předcházenou skupinou lichého počtu nul, před níž je opět 1 nebo začátek slova.
f) Slovo je zakončeno tím symbolem, který ve slově naposledy vytvořil úsek délky

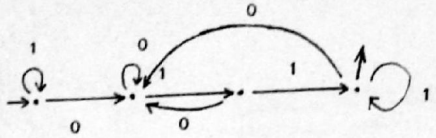
aspoň 2. Pokud žádný takový úsek neobsahuje, je slovo zakončeno stejným symbolem, jaký stojí na začátku slova.

1.6

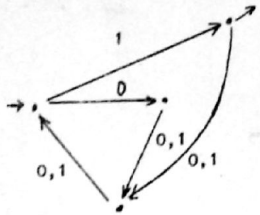
a)



b)



c)



1.7

Automat musí mít aspoň 5 stavů. Kdyby jich měl méně, pak pro libovolný symbol x dané abecedy a jistá $1 \le i < j < 5$ převedou řetězce x^i a x^j automat do téhož stavu. Proto také slova x^{i+5-j} a x^{j+5-j} převedou automat do téhož stavu. Avšak $5+i-j$ není dělitelné pěti, zatímco $j-j+5$ ano.

1.8

Přechodová funkce autonomního automatu je vždy dána grafem tvaru

pro jisté $i \ge 0$. Z toho plyne, že jazyk rozpoznávaný autonomním automatem je vždy unárním zápisem nějaké periodické posloupnosti přirozených čísel.

1.9

Na základě řetězce složeného z jediného typu symbolu se automat nejpozději po 3 krocích dostane do cyklu, který má délku 1,2,3 nebo 4. Číslo 12 je společným násobkem těchto čísel, tzn. 12 symbolů x zaručí pro každou z uvedených délek cyklu návrat do téhož stavu po několika obězích cyklu.

1.10

Ve všech případech předpokládejme, že existuje pravá kongruence $\sim$ jistého konečného indexu z taková, že uvažovaný jazyk je sjednocením některých jejích tříd. Tento předpoklad dovedeme ke sporu např. takto:

a) Uvažujme slova $0, 0^2, 0^3, \ldots, 0^{k+1}$. Potom pro jisté $1 \le i < j \le k+1$ je $0^i \sim 0^j$. Tedy také $0^i 1 0^i \sim 0^j 1 0^i$ - spor.

b) Vyjděme od slov $10, 10^2, \ldots, 10^{k+1}$. Aspoň dvě z nich padnou do stejné třídy kongruence, tzn. $10^i \sim 10^j$ pro nějaká $1 \le i < j \le k+1$. Potom také $10^i 10^i \sim 10^j 10^i$. Přitom $10^i 10^i \in L_2$ a $10^i 10^j \notin L_2$ - spor.

c) Aspoň dva z řetězců $0, 0^2, \ldots, 0^{k+1}$ padnou do téže třídy kongruence, tzn. $0^i \sim 0^j$ pro jisté $1 \le i < j \le k$. Proto také $0^i 1 10^i \sim 0^j 1 10^i$, což je ve sporu s tím, že první slovo do L_j patří a druhé nikoliv.

d) Zvolme libovolné prvočíslo $p > u$. Aspoň dvě ze slov $0, \ldots, 0^{k+1}$ padnou do téže třídy kongruence, tzn. $0^i \sim 0^j$ pro jisté $1 \le i < j \le k+1$. Označme $r = j - i$. Ze vztahu $0^i \sim 0^j$ plyne, že $0^{i+r} \sim 0^{j+r}$ tzn. $0^i \sim 0^j \sim 0^{j+r}$ a indukcí dokážeme, že $0^j \sim 0^{j+s.r}$ pro každé $s \ge 0$. Vynásobením slovem 0^{p-j} konečně dostáváme, že $0^p \sim 0^{p+s.r}$ pro každé $s \ge 0$. Speciálně $0^p \sim 0^{p+p.r}$, což je spor s definicí jazyka L_4, neboť číslo $p+p.r = p(1+r)$ je složené.

1.11

a) 1011001

b) 00001

1.12

a) 1110001

b) 0011

1.14

Přechod od Mealyho stroje $M = (Q, \Sigma, \Delta, \sigma, \lambda)$ k ekvivalentnímu Mooreovu stroji $M' = (Q', \Sigma, \Delta, \sigma', \lambda')$ provedeme "rozštěpením" každého stavu původního stroje na tolik stavů, aby ke každému nově vzniklému stavu bylo možno přidružit jediný výstupní symbol.

Zvolme pevně $y_0 \in \Delta$. Množinu stavů Q', přechodovou a značkovací funkci σ' a λ' definujme takto:

$Q' =_{df} Q_1 \cup Q_2$, kde $Q_1 =_{df} \{(q,y) \in Q \times \Delta;\ (\exists (\bar{q},x) \in Q \times \Sigma)(\sigma(\bar{q},x) = q \wedge \lambda(\bar{q},x) = y)\}$,

$Q_2 =_{df} \{(q,y_0);\ q \in Q \wedge (\forall (\bar{q},x) \in Q \times \Sigma)(\sigma(\bar{q},x) \neq q)\}$

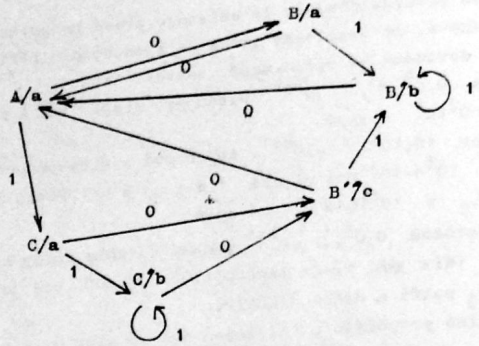
$\sigma'((q,y),x) =_{df} (\sigma(q,x), \lambda(q,x))$, pro všechna $(q,y) \in Q'$ a $x \in \Sigma$,

$\mu'((q,y)) =_{df} y$, pro všechna $(q,y) \in Q'$.

Abychom ověřili, že $M' \sim M$, stačí dokázat, že pro každé $(q,y) \in Q$ je $(q,y) \sim q$. Pro každé $u \in \Sigma^*, x \in \Sigma$ a $q \in Q$ je $\sigma'((q,y),ux) = (\sigma(q,ux), \lambda(\sigma(q,u),x))$, jak se snadno ověří indukcí podle délky řetězce u.

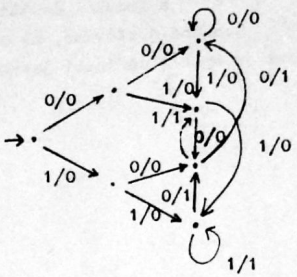
Podobně se ověří , že
$\lambda(\delta(q,u),x) = \mu(\delta(q,ux))$.
Tím je ekvivalence $(q,y) \sim q$ dokázána.

1.15

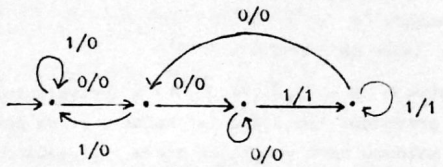


1.16

a)

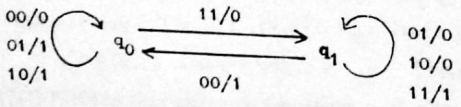


b)



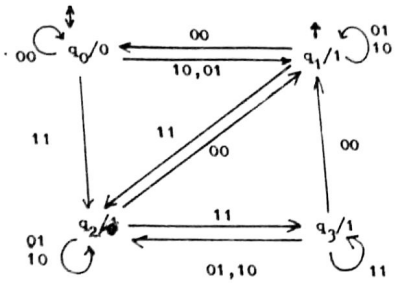
1.17

(q_0 je stav bez přenosu, stav q_1 s přenosem jedničky do vyššího řádu)



Stroj začíná výpočet ve stavu q_0. Končí-li v q_0, je vydán úplný výsledek. Zakončení v q_1 indikuje přetečení.

1.18

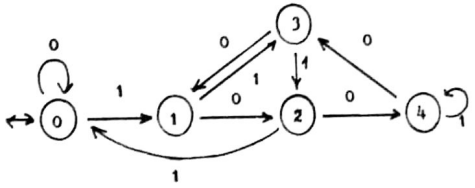


(stavy bez přenosu 1)

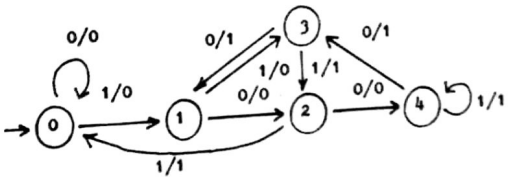
(stavy s přenosem 1)

Výpočet začíná v q_0. Stavy neoznačené jako koncové indikují přetečení.

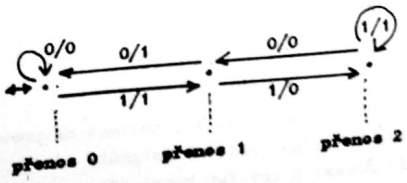
1.19



1.20

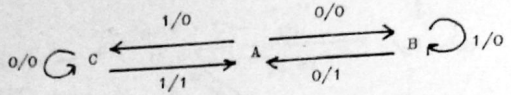


1.21



(1.22)

Automat



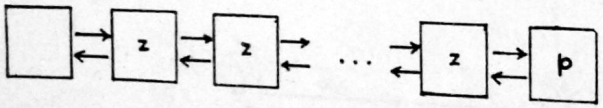
zpracuje danou vstupní posloupnost uvedeným způsobem, jestliže je výpočet zahájen ve stavu A.
Automat s menším počtem stavů nemůže být řešením, neboť v uvedeném výpočtu odpovídá automat na podslovo 00 trojím různým způsobem : 01, 10, 00.
V příslušných místech výpočtu se proto automat nutně nachází v různých stavech.

(1.23)

Úloha je podrobně popsána v knize: M. Minsky, "Computation: Finite and Infinite Machines", Prentice-Hall 1967.

Uvedeme nástin možného řešení. Na začátku vyšle "generál" dva signály - rychlý a pomalý. Signály jsou reprezentovány dvěma různými symboly. Rychlý symbol se pohybuje třikrát rychleji než pomalý. Toho lze docílit tak, že každý "voják" předává rychlý signál okamžitě v následujícím taktu dál, zatímco pomalý signál na dva takty zadrží.
Každý z automatů má jistý význačný stav p - "připraven". Automat na jednom konci řady je v tomto stavu od začátku, automat na druhém konci řady ("generál") do tohoto stavu přejde po vyslání signálů. V tomto stavu vydává výstupní symbol, podle něhož mohou sousední automaty zjistit, že se nachází ve stavu p .
Automaty lze navrhnout tak, že se oba signály od automatu ve stavu p "odrážejí". Podrobněji: jestliže nějaký automat obdrží signál např. zleva a zjistí, že že automat napravo je ve stavu p , potom signál nepředává napravo, ale vyšle tentýž signál zpět.
Jestliže se v automatu, který dosud není ve stavu p , setká pomalý a rychlý signál přecházející z opačných stran, automat přejde do stavu p , před tím však na obě strany vyšle rychlý i pomalý signál.
Automat, který zjistí, že oba jeho bezprostřední sousedé jsou ve stavu p ,přechází okamžitě do stavu "střelba".
Proberme, jak řada automatů funguje. Automaty jsou propojeny takto:

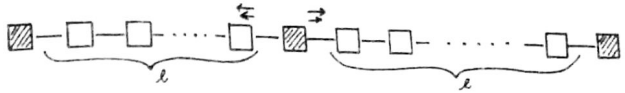
generál



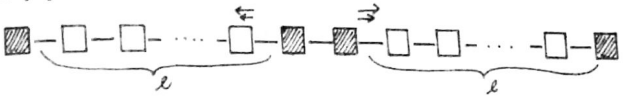
"Vojáci" jsou na počátku v základním stavu z , automat na pravém konci je trvale ve stavu p . V prvním taktu vyšle "generál" oba signály směrem doprava a přejde do stavu p . Rychlý signál dorazí k prvnímu konci dříve, odrazí se, šíří se zpět a setká se s pomalým signálem přesně uprostřed řady, díky tomu, že rychlý signál

je třikrát rychlejší než pomalý. Obsahuje-li řada lichý počet automatů , setkají se signály uvnitř jediného automatu, který vyšle dvojici signálů na obě strany a přejde do stavu p . Obsahuje-li řada sudý počet automatů, je křížení signálů detekováno současně ve dvojici automatů uprostřed řady. Tyto dva automaty přejdou do stavu p poté, co vyslaly dvojice signálů doprava i doleva. (Viz obr.)

a) lichý počet



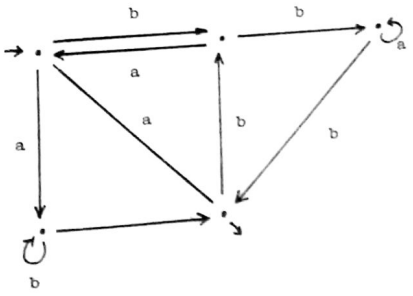
b) sudý počet



Každý z automatů ve stavu p v tuto chvíli sousedí s aspoň jedním úsekem 1 automatů, které dosud ve stavu p nejsou. Předpokládejme, že l = 2k+1 nebo l = 2k+2 pro nějaké k > 0. Potom k dalšímu setkání signálů dojde uprostřed zmíněných úseků délky l, které se uvedeným způsobem rozdělí na dvojice úseků délky k oddělené jedním nebo dvěma automaty ve stavu p atd. Každý automat ve stavu p sousedí nyní s aspoň jedním úsekem délky k složeným z automatů, které dosud ve stavu p nejsou. Popsaný proces pokračuje do okamžiku, kdy "živé" úseky mají délku 1 nebo 2. Automaty je třeba navrhnout tak, aby potom všechny automaty , které dosud nejsou ve stavu p , do tohoto stavu během jednoho taktu přešly. Vnásledujícím taktu přejdou všechny automaty do stavu "střelba".

Nakreslit graf vhodného automatu je velmi pracné. Rozmyslete si aspoň, jakým způsobem je jednotlivé požadované funkce možno v konečném automatu realizovat (šíření dvou různě rychlých signálů, odražení signálů od automatů ve stavu p , setkání signálů uprostřed úseku liché, resp. sudé délky, apod.).

(1.24)



1.25 (pro jednoduchost v tabulce uvádíme pouze indexy)

	0	1	0	1	0	1	0	1	0
q0	q1/0	q2/0	0	0	0	0	0	0	0
q1	q3/1	q4/0	1	1	1	1	1	1	1
q2	q0/0	q1/1	0	2	2	2	2	2	2
q3	q4/0	q5/1	0	2	2	2	3	3	3
q4	q5/0	q5/0	0	0	0	4	4	4	4
q5	q2/1	q4/0	1	1	1	1	1	1	5

Automat je redukovaný.

a) Rozlišit stavy q1, q5 vyžaduje posloupnost délky 4. Jednou takovou posloupností je 0010 .

b) Rozlišit stavy q0, q4 vyžaduje posloupnost délky 2. Takovou posloupností je např. 10 .

1.26

	0	1	0	1	0	1	0	1	0	1
A	B/1	C/0	A	A	A	A	A	A	A	A
B	B/1	A/0	A	A	A	B	B	B	B	B
C	A/0	B/0	C	C	C	C	C	C	C	C
D	E/1	A/0	A	A	A	B	B	B	D	D
E	D/1	E/0	A	A	A	B	B	E	E	E
F	B/1	C/0	A	A	A	A	A	A	A	A

a) K rozlišení stavů D, E je zapotřebí posloupnosti délky 3 . Tyto stavy rozliší např. posloupnost 110 .

b) K rozlišení stavů B, D je zapotřebí posloupnost délky 4. Tyto stavy rozliší např. posloupnost 0110 .

c) Stavy A, F jsou ekvivalentní. Nelze je tedy rozlišit žádnou vstupní posloupností.

1.27

a) K rozlišení stavů q_2, q_6 je zapotřebí posloupnost délky aspoň 3, rozliší je např. posloupnost 000.

b) Stavy q_0, q_6 rozliší např. posloupnost 000 , žádná kratší posloupnost je nerozliší.

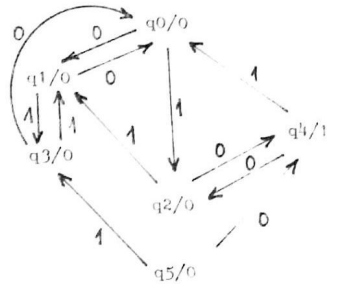
c) Stavy q_2, q_3 rozliší posloupnost 1 , prázdná posloupnost je nerozliší.

d) Stavy q_1, q_5 rozliší už prázdná posloupnost.

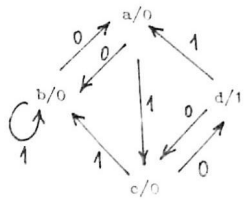
1.28

	0	1	výst.	0	1	0	1
q0	q1	q2	0	0	0	0	0
q1	q0	q3	0	0	0	1	1
q2	q4	q1	0	0	2	2	2
q3	q0	q1	0	0	0	1	1
q4	q2	q0	1	4	4	4	4
q5	q4	q3	0	0	2	2	2

graf výchozího automatu



graf reduktu



1.29

	0	1	výst.	0	1	0	1
q0	q1	q2	1	0	0	0	0
q1	q3	q0	0	1	1	1	1
q2	q4	q5	0	1	1	2	2
q3	q0	q2	0	1	3	3	3
q4	q2	q5	0	1	1	2	2
q5	q0	q3	0	1	3	5	5

1.30

	0	1	výst.	0	1	0	1
q0	q0	q1	0	0	0	0	0
q1	q1	q2	1	1	1	1	1
q2	q3	q1	1	1	2	2	2
q3	q2	q4	0	0	3	3	3
q4	q2	q3	0	0	3	3	3
q5	q5	q6	0	0	0	5	5
q6	q7	q5	0	0	3	6	6
q7	q7	q9	1	1	1	1	1
q8	q9	q8	0	0	3	3	3
q9	q8	q7	1	1	2	2	2

$q_3 \sim q_4 \sim q_8$,
$q_1 \sim q_7$,
$q_2 \sim q_9$

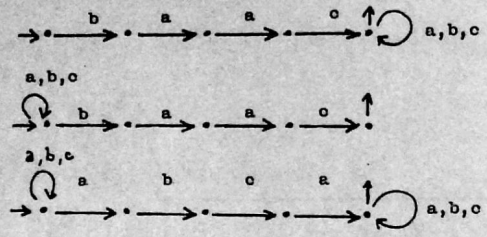
Všechny ostatní dvojice stavů jsou rozlišitelné.

1.31

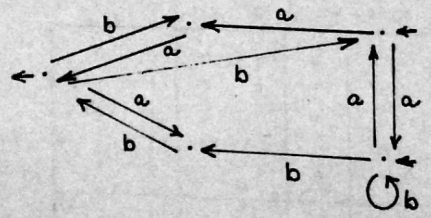
	0	1	výst.		0	1	0	1
q0	q0	q1	1	0	0	0	0	0
q1	q2	q3	1	0	1	1	1	1
q2	q3	q4	0	2	2	2	2	2
q3	q1	q0	0	2	3	3	3	3
q4	q3	q2	0	2	2	2	2	2
q5	q6	q7	0	2	3	5	5	5
q6	q8	q5	1	0	1	1	1	6
q7	q9	q6	1	0	1	7	7	7
q8	q5	q8	0	2	2	2	8	8
q9	q7	q6	0	2	3	3	9	9

$q2 \sim q4$, všechny ostatní dvojice stavů jsou rozlišitelné.

1.32

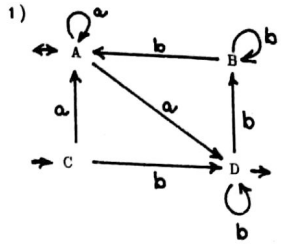


1.33

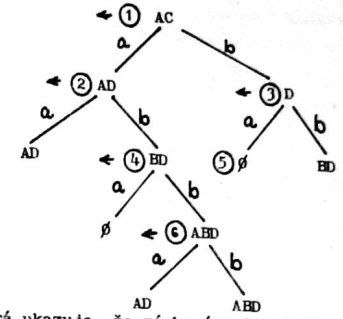


1.34

1)



2) Ekvivalentní deterministický automat je dán stromem:

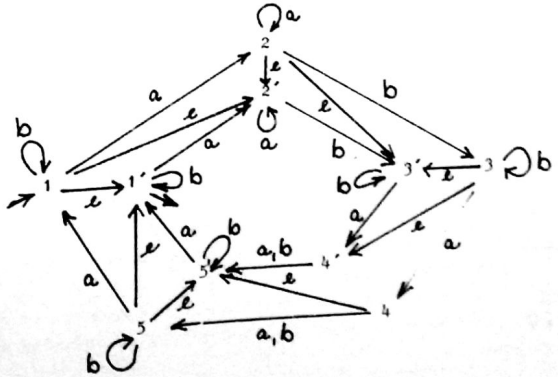


následuje tabulka redukce, která ukazuje, že získaný automat je redukovaný:

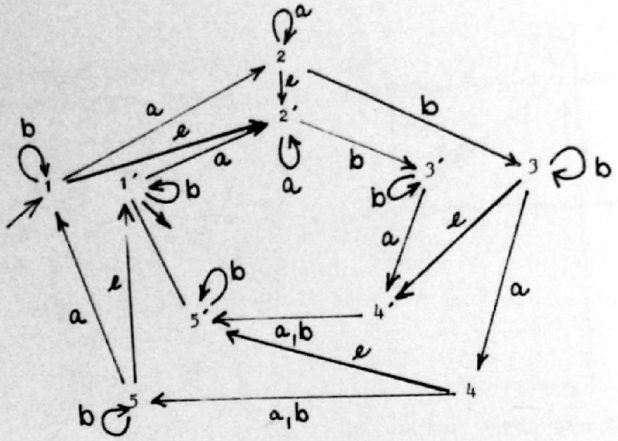
	a	b		a	b	a	b
← 1	2	3	1	1	1	1	1
← 2	2	4	1	1	1	2	2
← 3	5	4	1	3	3	3	3
← 4	5	6	1	3	4	4	4
5	5	5	5	5	5	5	5
← 6	2	6	1	1	6	6	6

1.35

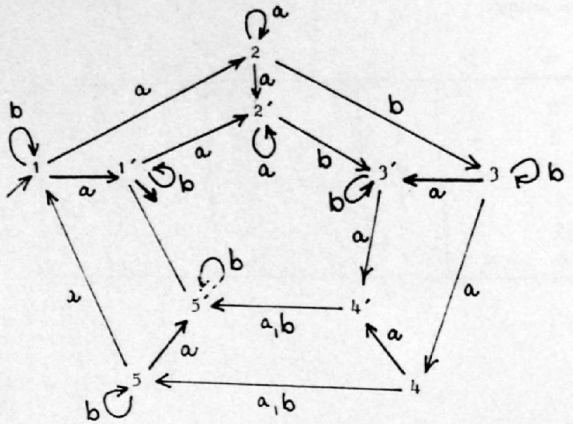
1)



2)



3)



4) Grafy jsou stejné jako v případech 1), 2), 3) s tím rozdílem , že vstupní vrchol je označen také jako koncový.

(1.36)

Stačí zjistit, zda v grafu automatu A vede cesta z některého počátečního do některého koncového stavu. To lze např. tímto algoritmem:

S:= množina počátečních stavů automatu A ;
<u>while</u> v grafu automatu A existuje hrana vedoucí z množiny S mimo tuto množinu
<u>do</u> S:= S ∪množ. stavů dosažitelných v jednom taktu z S ;

Zřejmě $L(A) \neq \emptyset$, právě když po skončení práce algoritmu obsahuje množina S aspoň jeden koncový stav.

(1.37)

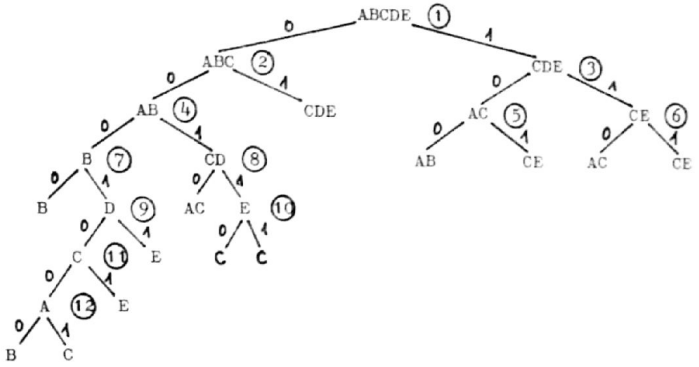
Od automatu A lze přejít k ekvivalentnímu deterministickému automatu B a od něho prohozením koncových a nekoncových stavů k automatu B' rozpoznávajícímu jazyk $X^* - L(A)$.

Potom $L(B') = \emptyset \Leftrightarrow L(A) = X^*$ a lze použít algoritmu z předchozího příkladu.

Všimněte si, že tento postup může být časově mnohem náročnější než postup z předchozího případu. To je dáno tím, že se jedná o problém výrazně složitější. Ve speciálním případě deterministických automatů jsou ovšem oba problémy stejně těžké.

(1.38)

Výstupní funkce automatu nemá pro určení synchronizačních posloupností žádný význam. Synchronizační posloupnosti snadno určíme z následujícího stavového stromu. U každého vrcholu je vypsáno, v kterých stavech se automat může nacházet po přečtení příslušné posloupnosti, jestliže na počátku mohl být v libovolném stavu.

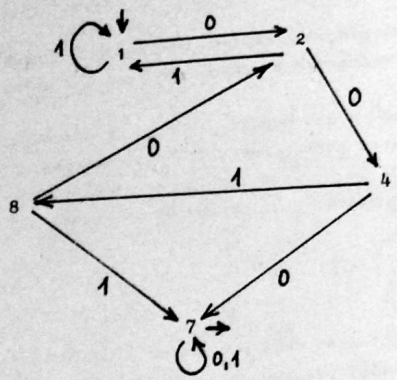


Stavový strom určuje nejen např. nejkratší synchronizační posloupnost, kterou je 000 , ale i deterministický automat rozpoznávající všechny synchronizační posloupnosti výchozího automatu. Počátečním stavem je stav v kořeni stromu, cílovými stavy všechny stavy odpovídající jednoprvkovým množinám, tzn. stavy označené jako 7, 9, 10, 11, 12.

Redukce automatu:

	0	1		0	1	0	1	0	1
→ 1	2	3	1	1	1	1	1	1	1
2	4	3	1	1	1	2	2	2	2
3	5	6	1	1	1	1	1	1	1
4	7	8	1	4	4	4	4	4	4
5	4	6	1	1	1	2	2	2	2
6	5	6	1	1	1	1	1	1	1
← 7	7	9	7	7	7	7	7	7	7
8	5	10	1	1	8	8	8	8	8
← 9	11	10	7	7	7	7	7	7	7
←10	11	11	7	7	7	7	7	7	7
←11	12	10	7	7	7	7	7	7	7
←12	7	11	7	7	7	7	7	7	7

Výsledný automat



dává celkem dobrý přehled o struktuře synchronizačních poslopností výchozího automatu.

(1.39)

Množina synchronizačních posloupností automatu A je rozpoznávána automatem $B = (Q', \Sigma, \sigma', q_0, F)$, kde
$Q' = \mathcal{P}(Q)$, $q_0 = Q$, $F = \{\{q\}; q \in Q\}$ a $\sigma'(M,x) = \{q; (\exists q' \in M)[\sigma(q',x)=q]\}$, pro každé $M \subseteq Q$ a $x \in \Sigma$.

(1.40)

1. alternativa řešení (obecně použitelná)

Stavy nedeterministického automatu jsou reprezentovány booleovským polem. K dispozici jsou dva exempláře tohoto pole reprezentované proměnnou S, proměnná sw určuje, který z obou exemplářů reprezentuje současný stav deterministického automatu. Druhý exemplář je tou dobou aktualizován tak, aby určoval následující stav deterministického automatu. Po skončení aktualizace se proměnná sw přepne na druhou hodnotu.

```
type  state = array [1..4] of boolean;
      insymbol = (a,b);
      outsymbol = (accept,reject);
var   S: array [0..1] of state;
      sw: 0..1 ;
```

Následující inicializační procedura nastavuje deterministický automat do počátečního stavu, tzn. právě všechny počáteční stavy výchozího nedeterministického automatu dostávají pravdivostní hodnotu "true".


```
procedure initialize ;
    begin   sw := 0;
            S [sw,1]  := true;
            S [sw,2]  := true;
            S [sw,3]  := false;
            S [sw,4]  := false
    end;
```

Vlastní automat je implementován následující procedurou, která je založena na myšlence podmnožinové konstrukce.

```
procedure automaton (in: insymbol; var out: outsymbol);
  var nsw: 0..1;
      i : 1..4;
  begin
    out := reject;
    nsw := 1-sw;
    for i := 1 to 4 do  S [nsw,i ] := false ;
    if S [sw,1 ] = true then
      case in of
        a: begin  S [nsw,1 ] := true;,
                  S [nsw,2 ] := true;
                  out:= accept
           end ;

        b: S [nsw,3 ] := true
      end;
    if S [sw,2 ] = true then
      case in of
        a: S [nsw,3 ] := true;
        b: begin S [nsw,4 ] := true;
                 out:= accept
           end
      end;
    if S [sw,3 ] = true then
      case in of
        a: begin S [nsw,3 ] := true;
                 S [nsw,1 ] := true;
                 out := accept
           end;
        b: S [nsw,3 ] := true
      end;
```

```
        if S[sw,4] = true then
          case in of
            a: begin S[nsw,1] := true;
                     out := accept
               end;
            b: S[nsw,3] := true
          end;
        sw := nsw
      end { konec procedury automaton } .
```

2. alternativa (o něco rychlejší, prostorově náročnější, použitelná při nevelkém počtu stavů)
Datový typ n_state odpovídá stavům výchozího nedeterministického automatu, typ state odpovídá stavům realizovaného deterministického automatu.

```
type n_state = 1..4;
     state = set of n_state;
     insymbol = (a,b);
     outsymbol = (accept, reject);
var  next_state: array [1..4,a..b] of state;
     q: state;
```

Inicializační procedura nyní nejen nastavuje proměnnou q na hodnotu počátečního stavu, ale ukládá také do proměnné next_state hodnoty přechodové funkce výchozího automatu.

```
procedure initialize;
  begin q := [1,2];
        next_state [1,a] := [1,2];
        next_state [2,a] := [3];
        next_state [3,a] := [1,3];
        next_state [4,a] := [1];
        next_state [1,b] := [3];
        next_state [2,b] := [4];
        next_state [3,b] := [3];
        next_state [4,b] := [3];
  end;

procedure automaton (in: insymbol; var out: outsymbol);
  var i: 1..4;
      qa: state;
  begin
    qa := [ ];
    for i:= 1 to 4 do
      if i in q then qa := qa + next_state [i,in];
    if 1 in qa or 4 in qa then out := accept
                          else out := reject;
    q := qa
  end
```


1.41

(n = 200)

Použijeme způsob odpovídající 1. alternativě z předchozího příkladu. Všimněte si, že implementace vycházející z převodu nedeterministického automatu na deterministický zde vůbec nepřichází v úvahu, protože nejmenší ekvivalentní deterministický automat má 2^{200} stavů.

```
const n = 200;

type state = array [0..n-1] of boolean;
     insymbol = (a,b);
     outsymbol = (accept, reject);

var  S: array [0..1] of state;
     sw: 0..1;


procedure initialize;

  var  i: integer

  begin
    for i:=0 to n-1 do S [0,i]  := false;
    S [0,0]  := true;
    sw := 0
  end ;
```

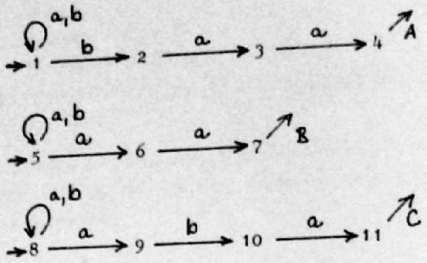
Ve vlastní proceduře implementující automat je oproti předchozímu příkladu využito značné uniformity přechodové funkce tohoto automatu.

```
procedure automaton (in: insymbol; var out: outsymbol);

  var i: 0..n-1;
      nsw: 0..1;

  begin
    out:= reject;
    nsw:= 1-sw;
    for i := 0 to n-1 do S [nsw,i] := false;
    if S [sw,0] = true and in = a then S [nsw,1]  := true;
    for i := 1 to n-2 do
        if S [sw,i] = true then
            case in of
                a: S [nsw,i+1] := true;
                b: begin S [nsw,i] := true;
                         S [nsw,0] := true;
                         out := accept
                   end
            end;
```

```
        if S[sw,n-1] = true then
          case in of
            a: begin S[nsw,0] := true;
                     out := accept
               end;
            b: begin S[nsw,n-1] := true;
                     S[nsw,0]   := true;
                     out:= accept
               end
          end;
        sw := nsw
      end
```

(1.42)

1) Nejprve sestrojíme nedeterministický konečný automat:



Všimněte si, že koncové stavy jsou rozlišeny na stav odpovídající nalezení klíče A, klíče B a klíče C. Z toho plynou modifikace jak podmnožinové konstrukce, tak způsobu implementace, které použijeme.

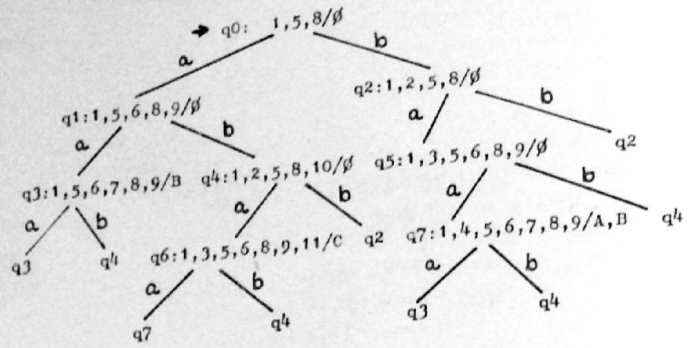
2) Popíšeme přímou softwarovou realizaci automatu na základě nedeterministického automatu sestrojeného sub 1).

```
type key = (A, B, C);
     insymbol = (a,b);
     outsymbol = set of key;
     state = array [1..11] of boolean;
var  S: array [0..1] of state;
     sw: 0..1;
procedure initialize;
    var i: 1..11;
    begin
      for i := 1 to 11 do S[0,i] := false;
      S[0,1] := true;
      S[0,5] := true;
      S[0,8] := true;
      sw := 0
    end;
```



```
procedure automaton ( in: insymbol; var out: outsymbol);
    var i: 1..11;
        nsw: 0..1;

  begin
    nsw := 1-sw ;
    out := [] ;
    for i := 1 to 11 do S[nsw,i] := false:
    if S[sw,1] = true then
       case in of
            a: S[nsw,1] := true;
            b: begin S[nsw,1] := true;
                     S[nsw,2] := true
               end
       end;
    if S[sw,2] = true and in = a then
       S[nsw,3] := true,
    if S[sw,3] = true and in = a then
       out := out + [A] ; { je zbytečné nastavovat S[nsw,4] , protože ze stavu 4 přechod neexistuje }
    { ze stavu 4 přechod neexistuje, není třeba testovat }
    if S[sw,5] = true then
       case in of
            a: begin S[nsw,5] := true;
                     S[nsw,6] := true
               end;
            b: S[nsw,5] := true
       end;
    if S[sw,6] = true and in = a then out := out + [B] ;
    { je zbytečné testovat i nastavovat S[sw,7] , resp. S[nsw,7] }
    if S[sw,8] = true then
       case in of
            a: begin S[nsw,8] := true;
                     S[nsw,9] := true
               end;
            b: S[nsw,8] := true
       end;
    if S[sw,9] = true and in = a then S[nsw,10] := true;
    if S[sw,10] = true and in = a then out := out + [C];
    sw := 1-nsw;
  end
```

3) Jinou možností je převést nedeterministický automat získaný v 1) na deterministický Mooreův automat a potom takto získaný automat implementovat. Užijeme podmnožinové konstrukce modifikované tak, že z každé podmnožiny zjišťujeme odpovídající výstupní symbol a nejenom, zda se jedná o koncový stav.



Redukce:

	a	b	výst.		a	b
q0	q1	q2	∅	0	0	0
q1	q3	q4	∅	0	1	1
q2	q5	q2	∅	0	0	2
q3	q3	q4	B	3	3	3
q4	q6	q2	∅	0	4	4
q5	q7	q4	∅	0	5	5
q6	q8	q4	C	6	6	6
q7	q3	q4	A,B	7	7	7

Automat je redukovaný.

4) Implementace shora uvedeného automatu.

```
type state = (q0,...,q7);
     insymbol = (a,b);
     outsymbol = (none, B, C, AB);

var  next-state: array [q0..q7,a..b] of state;
     out-function: array [q0..q7] of outsymbol;
     q: state;

procedure initialize;

   begin next-state [q0,a]  := q1 ; next-state [q0,b]  := q2;
         next-state [q1,a]  := q3 ; next-state [q1,b]  := q4;
         next-state [q2,a]  := q5 ; next-state [q2,b]  := q2;
             atd. podle shora uvedené tabulky

         next-state [q6,a]  := q7; next-state [q6,b]  := q4;
         next-state [q7,a]  := q3; next-state [q7,b]  := q4;
         out-function [q0]  := none;
         out-function [q1]  := none;,
         out-function [q2]  := none;
         out-function [q3]  := B;
             atd.
         out-function [q7]  := AB;
         q := q0
   end

procedure automaton (in: insymbol; var out: outsymbol);
     begin q := next-state [q,in] ;
           out:= out-function [q]
     end
```

Srovnáním obou přístupů zjišťujeme, že druhá verze je nejen rychlejší, ale v tomto případě vede i ke kratšímu programu. Časové nároky na vypracování obou implementací jsou přibližně stejné.

1.43

a) Řetězec w patří do $L_1(L_2 \cup L_3)$ právě když je tvaru $w=u.v$, kde $u \in L_1$ a $v \in L_2 \cup L_3$, tzn. právě když $w \in L_1L_2$ nebo $w \in L_1L_3$.

b) Analogicky jako a).

c) Podle definice je $(L^*)^* = \bigcup_{i=0}^{\infty} (L^*)^i$. Snadno se ověří, že $L^*.L^* = L^*$, a proto také $(L^*)^i = L^*$, pro každé $i > 0$. Jelikož $(L^*)^0 = \{\varepsilon\} \subseteq L^*$, dostáváme, že $(L^*)^* = L^*$.

d) Všechna slova z $(L_1 \cup L_2)^*$ buď patří do L_1^* nebo jsou tvaru $u_1v_1u_2v_2 \ldots u_nv_nu_{n+1}$, kde $u_i \in L_1^*$, pro všechna $1 \leq i \leq u+1$ a $v_j \in L_2$ pro všechna $1 \leq j \leq u$. Odtud plyne, že $(L_1 \cup L_2)^* \subseteq L_1^* (L_2L_1^*)^*$.
Obrácená inkluze je zřejmá.

e) Zřejmě $h(L_1), h(L_2) \subseteq h(L_1 \cup L_2)$, a proto $h(L_1) \cup h(L_2) \subseteq h(L_1 \cup L_2)$.
Obráceně, jestliže $w \in h(L_1 \cup L_2)$, je $w = h(v)$ pro nějaké $v \in L_1$ nebo $v \in L_2$.
Proto $w \in h(L_1) \cup h(L_2)$.

f) Plyne přímo z definic.

g) Pro každý řetězec u platí:
$u \in \partial_w(L_1 \cup L_2) \Leftrightarrow wu \in L_1 \cup L_2 \Leftrightarrow u \in \partial_w L_1 \vee u \in \partial_w L_2 \Leftrightarrow u \in \partial_w L_1 \cup \partial_w L_2$.

h) Pro každý řetězec u platí:
$u \in \partial_w(\Sigma^* - L) \Leftrightarrow wu \in \Sigma^* - L \Leftrightarrow wu \notin L \Leftrightarrow u \notin \partial_w L \Leftrightarrow u \in \Sigma^* - \partial_w L$

i) Plyne okamžitě ze skutečnosti, že pro libovolné dva řetězce u,v platí:
$(uv)^R = v^R u^R$.

j) Plyne ze skutečnosti, že pro každé dva řetězce u,v je $\partial(uv) = \partial(u).\partial(v)$.
Všimněte si, že vztah f) je speciálním případem vztahu j).

1.44

Vztahy neplatí např. pro

a) $L_1 = \{a\}$, $L_2 = \{b\}$;

b) $L_2 = \{a\}$, $L_2 = \{aa\}$;

c) $L_1 = \{a\}$ $L_2 = \{b\}$;

d) $L_1 = \{a\}$, $L_2 = \{b\}$, $h(a)=h(b)=c$. Inkluze $h(L_1 \cap L_2) \subseteq h(L_1) \cap h(L_2)$ ovšem platí vždy.

e) $L = \{a,b,c\}$, $h(c) = b$, $h(b) = a$, $h(a) = a$; $\varepsilon \in L_1^*(L_2L_1)^* - (L_1L_2)^*L_1$. Inkluze

f) $L_1 = \{a\}$, $L_2 = \{b\}$. Pak například ε, $ba \in L_1^*(L_2L_1)^* - (L_1L_2)^*L_1$. Inkluze $(L_1L_2)^*L_1 \subseteq L_1^*(L_2L_1)^*$ také vždy neplatí.

g) $L_1 = \{ab\}$, $L_2 = \{a,b\}$. Pak $L_2(L_2 \setminus L_1) = \{ab, bb\} \neq L_1$.

1.45

a) Plyne přímo ze vztahu $L_1 - L_2 = L_1 \cap (\Sigma^* - L_2)$ a uzavřenosti třídy $\mathcal{F}$ vůči průniku a doplňku.

b) Pro libovolný jazyk L je buď $L^+ = L^* - \{\varepsilon\}$, pokud $\varepsilon \notin L$, nebo $L^+ = L^*$, pokud $\varepsilon \in L$. Tvrzení je přímým důsledkem uzavřenosti třídy $\mathcal{F}$ vůči iteraci a a).

c) $L_2 = (L_1 - K_1) \cup K_2$ pro nějaké konečné jazyky K_1, K_2. Jelikož $K_1, K_2 \in \mathcal{F}$, plyne tvrzení z a) a uzavřenosti třídy $\mathcal{F}$ vůči sjednocení.

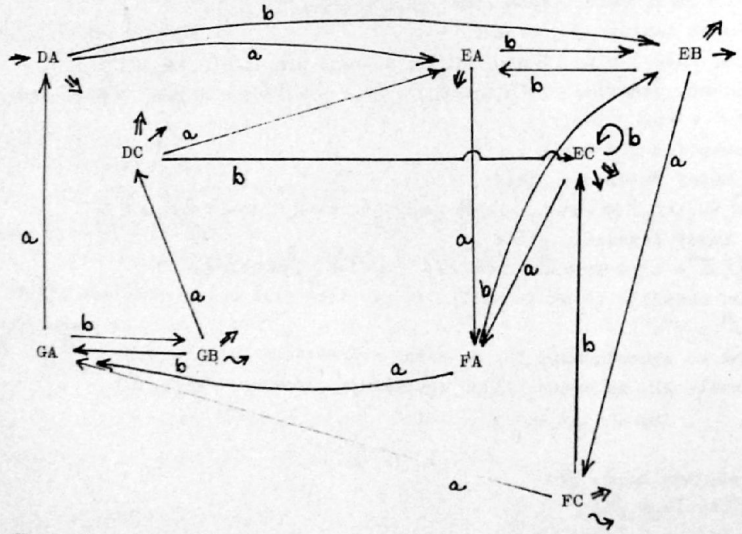
d) Budiž $A = (Q, \Sigma, \delta, q_0, F)$ konečný automat rozpoznávající L a $h: \tilde{\Sigma}^* \to \Sigma^*$ homomorfismus. Definujme automat $B = (Q, \tilde{\Sigma}, \tilde{\delta}, q_0, F)$ předpisem $\tilde{\delta}(q,a) = \delta^*(q,h(a))$, pro každé $q \in Q$, $a \in \tilde{\Sigma}$. (δ^* je zobecněním přechodové funkce δ). Ověřte, že pro každé $w \in \tilde{\Sigma}^*$ platí $\tilde{\delta}^*(q_0,w) = \delta^*(q_0, h(w))$ a že $L(B) = h^{-1}(L)$.

1.46

Stačí sestrojit konečné automaty rozpoznávající jazyky $L(A_1) - L(A_2)$ a $L(A_2) - L(A_1)$ a přesvědčit se, že oba tyto jazyky jsou neprázdné.

1.47

Grafy pro všechny tři jazyky se liší pouze koncovými ctavy.

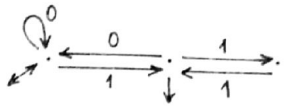


Koncové stavy pro jednotlivé případy jsou označeny takto:

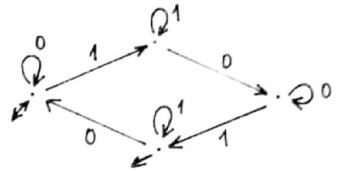
a) → b) ⇒ c) ⇝

1.48

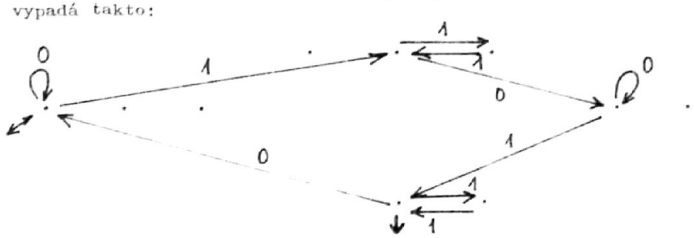
Automat rozpoznávající jazyk L_1 tvořený právě řetězci, v nichž všechny bloky jedniček mají lichou délku vypadá takto:



Automat rozpoznávající jazyk L_2 tvořený řetězci se sudým počtem bloků jedniček vypadá takto:

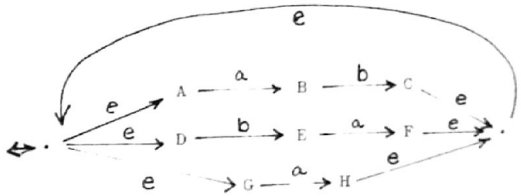


Automat rozpoznávající jazyk $L = L_1 \cap L_2$ získaný podle standardní konstrukce vypadá takto:

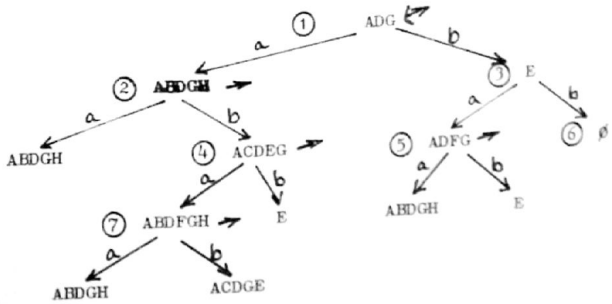


1.49

Nejprve sestrojíme graf nedeterministického automatu rozpoznávajícího zadaný jazyk:



Sestrojíme ekvivalentní deterministický automat:

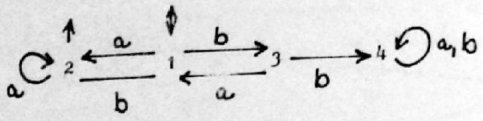


Automat zredukujeme:

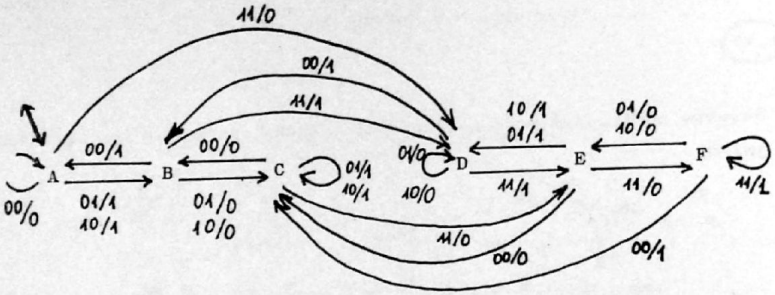
	a	b		a	b	a	b
↔1	2	3	1	1	1	1	1
← 2	2	4	1	1	2	2	2
3	5	6	3	3	3	3	3
← 4	7	3	1	1	1	1	1
← 5	2	3	1	1	1	1	1
6	6	6	3	6	6	6	6
← 7	2	4	1	1	2	2	2

normální tvar reduktu a jeho grafické vyjádření:

	a	b
↔1	2	3
← 2	2	1
3	1	4
4	4	4



(1.50)

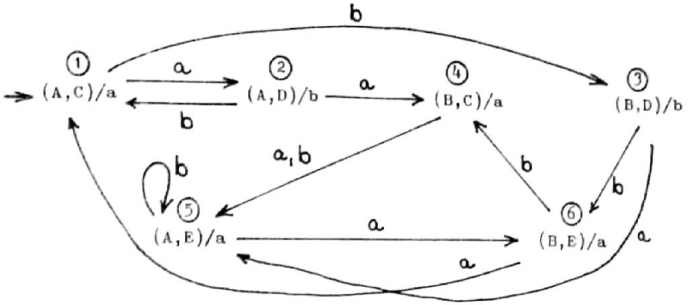


Získaný automat je redukovaný, jak ukazuje následující tabulka redukce.

	00	01 10	11		00	10 01	11
A	A/0	B/1	D/0	A	A	A	A
B	A/1	C/0	D/1	B	B	B	B
C	B/0	C/1	E/0	A	C	C	C
D	B/1	D/0	E/1	B	D	D	D
E	C/0	D/1	F/0	A	A	E	E
F	C/1	E/0	F/1	B	B	F	F

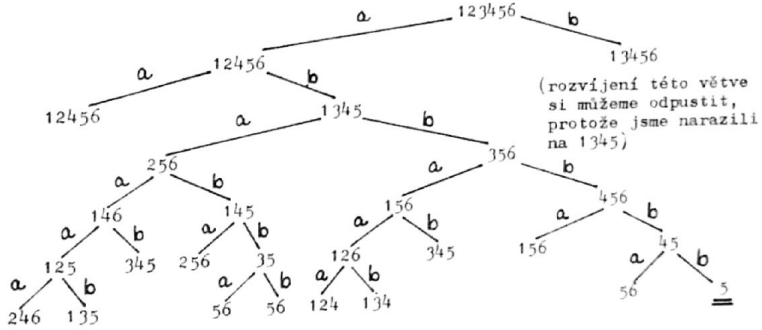
(1.51)

1) Uvažované zobrazení popisuje následující Mooreův automat, který sestrojíme standardním způsobem.



Snadno se můžeme přesvědčit, že získaný automat je redukovaný.

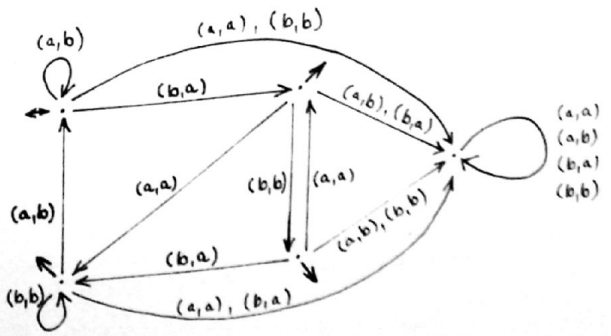
2) Sestrojujeme příslušný stavový strom, v tomto případě pouze tak dlouho, dokud nenalezneme nějakou synchronizační posloupnost.



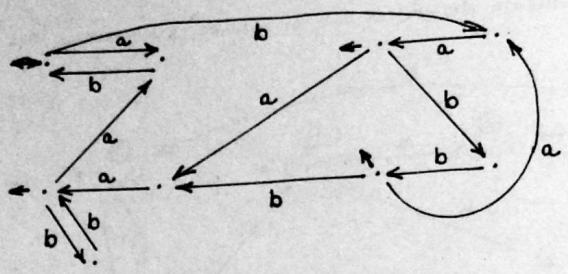
Pro systém tedy existují synchronizační posloupnosti. Jednou takovou posloupností je abbbbb.
Ze stavu 5 do stavu 1 lze přejít posloupností aa. Posloupnosti abbbbbaa lze proto použít k nastavení popsaného systému z libovolného stavu do stavu počátečního.

(1.52)

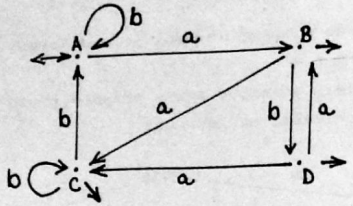
1)



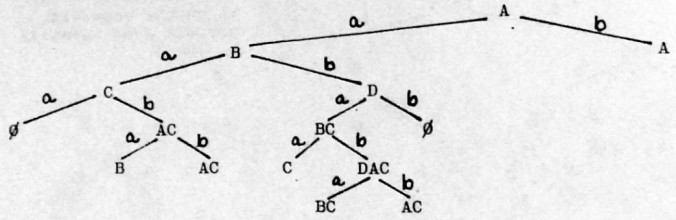
2)



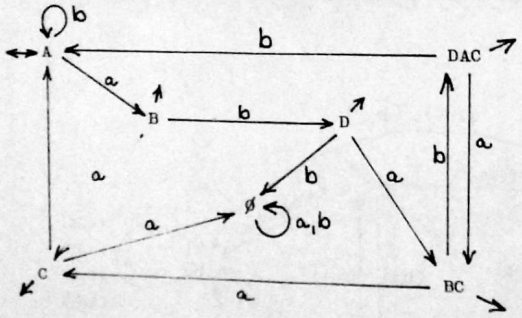
3) Jazyk nejprve popíšeme nedeterministickým konečným automatem:



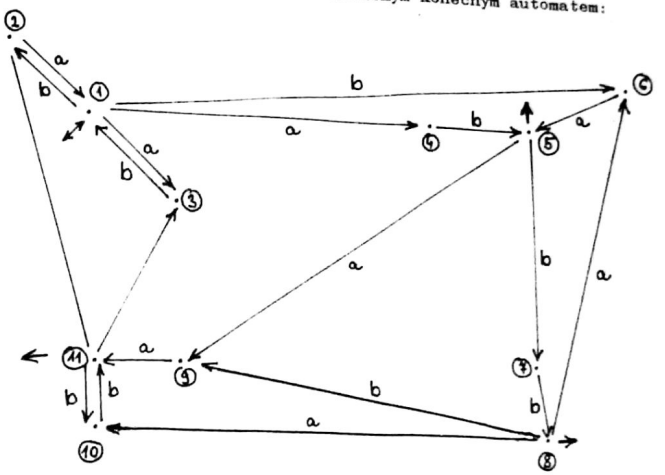
Odpovídající deterministický automat je dán stavovým stromem



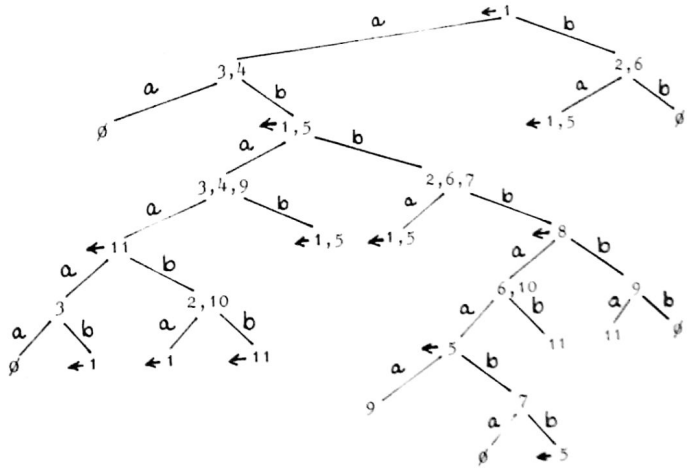
Algoritmus redukce ukáže, že ekvivalentní jsou pouze stavy A a AC. Minimální automat rozpoznávající zadaný jazyk je tedy dán následujícím grafem.



4) Jazyk nejprve popíšeme nedeterministickým konečným automatem:



Ekvivalentní deterministický automat reprezentujeme stromem:



Algoritmem redukce bychom se snadno mohli přesvědčit, že získaný automat už je redukovaný.

5) a) Přechodový diagram pro tento případ se z diagramu pro předchozí případ získá tak, že se ke každé hraně připíše navíc symbol n .

b) Redukovaný automat pro případ 5 musí mít aspoň tolik stavů jako redukovaný automat pro případ 4, protože posloupnosti rozlišitelné automatem z případu 4 jsou rozlišitelné i automatem z případu 5.

1.53 Označme L_i množinu všech synchronizačních posloupností stroje M_i $(1 \leq i \leq n)$. Všechny tyto jazyky jsou rozpoznatelné konečnými automaty, proto i jazyk $L_1 \cap \ldots \cap L_n$ je rozpoznatelný konečným automatem.

1.54 Nechť $L_1 = L(A_1), L_2 = L(A_2)$ pro deterministické konečné automaty $A_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$, $A_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$. Střídavý produkt jazyků L_1 a L_2 je pak rozpoznáván konečným automatem $A = (Q, \Sigma, \delta, q, F)$, kde $Q = Q_1 \times Q_2$, $\Sigma = \Sigma_1 \cup \Sigma_2$, $q = (q_1, q_2)$, $F = F_1 \times F_2$ a funkce δ je definována předpisem :

pro libovolné $(p_1, p_2) \in Q_1 \times Q_2$ a $x \in \Sigma_1 \cup \Sigma_2$ je $\delta((p_1, p_2), x) = (p_1', p_2')$

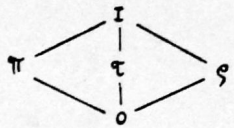
kde $$p_1' = \begin{cases} \delta_1(p_1, x), & \text{jestliže } x \in \Sigma_1 \\ p_1, & \text{jestliže } x \notin \Sigma_1 \end{cases} \qquad p_2' = \begin{cases} \delta_2(p_2, x), & \text{jestliže } x \in \Sigma_2 \\ p_2, & \text{jestliže } x \notin \Sigma_2 \end{cases}$$

1.55 Stačí zvolit libovolný deterministický automat $A = (Q, \Sigma, \delta, q_0, F)$ tak, aby $F = \{q_0\}$. Potom jazyk $L = L(A)$ má požadovanou vlastnost.

1.56 Nejprve prozkoumáme strukturu sv-rozkladů stavového prostoru stroje A :

$\langle 1,2 \rangle = \{1,2\}, \{3,4\}$
$\langle 1,3 \rangle = \{1,3\}, \{2,4\}$
$\langle 1,4 \rangle = \{1,4\}, \{2,3\}$
$\langle 2,3 \rangle = \langle 1,4 \rangle$
$\langle 2,4 \rangle = \langle 1,3 \rangle$
$\langle 3,4 \rangle = \langle 1,2 \rangle$.

Označme $\pi = \langle 1,2 \rangle$, $\tau = \langle 1,3 \rangle$, $\varrho = \langle 1,4 \rangle$.
Platí, že $\pi + \tau = \pi + \varrho = \tau + \varrho = I$ a $\pi \cdot \tau = \pi \cdot \varrho = \tau \cdot \varrho = 0$,
a proto další netriviální sv-rozklady již neexistují.
Struktura svazu sv-rozkladů vypadá takto :

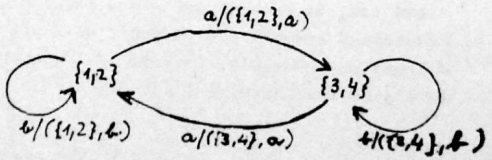


Stroj A má tedy netriviální sériovou dekompozici i netriviální paralelní dekompozici .

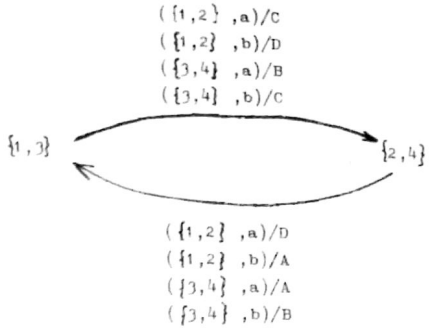
Ke konstrukci netriviální dekompozice zvolme např. sv-rozklad π . Ke konstrukci druhého stroje dekompozice můžeme využít např. rozkladu τ, neboť $\pi \cdot \tau = 0$.

Stroj A má tedy netriviální sériovou dekompozici $A_1 \circ A_2$, kde
$A_1 = (\pi, \Sigma, \pi \times \Sigma, \delta_\pi, id)$,
$A_2 = (\tau, \pi \times \Sigma, \Delta, \delta_2, \lambda_2)$
jsou dány stavovými diagramy

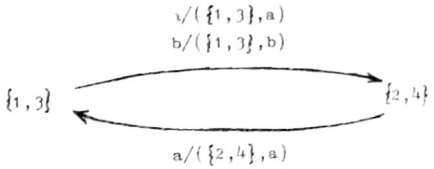
A_1 :



A_2 :



Ke konstrukci netriviální paralelní dekompozice opět můžeme použít sv-rozklady π a τ , neboť $\pi \cdot \tau = 0$. Stroj A má tedy netriviální paralelní dekompozici $B_1 | b | B_2$, kde $B_1 = A_1$ a
$B_2 = (\tau, \Sigma, \tau \times \Sigma, \delta_\tau, id)$ je dán stavovým diagramem



a spojka b je dána následující tabulkou.

b:

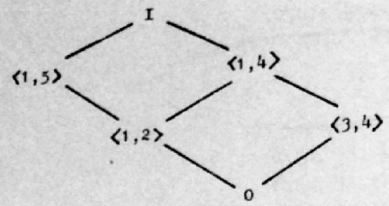
	({1,3},a)	({1,3},b)	({2,4},a)	({2,4},b)
({1,2},a)	C	-	D	-
({1,2},b)	-	D	-	A
({3,4},a)	B	-	A	-
({3,4},b)	-	C	-	B

1.57 1) Struktura svazu sv-rozkladů vypadá takto:

kde $\pi = \{1,4\}, \{2,3\}$.
Existuje tedy netriviální sériová dekompozice. Při konstrukci druhého automatu v dekompozici můžeme vyjít např. z rozkladu $\tau = \{1,2\}, \{3,4\}$, který vyhovuje podmínce $\pi \cdot \tau = 0$, ale který ovšem nemá substituční vlastnost.
Netriviální paralelní dekompozice tohoto stroje neexistuje.

2) V tomto případě jsou všechny sv-rozklady $\langle p,q\rangle$ $(p,q \in Q)$ rovny I. Neexistuje proto žádný netriviální sv-rozklad. Zadaný stroj nemá sériovou ani paralelní dekompozici.

3) Struktura svazu sv-rozkladů je v tomto případě dána diagramem



1.58

Uvedené jazyky je možno popsat například těmito regulárními výrazy:

a) $(a + b)^*aaba(a + b)^*$,

b) $abb(a + b)^* + (a + b)^*bbaa$,

c) $(b^*ab^*ab^*ab^*)^*b^*$

d) $aa(a + b)^*aa + ab(a + b)^*ab + ba(a + b)^*ba + bb(a + b)^*bb + aaa + aab + ba + bbb + bb$,

e) $b^* + b^*ab^*(bab^*)^*$.

1.59

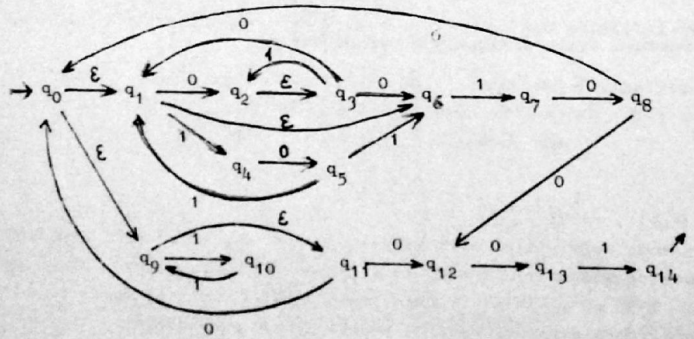
Ani jeden z obou vztahů neplatí:

a) Jazyk určený výrazem nalevo obsahuje prázdné slovo, zatímco jazyk určený výrazem napravo je neobsahuje.

b) Jazyk určený výrazem nalevo obsahuje slovo 0010011, zatímco jazyk určený výrazem napravo toto slovo neobsahuje.

1.60

Sestrojíme automat jehož stavy budou odpovídat místům v zadaném regulárním výrazu takto:



1.61

Nejprve utvoříme tři soustavy regulárních rovnic a jejich vyřešením získáme regulární výrazy pro jazyky L_1, L_2, L_3.

1) Soustava regulárních rovnic popisující automat reprezentující jazyk L_1 vypadá takto:

$$A = aA + aB + aC$$
$$B = bA + bB + bC$$
$$C = aB + \varepsilon \quad .$$

Z první rovnice dostáváme

$$A = a^*(aB + aC) = a^+(B + C) \quad .$$

Dosazením do druhé rovnice dostaneme

$$B = ba^+(B + C) + bB + bC = (ba^+ + b)B + (ba^+ + b)C.$$

Vyjádříme B:

$$B = (ba^+ + b)^*(ba^+ + b)C = (ba^+ + b)^+C.$$

Dosadíme do třetí rovnice:

$$C = a(ba^+ + b)^+C + \varepsilon \quad .$$

Odtud

$$C = [a(ba^+ + b)^+]^* \quad .$$

Zpětným dosazením získáme

$$B = (ba^+ + b)^+[a(ba^+ + b)^+]^*$$
$$A = a^+((ba^+ + b)^+[a(ba^+ + b)^+]^* + [a(ba^+ + b)^+]^*) = a^+([(ba^+ + b)^+ + \varepsilon][a(ba^+ + b)^+]^*) \quad .$$

Jazyk rozpoznávaný příslušným automatem je dán výrazem A + B, tzn.

$$(a^+[(ba^+ + b)^+ + \varepsilon] + (ba^+ + b)^+)[a(ba^+ + b)^+]^* .$$

2) Soustava regulárních rovnic pro druhý automat vypadá takto:

$$D = 0E + 1F + \varepsilon$$
$$E = 1D + 0E + 0F + \varepsilon$$
$$F = 0D + 1F \quad .$$

Postupnou eliminací proměnných získáme:

$$D = (1^+0)^*[0[1(1^+0)^*0 + 0(1^*0)(1^+0)^*0 + 0]^*.$$
$$[1(1^+0)^*(1^+0)^* + \varepsilon] + \varepsilon].$$

3) Regulární rovnice pro třetí automat vypadají takto:

$$G = 0G + 1H$$
$$H = 1G + 0H + \varepsilon .$$

Jejich řešením dostáváme:

$$G = 0^*1\ (10^*1 + 0)^* .$$

Hledaný regulární výraz pro jazyk $\sigma(L_1)$ bychom dostali tak, že za každý výskyt symbolu a v regulárním výrazu získaném v bodě 1) bychom dosadili celý regulární výraz pro D získaný v bodě 2) a za každý výskyt symbolu b bychom dosadili regulární výraz pro G získaný v bodě 3).

1.62

Mějme libovolný automat rozpoznávající jazyk L. Nechť u, v jsou prefixy nějakých slov patřících do L (tzn. $uu' \in L$, $vv' \in L$ pro nějaká $u', v' \in \{a,b\}^*$) a $|u| < |v|$. Potom u a v převedou automat A do různých stavů. Kdyby totiž u a v převedly A do téhož stavu q, potom slovo v' převede A ze stavu q do koncového stavu, protože $vv' \in L$. Potom ale i uv' převede automat do koncového stavu, což je spor s tím, že $|uv'| < 6$.

Spočítejme různé prefixy délky $i \leq 3$ slov z L:

i	0	1	2	3
počet prefixů	1	2	4	8

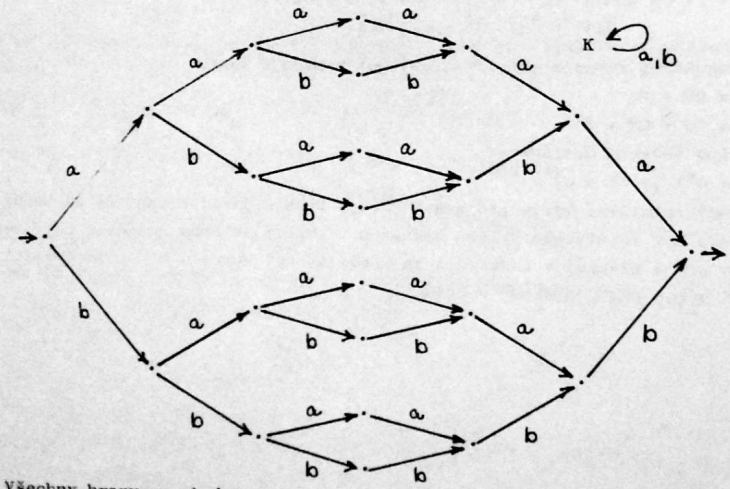
Různé prefixy délky $i \leq 3$ převádí automat A opět nutně do různých stavů.

Máme-li nyní prefix $a_1a_2a_3a_4$ délky 4, pak $a_3 = a_4$ a slovo lze doplnit na slovo z jazyka L pouze příponou a_2a_1. Proto prefixy délky 4 přivádějí automat A aspoň do tolika různých stavů, kolik je prefixů délky 2. Obdobně pro prefixy délky 5 a 6. Lze tedy sestavit tabulku

i	0	1	2	3	4	5	6
počet různých stavů pro prefixy délky i	1	2	4	8	4	2	1

Automat A musí tedy mít minimálně 22 stavů pro prefixy slov z L a aspoň jeden stav pro slova, která nelze prodloužit na slova z L. Automat A musí mít proto aspoň 23 stavů.

Získaný odhad je nejlepší možný, jak je vidět z automatu na následujícím obrázku.



Všechny hrany na obrázku neuvedené směřují do stavu K.

1.63

Zobecněním metody z předchozího příkladu dostáváme, že redukovaný automat rozpoznávající L má právě

$$2.(2^n-1) + 2^n + 1 = 3.2^n - 1 \text{ stavů}.$$

1.64

Budiž A libovolný nedeterministický konečný automat rozpoznávající L. Pro každé slovo $w \in L$ vybereme jeden pevný přijímající výpočet $V(w)$.

Mějme libovolné řetězce $vv' \in L$ a $uu' \in L$. Jestliže výpočet $V(vv')$ je po přečtení v ve stavu q a výpočet $V(uu')$ po přečtení u v tomtéž stavu q, pak i slova uv' a vu' patří do L, neboť od stavu q lze ve výpočtu nad u' pokračovat stejně jako ve výpočtu $V(uu')$ a ve výpočtu nad v' stejně jako ve výpočtu $V(vv')$. Z tohoto pozorování plyne, že

- jestliže $uu', vv' \in L$ a $|u| < |v|$, nebo
- jestliže $uu', vv' \in L$, $|u| = |v| \leq 3$ a $u \neq v$, nebo
- jestliže $uu', vv' \in L$, $|u'| = |v'| < 3$ a $u' \neq v'$,

potom výpočet $V(uu')$ je po přečtení u v jiném stavu než výpočet $V(vv')$ po přečtení v.

Odtud plyne analogicky jako v příkladě **1.62**, že automat A má alespoň 22 stavů. Tento dolní odhad je optimální, jak ukazuje automat v příkladě **1.62**. (Lze odstranit stav K a hrany do něj vedoucí.)

1.65

Zobecněním metody z předchozího případu dostáváme, že minimální nedeterministický konečný automat rozpoznávající L má právě

$$2.(2^n-1) + 2^n = 3.2^n - 2 \text{ stavů}.$$

1.66

Pro libovolný řetězec $w \in \{a,b\}^*$ definujme charakteristiku $ch(w)$ jako sufix délky n ze slova $b^n w$.

Budiž A libovolný deterministický konečný automat rozpoznávající jazyk L_n daný regulárním výrazem $(a+b)a(a+b)^{n-1}$. Ukážeme, že libovolná slova u, v různé charakteristiky převedou automat A do různých stavů.

Nechť $ch(u) = x_1 \ldots x_n$, $ch(v) = y_1 \ldots y_n$. Podle předpokladu $ch(u) \neq ch(v)$, tzn. existuje $1 \leq i \leq n$ takové, že $x_i \neq y_i$. Prodlužme obě slova u, v o totéž slovo w délky $i-1$, např. $w = b^{i-1}$. Potom n-tý symbol od konce ve slově uw je x_i, zatímco ve slově vw je to y_i. Proto pouze jedno ze slov uw, vw patří do L_n, tzn. každé z nich musí převést automat A do odlišného stavu. Tím je také dokázáno, že každé ze slov u, v převádí automat A do odlišného stavu.

Různých charakteristik slov z $\{a, b\}^*$ je 2^n. Proto automat A musí mít aspoň 2^n stavů.

Ukážeme, že získaný dolní odhad je optimální tím, že sestrojíme deterministický automat B_n s 2^n stavy rozpoznávající L_n.

Stavovým prostorem automatu B_n je množina $Q_n = \{a, b\}^n$. Počátečním stavem je $q0 = b^n$. Cílová množina je $F_n = a.\{a, b\}^{n-1}$. Pro každý stav $q = x_1 \dots x_n \in Q_n$ definujeme přechodovou funkci f_n takto:

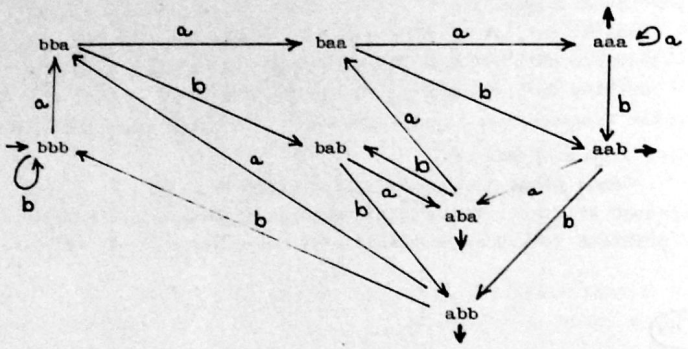
$$f_n(q,a) = x_2 \dots x_n a \quad , \qquad f_n(q,b) = x_2 \dots x_n b \ .$$

Je okamžitě zřejmé, že pro libovolný řetězec u je

$f_n(q0,u) = ch(u)$ a tedy

$f_n(q0,u) \in F_n$ právě když charakteristika řetězce u začíná symbolem a, tzn. když $u \in L_n$.

Např. pro n = 3 vypadá automat B_3 takto:



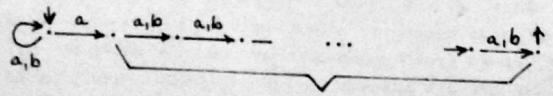
(1.67)

Budiž A libovolný nedeterministický konečný automat rozpoznávající jazyk L_n reprezentovaný daným regulárním výrazem. Automat A musí mít aspoň n+1 stavů. Jinak by výpočet přijímající slovo

$$ab^{n-1}$$

musel pc přečtení symbolu a projít v průběhu výpočtu dvakrát týmž stavem, tzn. existovalo by číslo $1 \le k \le n-1$ takové, že spolu s řetězcem ab^{n-1} by v L_n ležel i řetězec ab^{n-1+ik} pro každé $i \ge 0$, což je spor s definicí jazyka L_n.

Získaný dolní odhad je nejlepší možný, protože jazyk L_n je rozpoznáván nedeterministickým konečným automatem



který má n+1 stavů.

(1.68)

Tvrzení dokážeme sporem na základě Nerodovy věty. Předpokládáme, že existuje pravá kongruence $\sim$ na $\{a,b\}^*$ konečného indexu taková, že L je sjednocením jistých tříd z $\{a,b\}^*/\sim$. Nechť $\{a,b\}^*/\sim$ obsahuje k tříd. Potom aspoň dvě různá slova z posloupnosti $a, a^2, \dots, a^{k+1}$ padnou do téže třídy rozkladu, tzn. $a^i \sim a^j$ pro jistá $1 \le i < j \le k$. Pak také $a^i b a^i \sim a^j b a^i$, což vede ke sporu, neboť $a^i b a^j \in L$, zatímco $a^j b a^i \notin L$.

(1.69)

Vyjděme z toho, že $L \cap [0^*1^*] = \{0^n 1^n ; n \ge 0\}$ není regulární. Vzhledem k uzavřenosti třídy regulárních jazyků vůči průniku nemůže být regulární ani jazyk L.

(1.70)

Definujme homomorfismus h předpisem

$$h(a) = \begin{cases} a, & \text{jestliže } a \in \{(,)\} \\ \varepsilon, & \text{pro ostatní symboly,} \end{cases}$$

a regulární jazyk nad abecedou $\{(,)\}^*$ daný regulárním výrazem $(^*)^*$. Kdyby byl jazyk L regulární, byl by podle uzávěrových vlastností třídy regulárních jazyků i jazyk $L_1 = h(L) \cap [(^*)^*]$ regulárním jazykem. Avšak jazyk $L_1 = \{(^n)^n ; n \ge 0\}$ regulární není, takže není regulární ani jazyk L.

2.1

Pro každé $n \geq 1$ je řetězec $a^n b^n c^n$ odvoditelný takto: $S \Rightarrow^* a^{n-1}S(BC)^{n-1} \Rightarrow a^n bC(BC)^{n-1} \Rightarrow^* a^n bB^{n-1}C^n \Rightarrow^* a^n b^n C^n \Rightarrow^* a^n b^n c^n$. Je tedy $L \subseteq L(G)$.

Obrácená inkluze plyne z toho, že každý terminální řetězec z $L(G)$ obsahuje stejný počet symbolů a jako symbolů b i symbolů c a z toho, že pravidla gramatiky garantují, že všechny výskyty neterminálu C se podaří přepsat na terminál c pouze v případě, že se nejprve posunuly až za všechny neterminály B .

2.2

Jednotlivé jazyky lze generovat například těmito gramatikami:
(velká písmena označují neterminály, malá písmena terminály, S je vždy počáteční neterminál)

a) $S \to aS \mid bA \mid \varepsilon$
$A \to aA \mid bB$
$B \to aB \mid bS$

b) $S \to aS \mid bS \mid baabA$
$A \to aA \mid bA \mid \varepsilon$

c) $S \to 0S0 \mid A$
$A \to 1A \mid \varepsilon$

d) $S \to 0S0 \mid 1S1 \mid \varepsilon$

e) $S \to \varepsilon \mid \hat{A}\tilde{A}C \mid \hat{B}\tilde{B}C$
$C \to A\tilde{A}C \mid B\tilde{B}C \mid \varepsilon$

$\tilde{A}A \to A\vec{A}$
$\tilde{A}B \to B\vec{A}$
$\tilde{B}A \to A\vec{B}$
$\tilde{B}B \to B\vec{B}$

$\hat{A}A \to a\hat{A}$
$\hat{A}B \to a\hat{B}$
$\hat{B}A \to b\hat{A}$
$\hat{B}B \to b\hat{B}$

$\hat{A}\tilde{A} \to a\bar{A}$
$\hat{A}\tilde{B} \to a\bar{B}$
$\hat{B}\tilde{A} \to b\bar{A}$
$\hat{B}\tilde{B} \to b\bar{B}$

$\bar{A}\tilde{A} \to a\bar{A}$
$\bar{A}\tilde{B} \to a\bar{B}$
$\bar{B}\tilde{A} \to b\bar{A}$
$\bar{B}\tilde{B} \to b\bar{B}$

$\bar{A} \to a$
$\bar{B} \to b$

f) $S \to C\vec{E}aD$
$\vec{B}a \to a\vec{B}$
$\vec{B}D \to \varepsilon \mid \vec{E}D$
$a\vec{E} \to \vec{B}aa$
$C\vec{E} \to \varepsilon \mid C\vec{B}$
$C \to \varepsilon$
$D \to \varepsilon$

2.3

Jazyk L je možno charakterizovat například takto: pro každý řetězec $w \in \{a,b\}^*$ platí:
$w \in L \Leftrightarrow$ a) w obsahuje o jeden výskyt symbolu a více než je výskytů b
b) každý vlastní prefix obsahuje aspoň tolik symbolů b, kolik obsahuje symbolů a .

Implikace $\Rightarrow$ se ověří snadno přímo z definice dané gramatiky.
Implikace $\Leftarrow$ se dokazuje indukcí podle délky slova w;
1) pro $|w|=1$ je tvrzení triviální.
2) Nechť implikace platí pro všechna slova w kratší než jisté $k > 2$.
Ukažme, že platí i pro slova délky k . Zvolme libovolné $w=a_1 \ldots a_k$ splňující podmínky a) i b). Znich okamžitě plyne, že $a_1 = b$, $a_k = a$. Proto pro jisté i platí, že v $a_1 \ldots a_i$ je stejně symbolů a jako b . Podmínky a) a b) jsou pak ovšem splněny i pro slovo $a_2 \ldots a_i$ i pro slovo $a_{i+1} \ldots a_k$, která podle indukčního předpokladu také patří do L . Podle prvního pravidla gramatiky proto patří do L i celé slovo $a_1 \ldots a_k$.

2.4

Označme L jazyk tvořený slovy, která mají stejný počet výskytů symbolů a,b.
Inkluze $L(G) \subseteq L$ je zřejmá.
Obrácená inkluze plyne z následujícího pozorování.
Nechť $w \in L$, $|w| \geq 2$. Jestliže se první a poslední symbol slova w liší, lze w psát ve tvaru $w = aub$ nebo $w = bua$, kde $u \in L$.
V opačném případě je tvaru $w = ava$ nebo $w = bvb$ pro jisté slovo v .
Rozebereme pouze první případ, druhý je zcela analogický. Z rovnosti počtu symbolů a,b ve slově ava plyne, že v lze psát ve tvaru $v = v_1 v_2$ tak , že $av_1 \in L$, $v_2 a \in L$, tzn. w lze psát ve tvaru $w = w_1 w_2$, pro nějaká $w_1, w_2 \in L$.

2.5

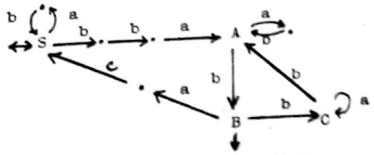
Množina neterminálů, ze kterých je odvoditelné prázdné slovo je $U = \{S,B,A\}$.
Hledaná nevypouštějící gramatika je pak

$S \to AB \mid A \mid B$
$A \to aAAb \mid aAb \mid ab \mid BS \mid B \mid S \mid CA \mid C$
$B \to BbA \mid bA \mid Bb \mid b \mid CaC$
$C \to aBB \mid aB \mid a \mid bS \mid b$.

2.6

$S \to A \mid C$
$A \to aA \mid aB \mid bC$
$B \to aB \mid bC$
$C \to bD \mid \varepsilon$
$D \to aB \mid bB \mid bD \mid \varepsilon$.

2.7



2.8

a) Jazyky popsané regulárními výrazy 01110, $(01)^*$, $(111)^*$ jsou po řadě generovány regulárními gramatikami

$S \longrightarrow 01110$, $S_1 \longrightarrow 01S_1 | \varepsilon$, $S_2 \longrightarrow 111S_2 | \varepsilon$.

Z nich vytvoříme hledanou gramatiku:

$S \longrightarrow 01110S_1$

$S_1 \longrightarrow 01S_1 \mid S_2$

$S_2 \longrightarrow 111S_2 | \varepsilon$.

b) Gramatiku vybudujeme induktivně podle stavby regulárního výrazu. V levém sloupci uvádíme regulární výrazy, v pravém sloupci jim odpovídající gramatiky.

0	$S_1 \longrightarrow 0$
1	$S_2 \longrightarrow 1$
1^*	$S_2 \longrightarrow 1S_2 \mid \varepsilon$
01^*	$S_1 \longrightarrow 0S_2$ $S_2 \longrightarrow 1S_2 \mid \varepsilon$
101	$S_3 \longrightarrow 101$
$(01^* + 101)$	$S_4 \longrightarrow S_1 \mid S_3$ $S_1 \longrightarrow 0S_2$ $S_2 \longrightarrow 1S_2 \mid \varepsilon$ $S_3 \longrightarrow 101$
$(01^* + 101)^*$	$S_4 \longrightarrow S_1 \mid S_3 \mid \varepsilon$ $S_1 \longrightarrow 0S_2$ $S_2 \longrightarrow 1S_2 \mid S_4$ $S_3 \longrightarrow 101S_4$

přímo lze vytvořit:

0^*1	$S_5 \longrightarrow 0S_5 \mid 1$

výsledek:

$(01^* + 101)^*0^*1$	$S_4 \longrightarrow S_1 \mid S_3 \mid S_5$ $S_1 \longrightarrow 0S_2$ $S_2 \longrightarrow 1S_2 \mid S_4$ $S_3 \longrightarrow 101S_4$ $S_5 \longrightarrow 0S_5 \mid 1$

c) $S \longrightarrow A \mid B \mid C$

$A \longrightarrow 01A \mid 101A \mid S$

$B \longrightarrow 1B \mid 00B \mid S$

$C \longrightarrow 0D$

$D \longrightarrow 1D \mid 0$

d) $S \longrightarrow aA \mid bcA$

$A \longrightarrow B \mid D \mid aA$

$B \longrightarrow aC$

$C \longrightarrow aC \mid A$

$D \longrightarrow dbD \mid cA$

e) Daný regulární výraz je ekvivalentní výrazu $(a + b + c)^*$. Proto lze tento jazyk generovat např. gramatikou $S \longrightarrow aS \mid bS \mid cS \mid \varepsilon$.

2.9

Nechť jazyky L_1, L_2 jsou generovány regulárními gramatikami $G_1 = (\Pi_1, \Sigma_1, S_1, P_1)$, $G_2 = (\Pi_2, \Sigma_2, S_2, P_2)$.

Případným přejmenováním neterminálů lze docílit, aby $\Pi_1 \cap \Pi_2 = \emptyset$.

a) Jazyk $L_1 . L_2$ je generován gramatikou $G_3 = (\Pi_1 \cup \Pi_2, \Sigma_1 \cup \Sigma_2, S_1, P_3)$, kde množina pravidel P_3 je utvořena takto: každé pravidlo z P_1 tvaru $X \longrightarrow w$, kde $w \in \Sigma_1^*$, je nahrazeno pravidlem $X \longrightarrow wS_2$. (Speciálně pravidlo tvaru $X \longrightarrow \varepsilon$ je nahrazeno pravidlem $X \longrightarrow S_2$). Ostatní pravidla z P_1 a všechna pravidla z P_2 jsou do P_3 převzata beze změny.

b) Jazyk $L_1 \cup L_2$ je generován gramatikou $G_4 = (\Pi_1 \cup \Pi_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, S, P_1 \cup P_2 \cup \{S \longrightarrow S_1 | S_2\})$

c) Jazyk L_1^* je generován gramatikou $G_5 = (\Pi_1, \Sigma_1, S, P_5)$, kde P_5 je utvořen takto: P_5 obsahuje
- pravidlo $S \longrightarrow \varepsilon$,
- pravidla $X \longrightarrow wS$ pro všechna $X \longrightarrow w \in P_1$ taková, že $w \in \Sigma_1^*$,
- všechna ostatní pravidla z P_1 .

2.10

a) $S \longrightarrow 0S0 \mid 1$

b) $S \longrightarrow 0S0 \mid 0A0$

$A \longrightarrow 1A \mid 1$

c) $S \longrightarrow 0S0 \mid 1S1 \mid \varepsilon$

d) $S \longrightarrow 0Si \mid S1 \mid \varepsilon$

e) $S \longrightarrow 0S1 \mid 0S11 \mid 01 \mid 011$

f) $S \longrightarrow S^* \mid (S+S) \mid (SS) \mid 0 \mid 1 \mid \emptyset \mid \varepsilon$

2.11

Nechť $G = (\Pi, \Sigma, S, P)$ je libovolná gramatika. Ke každému prvku $x \in \Sigma$ vytvoříme jeho "dvojníka" $\bar{x}$, tzn. nový symbol nevyskytující se v G. Takto vzniklou abecedu dvojníků označme $\bar{\Sigma}$.

Sestrojme gramatiku $G' = (\Pi \cup \bar{\Sigma}, \Sigma, S, P')$, kde $P' = \bar{P} \cup \{\bar{x} \longrightarrow x \,;\, x \in \Sigma\}$ a $\bar{P}$ vznikne z P nahrazením každého symbolu $x \in \Sigma$ v pravidlech jeho dvojníkem $\bar{x}$. V této gramatice se ještě mohou vyskytovat pravidla typu $\alpha \longrightarrow \varepsilon$, kde $|\alpha| \geq 2$. Taková pravidla definici separované gramatiky nevyhovují. Nechť $\alpha_1 \longrightarrow \varepsilon, \ldots, \alpha_n \longrightarrow \varepsilon$ jsou všechna takováto pravidla vyskytující se v P. Odstraníme je z G , přidáme nové neterminály $C_1, \ldots, C_n$ a pravidla $\left.\begin{matrix} \alpha_i \longrightarrow C_i \\ C_i \longrightarrow \varepsilon \end{matrix}\right\}$ $1 \leq i \leq n$.

Takto vzniklá gramatika už je separovaná a je ekvivalentní gramatice G.

2.12

Nechť $G = (\Pi, \Sigma, S, P)$ je libovolná monotónní gramatika . Podle předchozího cvičení z ní lze vytvořit ekvivalentní separovanou gramatiku. Přesvědčte se, že popsaným způsobem vzniká z monotónní gramatiky monotónní separovaná gramatika. Nazvěme ji G'. Každé pravidlo tvaru $A_1 A_2 \ldots A_m \longrightarrow B_1 \ldots B_n$ $(n \geq m)$, utvořené z neterminálů, které nemá tvar pravidel kontextové gramatiky, z gramatiky G odstraníme a nahradíme soustavou pravidel

$A_1A_2\dots A_m \longrightarrow C_1A_2\dots A_m$

$C_1A_2\dots A_m \longrightarrow C_1C_2\dots A_m$

.....

$C_1C_2\dots C_{m-1}A_m \longrightarrow C_1\dots C_{m-1}C_mB_{m+1}\dots B_n$

$C_1\dots C_mB_{m+1}\dots B_n \longrightarrow C_1 \quad C_{m-1}B_m\dots B_n$

.....

$C_1B_2\dots B_n \longrightarrow B_1\dots B_n$,kde $C_1, \dots, C_m$ jsou nově přidané neterminály , v žádných jiných pravidlech se nevyskytující. Tato pravidla už mají požadovaný tvar.

2.13

a) Jazyk L_1 je regulární, je generován například gramatikou

$S \longrightarrow bS \mid aS_1 \mid \varepsilon$

$S_1 \longrightarrow bS_1 \mid aS_2 \mid \varepsilon$

$S_2 \longrightarrow bS_2 \mid aS_3 \mid \varepsilon$

$S_3 \longrightarrow bS_3 \mid \varepsilon$.

b) Jazyk L_2 je bezkontextový, je generován například gramatikou

$S \longrightarrow aSB \mid aB$

$B \longrightarrow b \mid bb$.

Jazyk L_2 však není regulární, jak se dokáže buď přímo z Nerodovy věty (jestliže $0^i \not\sim 0^j$, pro $i < j$, potom $0^i1^i \not\sim 0^j1^i$, zatímco $0^i1^i \in L_2$ a $0^j1^i \notin L_2$), nebo např. z uzávěrových vlastností:

jazyk $L' = \{0^i1^j \; ; \; 1 \leq j \leq i \leq 2j\}$ se získá z L_2 operací zrcadlového obrazu a homomorfismu. Kdyby tedy byl L_2 regulární, byl by takový i L'. Pak by byl regulární i jazyk $L_2 \cap L' = \{0^i1^{i^2} \; ; \; i \geq 1\}$.

c) Jazyk L_3 je regulární, neboť je generován například gramatikou

$S \longrightarrow a + S \mid a$.

d) Jazyk L_4 není bezkontextový. Kdyby byl, existovala by pro něj čísla $m, n > 0$ s vlastnostmi uvedenými v lemmatu o vkládání. Zvolme tedy $a^icb^{2^i}$, kde $i = \max(m,n)$.

Potom podle uvedeného lemmatu musí v L_4 ležet i slovo délky $i + 1 + 2^i + r$, kde $0 < r \leq m \leq i$, žádné takové slovo však v L_4 neleží.

Jazyk L_4 je ovšem kontextový, neboť je generován například monotónní gramatikou

$S \longrightarrow SA \mid cBA$

$BA \longrightarrow ABB$

$cA \longrightarrow ac$

$B \longrightarrow b$.

Například slovo a^2cb^4 je generováno takto: $S \Longrightarrow SA \Longrightarrow cBAA \Longrightarrow cABBA \Longrightarrow acBBA \Longrightarrow acBABB \Longrightarrow acABBBB \Longrightarrow aacBBBB \overset{*}{\Longrightarrow} a^2cb^4$.

e) Jazyk L_5 je tvořen právě slovy v abecedě $\{a,b\}$, která obsahují stejný počet obou symbolů. Tento jazyk je však generovatelný i bezkontextovou gramatikou

$S \longrightarrow SS \mid aSb \mid bSa \mid \varepsilon$.

Regulární už L_5 není, neboť pak by byl regulární i jazyk $L_5 \cap [a^*b^*] = \{a^nb^n \; ; \; n \geq 0\}$.

f) Jazyk L_6 je regulární, je reprezentován např. regulárrním výrazem $(000)^*(111)^* + (000)^*0\,(111)^*1 + (000)^*00(111)^*11$.

g) Jazyk L_7 je regulární, je generován např. gramatikou

$S \longrightarrow a^3S \mid aB \mid aabB \mid bbB$

$B \longrightarrow b^3B \mid \varepsilon$.

3.1

1) V prvním kroku najdeme všechny neterminály, které lze přepsat na terminální řetězec. Postupně tedy konstruujeme:

$V_0 = \{a,b,c\}$,

$V_1 = \{a,b,c,S\}$, neboť $S \Rightarrow \varepsilon$ a $\varepsilon \in V_0^*$

$V_2 = \{a,b,c,S,C\}$, neboť $C \Rightarrow cS$ a $cS \in V_1^*$

$V_3 = V_2$

Terminální řetězec lze tedy odvodit jen z neterminálů S a C. (Kdyby mezi nimi nebyl počáteční symbol, nemělo by smysl dál pokračovat v algoritmu redukce, protože jazyk generovaný gramatikou by byl prázdný.)

Odstraněním všech pravidel, ve kterých se vyskytuje A nebo B dostaneme gramatiku

$S \longrightarrow aSb \mid \varepsilon$

$C \longrightarrow CC \mid cS$

Ve druhém kroku algoritmu hledáme všechny neterminály dosažitelné z S (v gramatice vzniklé po prvním kroku.) Konstruujeme tedy množiny

$U_0 = \{S\}$

$U_1 = \{S\} = U_0$.

Neterminál C se tedy v žádném odvození nevyskytuje. Jeho odstraněním dospíváme ke gramatice $S \longrightarrow aSb \mid \varepsilon$.

2) Redukcí dostaneme gramatiku

$S \longrightarrow aSa \mid bSb \mid E$

$E \longrightarrow \varepsilon$.

Přesvědčte se, že prohozením obou kroků redukce dostáváme gramatiku

$S \longrightarrow aSa \mid bSb \mid E$

$C \longrightarrow c$

$E \longrightarrow \varepsilon$, která není redukovaná.

3) Redukcí dostaneme gramatiku

$S \longrightarrow B \mid 0SB \mid \varepsilon$

$B \longrightarrow 1B \mid 1$.

3.2

Zkonstruujeme množinu neterminálů, které je možno přepsat na terminální řetězec:

$V_0 = \{a,b\}$

$V_1 = \{a,b,D\}$

$V_2 = \{a,b,D,A\}$

$V_3 = V_2$.

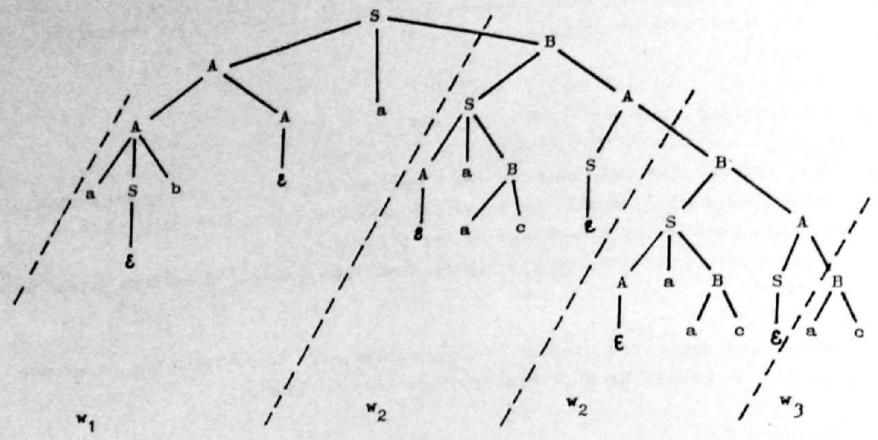
Hledaná množina je $\{A,D\}$, neobsahuje S, proto $L(G) = \emptyset$.

3.3

1) $S \Rightarrow AaB \Rightarrow AAaB \Rightarrow aSbAaB \Rightarrow abAaB \Rightarrow abaB \Rightarrow abaSA \Rightarrow abaaSBA \Rightarrow abaaBA \Rightarrow abaaacA \Rightarrow abaaacSB \Rightarrow abaaacB \Rightarrow abaaacac$.

2) $S \Rightarrow AaB \Rightarrow AaSA \Rightarrow AaSSB \Rightarrow AaSSac \Rightarrow AaSac \Rightarrow AaAaBac \Rightarrow AaAaacac \Rightarrow Aaaacac \Rightarrow AAaaacac \Rightarrow Aaaacac \Rightarrow aSbaaacac \Rightarrow abaaacac$.

3) Využijeme opakování neterminálu B na pravé krajní větvi derivačního stromu. Nahrazením menšího podstromu určeného výskytem neterminálu B větším podstromem určeného druhým výskytem B na zmíněné větvi dostaneme strom



Tento strom je opět derivačním stromem podle téže gramatiky (zdůvodněte) . Příslušná dekompozice je naznačena na obrázku.

4) Z daného stromu je vidět, že gramatika G obsahuje mimo jiné pravidla

$S \longrightarrow AaB$, $A \longrightarrow AA$, $A \longrightarrow \varepsilon$, $B \longrightarrow ac$. Řetězec aac je proto možné odvodit levými derivacemi:

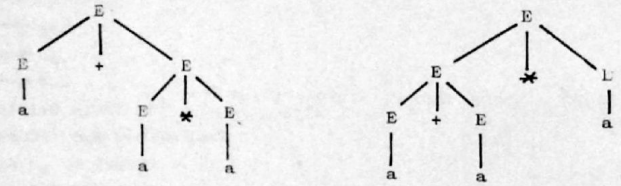
$S \Rightarrow AaB \Rightarrow aB \Rightarrow aac$,

$S \Rightarrow AaB \Rightarrow AAaB \Rightarrow AaB \Rightarrow aB \Rightarrow aac$ atd.

Gramatika G je proto víceznačná.

3.4

Jeden typ nejednoznačnosti se objevuje u výrazů typu a+a*a , u kterých není z dané gramatiky patrné, která z obou operací by se měla provádět jako první. Uvedený výraz má totiž podle uvedené gramatiky dva derivační stromy:



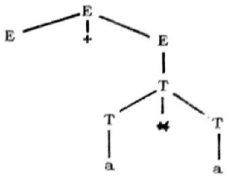
Tento typ nejednoznačnosti odstraníme obvyklým způsobem - stanovíme, že násobení váže těsněji než sčítání. V gramatice proto odlišíme , zda se jedná o součet (označíme E) , součin (označíme T) , nebo atomický výraz (označíme F). Konvenci o vztahu operace sčítání a násobení pak vyjádříme soustavou pravidel:

$E \longrightarrow E+E \mid T$

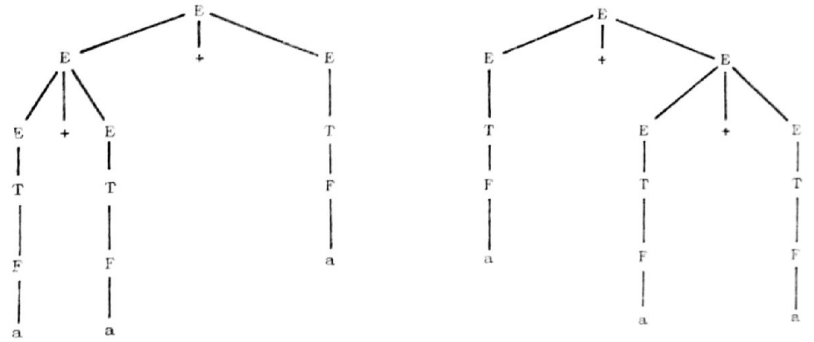
$T \longrightarrow T*T \mid F$

$F \longrightarrow (E) \mid a$

Získaná gramatika generuje původní jazyk, ale zmíněný typ nejednoznačnosti se v ní již neobjevuje. Např. výraz a+a*a má pouze derivační strom



I v upravené gramatice se však objevuje nejednoznačnost v tom, jak asociovat členy v delších součtech nebo součinech. Např. výrazu a+a+a odpovídají derivační stromy



Této víceznačnosti se můžeme zbavit tak, že do gramatiky zakódujeme pravidlo, že v součtech a součinech je třeba členy asociovat odleva , resp. odprava. Prvnímu případu odpovídá gramatika

$E \longrightarrow E+T \mid T$

$T \longrightarrow T*F \mid F$

$F \longrightarrow (E) \mid a$

druhému případu gramatika

$E \longrightarrow T+E \mid T$

$T \longrightarrow F*T \mid F$

$F \longrightarrow (E) \mid a$.

(Sestrojte u obou gramatik derivační strom výrazu a+a+a. Jak by vypadala gramatika určující pro sčítání asociování odleva a pro násobení asociování odprava?)

3.5

Nechť $L = L(A)$ pro nějaký deterministický automat $A = (Q, \Sigma, \delta, q_0, F)$. Od automatu A přejdeme standardním způsobem k regulární gramatice G , která obsahuje pravidla tvaru $q \longrightarrow ap$, pokud $\delta(q,a) = p$ resp $q \longrightarrow \varepsilon$, pokud $q \in F$. Výpočty automatu A jsou ve vzájemně jednoznačné korespondenci s odvozeními podle gramatiky G. Ke každému řetězci $w \in L$ tedy existuje právě jediné odvození podle gramatiky G.

3.6

Konstruovaný automat má jediný stav q , vstupní abecedu $\{a,+,*,(,)\}$, zásobníkovou abecedu $\{a,+,*,(,),E,T,F\}$, počátečním zásobníkovým symbolem je E. Přechodová funkce δ je definována takto:

$$\delta(q,\varepsilon,E) = \{(q,E+T),\ (q,T)\}$$
$$\delta(q,\varepsilon,T) = \{(q,T*F),\ (q,F)\}$$
$$\delta(q,\varepsilon,F) = \{(q,(E)),\ (q,a)\}$$
$$\delta(q,a,a) = \delta(q,+,+) = \delta(q,*,*) = \delta(q,(,() = \delta(q,),)) = \{(q,\varepsilon)\}.$$

3.7

Snadno se ověří, že libovolný řetězec $w \in \{[,]\}^*$ je generován uvedenou gramatikou právě když jsou splněny tyto dvě podmínky:

1) v každém prefixu u řetězce w je počet výskytů pravé závorky nejvýše roven počtu výskytů levé závorky;
2) v řetězci w jsou počty výskytů obou závorek stejné. Navržený automat proto pomocí počtu symbolů A v zásobníku registruje rozdíl mezi počtem levých a pravých závorek. Do koncového stavu přichází , jestliže bilance obou druhů závorek je vyrovnaná. Automat má vstupní abecedu $\{[,]\}$, zásobníkovou abecedu $\{Z,A\}$, počáteční zásobníkový symbol Z , množinu stavů $\{p,q\}$, počáteční stav p , koncový stav p a přechodovou funkci definovanou takto:

$$\delta(p,[,Z) = (q,AZ),$$
$$\delta(q,[,A) = (q,AA),$$
$$\delta(q,],A) = (q,\varepsilon),$$
$$\delta(q,\varepsilon,Z) = (p,Z).$$

Projděte výpočet automatu nad slovem [[][]][[[]]][] a nad slovem [[]]][[].

3.8

Navržený automat registruje pomocí počtu symbolů A v zásobníku bilanci počtu nul a jedniček v dosud přečteném úseku slova. Pomocí vnitřního stavu uchovává informaci o tom, který z obou symbolů zatím převažuje. Automat má stavy p,q_0,q_1 z nichž p je počáteční a zároveň jediný koncový. Zásobníková abeceda je $\{Z,A\}$, Z je počáteční zásobníkový symbol. Přechodová funkce je definována takto:

$$\delta(p,0,Z) = (q_0,AZ) \qquad \delta(q_0,\varepsilon,Z) = (p,Z)$$
$$\delta(p,1,Z) = (q_1,AZ) \qquad \delta(q_1,1,A) = (q_1,AA)$$
$$\delta(q_0,0,A) = (q_0,AA) \qquad \delta(q_1,0,A) = (q_1,\varepsilon)$$
$$\delta(q_0,1,A) = (q_0,\varepsilon) \qquad \delta(q_1,\varepsilon,Z) = (p,Z)$$

3.9

Slovo ucv ($u,v \in \{a,b\}^*$) patří do jazyka L právě když je splněna aspoň jedna z následujících podmínek :

1) $|u|<|v|$
2) $|u|>|v|$
3) existuje $i \leq \min(|u|,|v|)$ takové, že i-tý symbol řetězce u se liší od i-tého symbolu řetězce v .

Navržený automat přijímá slovo, jestliže nedeterministicky ověří jednu z uvedených podmínek. Stavová množina automatu je $q_0,q_1,\bar{q}_1,q_2,\bar{q}_2,q_3,q_a,\bar{q}_a,q_b,\bar{q}_b,p$, počáteční stav je q_0, koncové stavy p, $\bar{q}_2$. Vstupní abeceda je $\{a,b,c\}$, zásobníková abeceda $\{Z,A\}$, počáteční zásobníkový symbol je Z. Přechodová funkce je definována takto:

- větvení do 3 základních větví výpočtu:
  $\delta(q_0,\varepsilon,Z) = \{(q_1,Z),(q_2,Z),(q_3,Z)\}$,
- verifikace 1. podmínky:
  $\delta(q_1,a,Z) = \delta(q_1,b,Z) = \{(q_1,AZ)\}$,
  $\delta(q_1,a,A) = \delta(q_1,b,A) = \{(q_1,AA)\}$,
  $\delta(q_1,c,Z) = \{(\bar{q}_1,Z)\}$,
  $\delta(q_1,c,A) = \{(\bar{q}_1,A)\}$,
  $\delta(\bar{q}_1,a,A) = \delta(\bar{q}_1,b,A) = \{(\bar{q}_1,\varepsilon)\}$,
  $\delta(\bar{q}_1,a,Z) = \delta(\bar{q}_1,b,Z) = \{(p,Z)\}$,
- verifikace 2. podmínky:
  $\delta(q_2,a,Z) = \delta(q_2,b,Z) = \{(q_2,AZ)\}$,
  $\delta(q_2,a,A) = \delta(q_2,b,A) = \{(q_2,AA)\}$,
  $\delta(q_2,c,A) = \{(\bar{q}_2,\varepsilon)\}$,
  $\delta(\bar{q}_2,a,A) = \delta(\bar{q}_2,b,A) = \{(q_2,\varepsilon)\}$,
- verifikace 3. podmínky (Automat nedeterministicky zvolí i-tý symbol slova u, který si zapamatuje. Přitom uloží do zásobníku i symbolů A. Poté se přesune k symbolu c, aniž mění obsah zásobníku. Pomocí zásobníku pak vyhledá i-tý symbol slova v a srovná ho se zapamatovaným symbolem.) :
  $\delta(q_3,a,Z) = \{(q_3,AZ),(q_a,Z)\}$,
  $\delta(q_3,b,Z) = \{(q_3,AZ),(q_b,Z)\}$,
  $\delta(q_3,a,A) = \{(q_3,AA),(q_a,A)\}$,
  $\delta(q_3,b,A) = \{(q_3,AA),(q_b,A)\}$,
  $\delta(q_a,c,Z) = \{(\bar{q}_a,Z)\}$,
  $\delta(q_b,c,Z) = \{(\bar{q}_b,Z)\}$,
  $\delta(q_a,c,A) = \{(\bar{q}_a,A)\}$,
  $\delta(q_b,c,A) = \{(\bar{q}_b,A)\}$
  $\delta(q_a,a,A) = \delta(q_a,b,A) = \{(q_a,A)\}$, $\delta(q_b,a,A) = \delta(q_b,b,A) = \{(q_b,A)\}$,
  $\delta(\bar{q}_a,a,A) = \delta(\bar{q}_a,b,A) = \{(\bar{q}_a,\varepsilon)\}$,
  $\delta(\bar{q}_b,a,A) = \delta(\bar{q}_b,b,A) = \{(\bar{q}_b,\varepsilon)\}$,
  $\delta(\bar{q}_a,a,Z) = \{(p,Z)\}$,
  $\delta(\bar{q}_b,b,Z) = \{(p,Z)\}$
- doběh do konce vstupního řetězce v koncovém stavu:
  $\delta(p,a,Z) = \delta(p,b,Z) = \{(p,Z)\}$.

3.10

$$S \longrightarrow \langle q_0,A,q_0\rangle \mid \langle q_0,A,q_1\rangle \mid \langle q_0,A,q_2\rangle$$

$$\langle q_0,A,q_0\rangle \longrightarrow a\langle q_1,A,q_0\rangle\langle q_0,A,q_0\rangle \mid a\langle q_1,A,q_1\rangle\langle q_1,A,q_0\rangle \mid a\langle q_1,A,q_2\rangle\langle q_2,A,q_0\rangle \mid a\langle q_0,B,q_0\rangle ,$$

$$\langle q_0,A,q_1\rangle \longrightarrow a\langle q_1,A,q_0\rangle\langle q_0,A,q_1\rangle \mid a\langle q_1,A,q_1\rangle\langle q_1,A,q_1\rangle \mid a\langle q_1,A,q_2\rangle\langle q_2,A,q_1\rangle \mid a\langle q_0,B,q_1\rangle ,$$

$\langle q_0,A,q_2\rangle \longrightarrow a\langle q_1,A,q_0\rangle\langle q_0,A,q_2\rangle \mid a\langle q_1,A,q_1\rangle\langle q_1,A,q_2\rangle \mid a\langle q_1,A,q_2\rangle\langle q_2,A,q_2\rangle \mid a\langle q_0,B,q_2\rangle$,

$\langle q_1,A,q_0\rangle \longrightarrow b\langle q_1,A,q_0\rangle\langle q_0,A,q_0\rangle \mid b\langle q_1,A,q_1\rangle\langle q_1,A,q_0\rangle \mid b\langle q_1,A,q_2\rangle\langle q_2,A,q_0\rangle$

$\langle q_1,A,q_1\rangle \longrightarrow b\langle q_1,A,q_0\rangle\langle q_0,A,q_1\rangle \mid b\langle q_1,A,q_1\rangle\langle q_1,A,q_1\rangle \mid b\langle q_1,A,q_1\rangle\langle q_1,A,q_1\rangle$

$\langle q_1,A,q_2\rangle \longrightarrow b\langle q_1,A,q_0\rangle\langle q_0,A,q_2\rangle \mid b\langle q_1,A,q_1\rangle\langle q_1,A,q_2\rangle \mid b\langle q_1,A,q_2\rangle\langle q_1,A,q_2\rangle$

$\langle q_0,B,q_0\rangle \longrightarrow \langle q_1,A,q_0\rangle$
$\langle q_0,B,q_1\rangle \longrightarrow \langle q_1,A,q_1\rangle$
$\langle q_0,B,q_2\rangle \longrightarrow \langle q_1,A,q_2\rangle$

$\langle q_1,A,q_2\rangle \longrightarrow \varepsilon$

$\langle q_2,A,q_0\rangle \longrightarrow a\langle q_2,A,q_0\rangle$
$\langle q_2,A,q_1\rangle \longrightarrow a\langle q_2,A,q_1\rangle$
$\langle q_2,A,q_2\rangle \longrightarrow a\langle q_2,A,q_2\rangle$

$\langle q_2,A,q_2\rangle \longrightarrow b$

Redukcí získáme gramatiku

$S \longrightarrow \langle q_0,A,q_2\rangle$

$\langle q_0,A,q_2\rangle \longrightarrow a\langle q_1,A,q_2\rangle\langle q_2,A,q_2\rangle \mid a\langle q_0,B,q_2\rangle$
$\langle q_1,A,q_2\rangle \longrightarrow b\langle q_1,A,q_2\rangle\langle q_2,A,q_2\rangle$
$\langle q_0,B,q_2\rangle \longrightarrow \langle q_1,A,q_2\rangle$
$\langle q_1,A,q_2\rangle \longrightarrow \varepsilon$
$\langle q_2,A,q_2\rangle \longrightarrow b$.

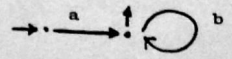
Jestliže pro lepší přehlednost přejmenujeme neterminály, získáme gramatiku

$S \longrightarrow A$
$A \longrightarrow aBC \mid aD$
$B \longrightarrow bBC$
$D \longrightarrow B$
$B \longrightarrow \varepsilon$
$C \longrightarrow b$,

a po jednoduchých úpravách gramatiku

$S \longrightarrow aBb \mid aB$
$B \longrightarrow bBb \mid \varepsilon$.

Tato gramatika zřejmě generuje jazyk ab^*, takže zadaný zásobníkový automat by bylo možno nahradit ekvivalentním konečným automatem



3.11

a) Převedením gramatiky na nevypouštějící získáme gramatiku

$S \longrightarrow A$
$S \longrightarrow OSA \mid OA$
$A \longrightarrow 1A$
$A \longrightarrow 1$
$A \longrightarrow B1$
$B \longrightarrow OB \mid O$.

b) Odstraněním "řetězových" pravidel tvaru $X \longrightarrow Y$ dostaneme gramatiku

$S \longrightarrow OSA \mid OA \mid 1A \mid 1 \mid B1$
$A \longrightarrow 1A \mid 1 \mid B1$
$B \longrightarrow OB \mid O$
$S \longrightarrow 1A \mid 1 \mid B1$

c) Z pravidel s pravou stranou délky aspoň 2 odstraníme terminály zavedením nových neterminálů Z_1 a Z_2 a dostáváme gramatiku

$S \longrightarrow Z_1SA \mid Z_1A \mid Z_2A \mid 1 \mid BZ_2$
$Z_1 \longrightarrow O$
$Z_2 \longrightarrow 1$, $A \longrightarrow Z_2A \mid 1 \mid BZ_2$
$B \longrightarrow Z_1B \mid O$.

d) Pravidla délky 3 nahradíme soustavou 2 pravidel a dostáváme gramatiku v Chomského normální formě:

$S \longrightarrow Z_1D_1 \mid Z_1A \mid Z_2A \mid BZ_2 \mid 1$
$D_1 \longrightarrow SA$, $A \longrightarrow Z_2A \mid 1 \mid BZ_2$
$Z_1 \longrightarrow O$
$Z_2 \longrightarrow 1$
$B \longrightarrow Z_1B \mid O$

3.12

1) $S \longrightarrow ND_1 \mid ND_2$
$D_1 \longrightarrow AD_2$
$D_2 \longrightarrow JD_3$
$D_3 \longrightarrow ND_4 \mid ND_5$
$D_4 \longrightarrow BD_5$
$D_5 \longrightarrow JJ$

$A \longrightarrow ND_6 \mid NJ$
$D_6 \longrightarrow AJ$
$B \longrightarrow ND_4 \mid ND_5$
$N \longrightarrow 0$
$J \longrightarrow 1$

2) $S \longrightarrow Z_LD_1$
$D_1 \longrightarrow EZ_P$
$E \longrightarrow FD_2 \mid FD_3$
$D_2 \longrightarrow PF$
$D_3 \longrightarrow KF$

$F \longrightarrow a \mid Z_LD_1$
$Z_L \longrightarrow ($
$Z_P \longrightarrow)$
$P \longrightarrow +$
$K \longrightarrow *$

3) Gramatiku zredukujeme a pak převedeme na gramatiku

$S \longrightarrow Z_1D_1 \mid a$
$D_1 \longrightarrow Z_2S$
$Z_1 \longrightarrow a$
$Z_2 \longrightarrow b$.

3.13

Výsledek může vypadat takto:

$A \longrightarrow BA'$

$A' \longrightarrow +BA' \mid \varepsilon$

$B \longrightarrow CB'$

$B' \longrightarrow *CB' \mid \varepsilon$

$C \longrightarrow (A) \mid a$.

3.14

Nechť $w \in L(G)$. Na základě levého odvození řetězce w podle gramatiky G zkonstruujeme odvození (ne nutně levé) řetězce w podle gramatiky G´takto:

Jakmile se objeví v levém odvození řetězce w podle gramatiky G úsek tvaru

$$(*)\quad \gamma_1 A \gamma_2 \Rightarrow \gamma_1 A \alpha_{j_1} \gamma_2 \Rightarrow \gamma_1 A \alpha_{j_1} \alpha_{j_2} \gamma_2 \Rightarrow \dots \Rightarrow \gamma_1 A \alpha_{j_1} \dots \alpha_{j_2} \alpha_{j_1} \gamma_2 \Rightarrow \gamma_1 \beta_i \alpha_{j_1} \dots \alpha_{j_2} \alpha_{j_1} \gamma_2$$

nahradíme ho úsekem

$$\gamma_1 A \gamma_2 \Rightarrow \gamma_1 \beta_i Z \gamma_2 \Rightarrow \gamma_1 \beta_i \alpha_{j_1} Z \gamma_2 \Rightarrow \dots \Rightarrow \gamma_1 \beta_i \alpha_{j_1} \dots \alpha_{j_2} Z \gamma_2 \Rightarrow \gamma_1 \beta_i \alpha_{j_1} \dots \alpha_{j_2} \alpha_{j_1} \gamma_2 .$$

Jelikož se jednalo o levé odvození, musí být každé použití A-pravidla obsaženo v úseku odvození tvaru(*).

Proto popsaným způsobem přeměníme každé levé odvození řetězce w podle gramatiky G na odvození téhož řetězce podle gramatiky G´. Odtud plyne, že $L(G) \subseteq L(G')$.

Předpokládejme nyní obráceně, že $w \in L(G')$. Existuje tedy levé odvození řetězce w podle G´. Nevyskytne-li se v tomto odvození symbol Z , je toto odvození také odvozením podle G. Soustředíme se tedy na případ, že se během odvození objeví symbol Z. To se nutně stane pravidlem tvaru $A \longrightarrow \beta_i Z$. Řetězec β_i může obsahovat neterminály a v tom případě levé odvození přepisuje nejprve tyto proměnné. My v tomto případě přeskupíme pořadí kroků v odvození , přepisování řetězce β_i odsuneme na později a nejprve použijeme pravidla přepisujícího Z .

Pokud je tvaru $Z \longrightarrow \alpha_j Z$, opět přednostně aplikujeme pravidlo přepisující Z atd., dokud Z nezmizí. Pak teprve dokončíme odvození řetězce β_i atd. Tím způsobem přeměníme pořadí v odvození tak, že všechna pravidla přepisující Z se ocitnou vždy v úsecích tvaru

$$\gamma_1 A \gamma_2 \Rightarrow \gamma_1 \beta_i Z \gamma_2 \Rightarrow \gamma_1 \beta_i \alpha_{j_1} Z \gamma_2 \Rightarrow \dots \gamma_1 \beta_i \alpha_{j_1} \dots \alpha_{j_2} Z \gamma_2 \Rightarrow \gamma_1 \beta_i \alpha_{j_1} \dots \alpha_{j_2} \alpha_{j_1} \gamma_2 .$$

Každý takový úsek nahradíme úsekem

$$\gamma_1 A \gamma_2 \Rightarrow \gamma_1 A \alpha_{j_1} \gamma_2 \Rightarrow \gamma_1 A \alpha_{j_2} \alpha_{j_1} \gamma_2 \Rightarrow \dots \Rightarrow \gamma_1 A \alpha_{j_1} \dots \alpha_{j_2} \gamma_2 \Rightarrow \gamma_1 \beta_i \alpha_{j_1} \dots \alpha_{j_2} \gamma_2 .$$

Výsledné odvození je odvozením podle gramatiky G . Tedy $L(G') \subseteq L(G)$.

3.15

Použijeme obecného postupu, který lze aplikovat na každou bezkontextovou gramatiku. Zadaná gramatika je v Chomského normální formě . Pro náš postup stačí, když má gramatika pouze pravidla tvaru

$X \longrightarrow a$, kde X je neterminál a a je terminál, nebo

$X \longrightarrow Y_1 \dots Y_k$, $k \geq 2$, kde $X, Y_1, \dots, Y_k$ jsou neterminály. Do tohoto tvaru můžeme každou bezkontextovou gramatiku převést známým způsobem (viz počáteční fázi převodu do Chomského normální formy).

Nejprve neterminály gramatiky lineárně uspořádáme. V našem příkladu můžeme pro větší názornost nahradit neterminály A,B,C číslovanými neterminály A_1, A_2, A_3 .

Dostaneme tak gramatiku

$A_1 \longrightarrow A_2 A_1$ | $A_3 \longrightarrow A_1 A_3$

$A_1 \longrightarrow A_2 A_3$ | $A_2 \longrightarrow 0$

$A_2 \longrightarrow A_1 A_2$ | $A_3 \longrightarrow 1$.

Nyní budeme gramatiku transformovat tak, aby se pravidlo $A_i \longrightarrow A_j \alpha$ objevilo v soustavě pravidel pouze když $j > i$. Začínáme od neterminálu A_1 a postupujeme podle vzrůstajících indexů.

U A_1-pravidel jsou nepřijatelná pouze pravidla tvaru $A_1 \longrightarrow A_1 \alpha$ a zbavili bychom se jich postupem z cvičení 3.14. V naší gramatice se však taková pravidla nevyskytují , pravé strany všech A_1-pravidel začínají neterminálem s vyšším indexem. Začneme upravovat až pravidlo $A_2 \longrightarrow A_1 A_2$.

Za A_1 do něho dosadíme pravé strany A_1-pravidel. Dostaneme tak z něho pravidla $A_2 \longrightarrow A_2 A_1 A_2$, $A_2 \longrightarrow A_2 A_3 A_2$.

Celá soustava pravidel je nyní

$A_1 \longrightarrow A_2 A_1$ | $A_3 \longrightarrow A_1 A_3$

$A_1 \longrightarrow A_2 A_3$ | $A_2 \longrightarrow 0$

$A_2 \longrightarrow A_2 A_1 A_2$ | $A_3 \longrightarrow 1$.

$A_2 \longrightarrow A_2 A_3 A_2$

A_2-pravidla upravíme podle postupu z příkladu 3.14 , přičemž přidáme nový neterminál Z_2. Místo dosavadních A_2-pravidel tedy mezi pravidla gramatiky zahrneme soustavu pravidel

$A_2 \longrightarrow 0$ | $Z_2 \longrightarrow A_1 A_2$ | $Z_2 \longrightarrow A_3 A_2$

$A_2 \longrightarrow 0Z$ | $Z_2 \longrightarrow A_1 A_2 Z_2$ | $Z_2 \longrightarrow A_3 A_2 Z_2$.

Celá soustava pravidel je nyní

$A_1 \longrightarrow A_2 A_1$ | $Z_2 \longrightarrow A_1 A_2 Z_2$

$A_1 \longrightarrow A_2 A_3$ | $Z_2 \longrightarrow A_3 A_2$

$A_2 \longrightarrow 0$ | $Z_2 \longrightarrow A_3 A_2 Z_2$

$A_2 \longrightarrow 0Z_2$ | $A_3 \longrightarrow A_1 A_3$

$Z_2 \longrightarrow A_1 A_2$ | $A_3 \longrightarrow 1$.

Nyní upravíme A_3-pravidlo $A_3 \longrightarrow A_1 A_3$. Nahradíme je soustavou

$A_3 \longrightarrow A_2 A_1 A_3$

$A_3 \longrightarrow A_2 A_3 A_3$

vzniklou dosazením za A_1. Tato pravidla je třeba dále nahradit dosazením za neterminál A_2 soustavou

$A_3 \longrightarrow 0A_1 A_3$ | $A_3 \longrightarrow 0A_3 A_3$

$A_3 \longrightarrow 0Z_2 A_1 A_3$ | $A_3 \longrightarrow 0Z_2 A_3 A_3$.

Množina pravidel gramatiky tedy nyní vypadá takto:

$A_1 \longrightarrow A_2 A_1$ | $A_3 \longrightarrow 0A_1 A_3$

$A_1 \longrightarrow A_2 A_3$ | $A_3 \longrightarrow 0Z_2 A_1 A_3$

$A_2 \longrightarrow 0$ | $A_3 \longrightarrow 0A_3 A_3$

$A_2 \longrightarrow 0Z_2$ | $A_3 \longrightarrow 0Z_2 A_3 A_3$

$Z_2 \longrightarrow A_1 A_2$ | $A_3 \longrightarrow 1$

$Z_2 \longrightarrow A_1 A_2 Z_2$

$Z_2 \longrightarrow A_3A_2$
$Z_2 \longrightarrow A_3A_2Z_2$

V tuto chvíli již platí , že výskyt pravidla $A_i \longrightarrow A_j$ implikuje $i < j$. Všechna A_3- i A_2-pravidla nyní mají na pravé straně slovo, které začíná terminálem. Uvědomte si, že v obecném případě tuto vlastnost mají všechna A_i-pravidla, kde i je nejvyšší index. Toho využijeme k tomu, abychom také A_1-pravidla modifikovali tak, aby jejich pravá strana začínala terminálem. Z pravidla $A_1 \longrightarrow A_2A_1$ dostaneme $A_1 \longrightarrow 0A_1$, $A_1 \longrightarrow 0Z_2A_1$. Z pravidla $A_1 \longrightarrow A_2A_3$ dostaneme $A_1 \longrightarrow 0A_3$, $A_1 \longrightarrow 0Z_2A_3$.

Nyní soustava pravidel vypadá takto:

$A_1 \longrightarrow 0A_1$
$A_1 \longrightarrow 0Z_2A_1$
$A_1 \longrightarrow 0A_3$
$A_1 \longrightarrow 0Z_2A_3$
$A_2 \longrightarrow 0$
$A_2 \longrightarrow 0Z_2$
$A_3 \longrightarrow 0A_1A_3$
$A_3 \longrightarrow 0Z_2A_1A_3$
$A_3 \longrightarrow 0A_3A_3$
$A_3 \longrightarrow 0Z_2A_3A_3$
$A_3 \longrightarrow 1$
$Z_2 \longrightarrow A_1A_2$
$Z_2 \longrightarrow A_1A_2Z_2$
$Z_2 \longrightarrow A_3A_2$
$Z_2 \longrightarrow A_3A_2Z_2$

První část pravidel už je tedy ve vhodném tvaru. Zbývá ještě upravit Z_2-pravidla.

Pravidlo $Z_2 \longrightarrow A_1A_2$ nahradíme soustavou pravidel

$Z_2 \longrightarrow 0A_1A_2$
$Z_2 \longrightarrow 0Z_2A_1A_2$
$Z_2 \longrightarrow 0A_3A_2$
$Z_2 \longrightarrow 0Z_2A_3A_2$.

Podobně pravidla $Z_2 \longrightarrow A_1A_2Z_2$, $Z_2 \longrightarrow A_3A_2$, $Z_2 \longrightarrow A_3A_2Z_2$ nahradíme po řadě soustavami:

$Z_2 \longrightarrow 0A_1A_2Z_2$, $Z_2 \longrightarrow 0Z_2A_1A_2Z_2$, $Z_2 \longrightarrow 0A_3A_2Z_2$, $Z_2 \longrightarrow 0Z_2A_3A_2Z_2$;
$Z_2 \longrightarrow 0A_1A_3A_2$, $Z_2 \longrightarrow 0Z_2A_1A_3A_2$, $Z_2 \longrightarrow 0A_3A_3A_2$, $Z_2 \longrightarrow 0Z_2A_3A_3A_2$,
$Z_2 \longrightarrow 1A_2$;

$Z_2 \longrightarrow 0A_1A_3A_2Z_2$; $Z_2 \longrightarrow 0Z_2A_1A_3A_2Z_2$, $Z_2 \longrightarrow 0A_3A_3A_2Z_2$,
$Z_2 \longrightarrow 0Z_2A_3A_3A_2Z_2$, $Z_2 \longrightarrow 1A_2Z_2$.

Konečný tvar množiny pravidel je tedy:

$A_1 \longrightarrow 0A_1$
$A_1 \longrightarrow 0A_3$
$A_2 \longrightarrow 0$
$A_3 \longrightarrow 0A_1A_3$
$A_3 \longrightarrow 0A_3A_3$
$A_3 \longrightarrow 1$
$Z_2 \longrightarrow 0A_1A_2$
$Z_2 \longrightarrow 0A_3A_2$
$Z_2 \longrightarrow 0A_1A_2Z_2$
$Z_2 \longrightarrow 0A_3A_2Z_2$
$Z_2 \longrightarrow 0A_1A_3A_2$
$Z_2 \longrightarrow 0A_3A_3A_2$
$Z_2 \longrightarrow 1A_2$
$Z_2 \longrightarrow 0Z_2A_1A_3A_2Z_2$
$Z_2 \longrightarrow 0Z_2A_3A_3A_2Z_2$
$A_1 \longrightarrow 0Z_2A_1$
$A_1 \longrightarrow 0Z_2A_3$
$A_2 \longrightarrow 0Z_2$
$A_3 \longrightarrow 0Z_2A_1A_3$
$A_3 \longrightarrow 0Z_2A_3A_3$
$Z_2 \longrightarrow 0Z_2A_1A_2$
$Z_2 \longrightarrow 0Z_2A_3A_2$
$Z_2 \longrightarrow 0Z_2A_1A_2Z_2$
$Z_2 \longrightarrow 0Z_2A_3A_2Z_2$
$Z_2 \longrightarrow 0Z_2A_1A_3A_2$
$Z_2 \longrightarrow 0Z_2A_3A_3A_2$
$Z_2 \longrightarrow 0A_1A_3A_2Z_2$
$Z_2 \longrightarrow 0A_3A_3A_2Z_2$
$Z_2 \longrightarrow 1A_2Z_2$.

(3.16)

Tvrzení dokážeme ve všech čtyřech případech sporem s lemmatem o vkládání. Předpokládáme, že daný jazyk bezkontextový je. Potom existují konstanty $p,q > 0$ splňující tvrzení lemmatu o vkládání. Probereme nyní postupně všechny čtyři příklady.

a) Zvolme slovo $z = 0^k1^k0^k$ pro nějaké $k > p/3$. Uvažujme libovolný rozklad $z = uvwxy$. Jestliže má být aspoň jedno ze slov v,x neprázdné a zároveň $uv^iwx^iy \in L_1$ pro všechna $i \geq 0$, musí být každé ze slov v,x tvořeno nejvýše jedním druhem symbolů.

Kdyby $v = 0^m, x = 1^n$ pro nějaká $m,n > 0$, bylo by slovo uv^2wx^2y tvaru $0^{r}1^{s}0^{t}$ pro nějaká $r,s > t$, tzn. toto slovo by nepatřilo do L_1. Podobně se vyloučí možnost $v = 1^m, x = 0^n$. Zbývá tedy pouze možnost , že v i x jsou tvořeny stejným druhem symbolů. Jestliže $v = 1^m, x = 1^n$ pro nějaká $m,n \geq 0$ a $m+n > 0$, potom slovo uv^2wx^2y je tvaru $0^r1^s0^r$ pro $0 \leq r < s$ a nepatří tedy do L_1.
Jestliže $v = 0^m, x = 0^n$ pro nějaká $m,n \geq 0$ a $m+n > 0$, potom řetězec uv^0wx^0y je tvaru $0^r1^s0^t$, kde $\min(r,t) < s$ a ani toto slovo do L_1 nepatří. Dospěli jsme tedy ke sporu s předpokladem, že jazyk L_1 je bezkontextový.

b) Zvolme slovo $z = 0^k1^k0^k1^k$ pro nějaké $k > p,q$ a jeho libovolný rozklad $z = uvwxy$. Jestliže $vx \neq \varepsilon$ a slova uv^iwx^iy pro $i \geq 0$ mají patřit do L_2, potom žádné ze slov v,x nemůže obsahovat výskyt obou symbolů 0,1. Jelikož navíc můžeme předpokládat, že $|vwx| \leq q$, nemohou se řetězce v,x ve slově z vyskytovat dále od sebe než ve dvou sousedících blocích tvořených týmiž symboly. Potom ovšem slovo uv^2wx^2y nepatří do L_2, což je spor s tvrzením lemmatu o vkládání.

Důkaz zbývajících dvou příkladů už pouze v hlavních rysech:

c) Zvolíme $z = a^nb^nc^n$ pro nějaké $n > p/3$ a libovolný rozklad $z = uvwxy$ splňující podmínky lemmatu o vkládání. Podobně jako v předchozích případech nemůže žádné ze slov v,x obsahovat dva různé druhy symbolů. Jeden ze symbolů a,b,c tedy není obsažen ani ve slově v, ani ve slově x. Jestliže tímto symbolem je a, potom slovo uv^0wx^0y je tvaru $a^nb^rc^s$ pro nějaká r,s taková, že $\min(r,s) < n$. Takové slovo ovšem nepatří do L_3. Jestliže ve slovech v,x není obsažen symbol c , potom v L_3 neleží řetězec uv^2wx^2y atd.

d) Zvolme $z = a^{n^2}$ pro $n > q/2$ a rozklad $z = uvwxy$ splňující podmínky lemmatu o vkládání. Potom pro $z_2 = uv^2wx^2y$ je $n^2 < |z_2| < n^2+q < n^2+2n+1 = (n+1)^2$. Proto $z_2 \notin L_4$ - spor.

(3.17)

Z důkazu lemmatu o vkládání je vidět, že pro každou bezkontextovou gramatiku G lze příslušná čísla m,n 0 nalézt algoritmicky Není také těžké se přesvědčit, že otázka, zda dané slovo patří do L(G), je algoritmicky rozhodnutelná.
Ukážeme, že jazyk L(G) je nekonečný právě když v L(G) leží slovo z takové, že $p < |z| < p+m$, kde $p = \max(m,n)$.
Tuto otázku lze algoritmicky rozhodnout probráním všech slov, jejichž délka je v rozmezí od p do p+m. Dokažme tedy uvažovanou ekvivalenci.

a) Nechť existuje slovo z L(G) takové, že $p < |z| < p+m$. Potom z lze psát ve tvaru $z = uvwxy$ tak, že všechna slova uv^iwx^iy, $i \geq 0$ patří do L(G). Jelikož $vx \neq \delta$ tvoří tato slova nekonečnou podmnožinu jazyka L(G).

b) Nechť L(G) je nekonečný jazyk. Pak v L(G) existuje slovo délky větší než p. Nechť z je takové slovo minimální délky. Ukážeme, že $p < |z| < p+m$, čímž bude důkaz ukončen.

Předpokládejme opak, tzn. že $|z|>p+m$. Řetězec z lze psát ve tvaru z = uvwxy tak, že $vx \neq \varepsilon$, $uwy \in L$ a $|vwx| \leq m$. Potom ovšem $p < |uwy| < |z|$, což je spor s volbou z.

3.18

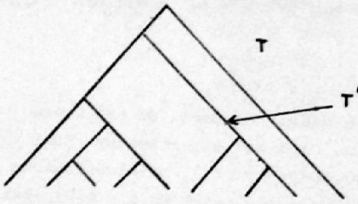
Mějme libovolnou lineární gramatiku $G = (\Pi, \Sigma, S, P)$. Bez újmy na obecnosti můžeme předpokládat, že gramatika G je nevypouštějící a neobsahuje pravidla tvaru $X \longrightarrow Y$. Označme $k = \max \{ |\alpha|, X \longrightarrow \alpha \in P \}$.
Každým odvozením v gramatice G, které má délku $d \geq 0$, lze vytvořit slovo délky nejvýše $(d-1)(k-1) + k = d(k-1) + 1$.
Nechť s označuje počet neterminálů gramatiky G. Položme $p = s(k-1) + 2$.
Nechť $z \in L$ délky alespoň p a nechť $S = X_0 \Rightarrow u_1 X_1 v_1 \Rightarrow$
$\Rightarrow u_1 u_2 X_2 v_2 v_1 \Rightarrow u_1 u_2 \dots u_d t v_d \dots v_2 v_1 = z$ je jeho libovolné odvození. Z předchozích úvah plyne, že $d > s$. Proto musí být $X_i = X_j$ pro nějaké $0 \leq i < j \leq d-1$.
Zvolme nejmenší j takové, že $X_i = X_j$ pro nějaké $i < j$. Zřejmě je $j \leq s$.
Položme $u = u_1 \dots u_i$, $v = u_{i+1} \dots u_j$, $w = u_{j+1} \dots u_d t v_d \dots v_{j+1}$, $x = v_j \dots v_{i+1}$, $y = v_i \dots v_1$. Potom $|uvxy| \leq j.(k-1) < p$.
Dále aspoň jeden z řetězců v,x je neprázdný, což plyne z nepřítomnosti pravidel $X \longrightarrow Y$. Konečně $S \Rightarrow^* uX_i y \Rightarrow^* uvX_j xy = uvX_i xy \Rightarrow^* uvwxy$. Z toho plyne, že $S \Rightarrow^* uv^l X_j x^l y \Rightarrow^* uv^l wx^l y$, pro každé $l \geq 0$.

3.19

Předpokládejme, že uvedený jazyk L je lineární. Potom existuje číslo p s vlastnostmi popsanými v lemmatu z příkladu 3.18. Zvolme slovo $z = 0^p 1^p 0^p 1^p \in L$. Zvolíme-li libovolný rozklad z = uvwxy takový, že $|uvxy| \leq p$, obsahuje slovo w celý řetězec $1^p 0^p$ jako vlastní podslovo. Jestliže $vx \neq \varepsilon$, má slovo $z_2 = uv^2wx^2y$ nutně tvar $0^{p+r} 1^p 0^p 1^{p+s}$ pro nějaké $r,s \geq 0$ takové, že $\max(r,s) > 0$. Pak ovšem $z_2 \notin L$. Tím jsme dospěli ke sporu s uvedeným lemmatem.

3.20

Návod k řešení. Uvažujme derivační strom T příslušný k řetězci z. Cesty vedené od označených písmen ve slově z ke kořeni stromu T tvoří jistý podstrom T′ stromu T (viz obr.).



Jestliže je ve slově z dostatečný počet označených symbolů, obsahuje strom T′ tak dlouhou cestu, že dva vrcholy (s větvením) ležící na této cestě jsou ohodnoceny stejným neterminálem...
A dále úvahy pokračují analogicky jako v důkazu lemmatu o vkládání.

3.21

Užijeme lemmatu z příkladu 3.20. Nechť L je bezkontextový. Pak existují čísla m,n s vlastnostmi popsanými zmíněným lemmatem. Zvolme například $z = a^{n+1} b^{n+1+m!} c^{n+1+2(m!)}$. Označme v tomto slově všechny výskyty symbolu a. Slovo z patří do L a počet označených symbolů je větší než n. Zvolíme-li libovolnou dekompozici z = uvwxy takovou, že $uv^2wx^2y \in L$, je každý z řetězců v,x tvořen nejvýše jedním druhem symbolů. Pokud navíc v nebo x má obsahovat označený symbol, přibývá ve slovech $uv^i wx^i y$ $(i > 0)$ symbolů a a dále nejvýše jeden další druh symbolů. Jestliže počet symbolů a v řetězci vwx je omezen konstantou m, přibývá vždy při přechodu od $uv^i wx^i y$ k $uv^{i+1} wx^{i+1} y$ p symbolů a pro jistou konstantu $p \leq n$.
Každá taková konstanta k však dělí číslo m! i 2(m!). Proto pro jisté i bude ve slově $z_i = uv^i wx^i y$ počet symbolů a roven počtu symbolů b nebo počtu symbolů c. Pak ovšem $z_i \notin L$ - spor.

3.22

Mějme jazyky L_1, L_2 generované po řadě bezkontextovými gramatikami $G_1 = (\Pi_1, \Sigma_1, S_1, P_1)$ a $G_2 = (\Pi_2, \Sigma_2, S_2, P_2)$.
Případným přejmenováním neterminálů můžeme docílit toho, že $\Pi_1 \cap \Pi_2 = \Pi_1 \cap \Sigma_2 = \Sigma_1 \cap \Pi_2 = \emptyset$.

a) Jazyk $L_1 \cup L_2$ je generován gramatikou $(\Pi_1 \cup \Pi_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, S, P_1 \cup P_2 \cup \{S \longrightarrow S_1, S \longrightarrow S_2\})$ kde S je nově přidaný neterminál.
b) Jazyk $L_1 . L_2$ je generován gramatikou $(\Pi_1 \cup \Pi_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, S, P_1 \cup P_2 \cup \{S \longrightarrow S_1 S_2\})$.
c) Jazyk L_1^* je generován gramatikou $(\Pi_1 \cup \{S\}, \Sigma_1, S, P_1 \cup \{S \longrightarrow S_1 S, S \longrightarrow \varepsilon\})$.
d) Jazyk L_1^R je generován gramatikou $(\Pi_1, \Sigma_1, S, P^R)$, kde $P^R = \{X \longrightarrow \alpha^R, X \longrightarrow \alpha \in P\}$.
e) Budiž $\delta : \Sigma^* \longrightarrow \Delta^*$ substituce taková, že pro každé $a \in \Sigma$ je $\delta(a)$ bezkontextový jazyk. Nechť $L \subseteq \Sigma^*$ je jazyk generovaný bezkontextovou gramatikou $G = (\Pi, \Sigma, S, P)$, nechť dále $\Sigma = \{a_1, \dots, a_n\}$ a pro každé $1 \leq i \leq n$ je jazyk $\delta(a_i)$ generován bezkontextovou gramatikou $G_i = (\Pi_i, \Sigma_i, S_i, P_i)$.
Přejmenováním neterminálů můžeme docílit toho, že množiny neterminálů $\Pi_1, \dots \Pi_n$ jsou vzájemně disjunktní a že žádný terminál libovolné gramatiky G_i není pro jinou z těchto gramatik neterminálem. Potom jazyk $\delta(L)$ je generován bezkontextovou gramatikou $(\bigcup_1 \Pi_i, \bigcup_1 \Sigma_i, S, \bigcup_1 P_i \cup \hat{P})$, kde $\hat{P}$ je množina pravidel vzniklých z pravidel z množiny P nahrazením každého výskytu terminálu a_i neterminálem S_i (pro každé $1 \leq i \leq n$).
f) je přímým důsledkem e).

3.23

Třída bezkontextových jazyků je podle 3.22 uzavřena vůči sjednocení, iteraci i homomorfismu. Podle věty 4.5.10 v [2] je uzavřena i vůči průniku s regulárními jazyky. Zbývá tedy dokázat pouze její uzavřenost vůči inverznímu homomorfismu.
Předpokládejme, že bezkontextový jazyk L je rozpoznáván (koncovým stavem) zásobníkovým automatem $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$. Nechť dále $h: \Delta^* \longrightarrow \Sigma^*$ je libovolný homomorfismus, kde $\Delta = \{a_1, \dots, a_n\}$ je libovolná abeceda. Označme pro každé $a_i \in \Delta$:

$h(a_i) = b_{i1} \dots b_{ik_i}$, kde $k_i \geq 0$, $b_{ij} \in \Sigma$. Speciálně $k_i = 0$, jestliže $h(a_i) = \varepsilon$.

Popíšeme nyní zásobníkový automat M′, který rozpoznává jazyk $h^{-1}(L)$. Tento auto-

mat (nedeterministicky) rozpoznává ve vstupním řetězci podslova tvaru $b_{i1}\dots b_{ik_i}$ a reaguje na ně stejně, jako M reaguje na symbol a_i. Přesný popis vypadá takto:

$M' = (Q', \Sigma, \Gamma, \delta', q_0', Z_0, F')$, kde

$Q' = \{(q,i,j);\ q \in Q,\ 1 \leq i \leq n,\ 0 \leq j \leq i_k\} \cup \{q_0'\}$,

$F' = \{(q,i,k_i),\ q \in F,\ 1 \leq i \leq n$ a přechodová funkce je definována takto:

$\delta'(q_0', \varepsilon, Z_0) = \{((q_0,i,0),Z_0);\ 1 \leq i \leq n\}$,

$\delta'((q,i,j),b_{i,j+1},Z) = \{((q,i,j+1),Z)\},\ q \in Q,\ 1 \leq i \leq n,\ 0 \leq j < k_i$.

$\delta'((q,i,k_i),\varepsilon,Z) = \{(p,j,0),\gamma);(p,\gamma) \in \delta(q,a_i,Z),\ 1 \leq j \leq n\}$, $q \in Q,\ 1 \leq i \leq n$.

Dokazované tvrzení plyne i z 4.8.13 v [2] .

(3.24)

Nechť $\mathcal{L}$ je libovolná plná abstraktní třída jazyků.

a) Buďte $L_1, L_2 \in \mathcal{L}$, $L_1, L_2 \subseteq \Sigma^*$. K abecedě Σ sestrojme dvě abecedy dvojníků: $\bar{\Sigma} = \{\bar{x};\ x \in \Sigma\}$, $\hat{\Sigma} = \{\hat{x};\ x \in \Sigma\}$. Položme $\Delta = \Sigma \cup \bar{\Sigma} \cup \hat{\Sigma}$ a definujme homomorfismus $h: \Delta^* \longrightarrow \Sigma^*$ předpisem $h(x) = h(\bar{x}) = h(\hat{x}) = x$, pro všechna $x \in \Sigma$.

Vytvořme jazyky

$\bar{L}_1 = h^{-1}(L_1) \cap \{\bar{x}w;\ w \in \Sigma^*, \bar{x} \in \bar{\Sigma}\}$

$\hat{L}_2 = h^{-1}(L_2) \cap \{\hat{x}w,\ w \in \Sigma^*, \hat{x} \in \hat{\Sigma}\}$,

tj. jazyky $\bar{L}_1$ a $\hat{L}_2$ vzniknou z L_1 a L_2 tak, že se v každém neprázdném slově nahradí první písmeno příslušným dvojníkem. Vzhledem k uzavřenosti $\mathcal{L}$ vůči inverznímu homomorfismu a průniku s regulárními jazyky je $\bar{L}_1, \hat{L}_2 \in \mathcal{L}$.

Dále definujme

$L_3 = (\bar{L}_1 \cup \hat{L}_2)^+ \cap \{\bar{a}ub\hat{v};\ \bar{a} \in \bar{\Sigma}, \hat{b} \in \hat{\Sigma},\ u,v \in \Sigma^*\}$.

Z uzávěrových vlastností $\mathcal{L}$ opět plyne, že $L_3 \in \mathcal{L}$ a $h(L_3) \in \mathcal{L}$. Jelikož platí, že

$$L_1 . L_2 = \begin{cases} h(L_3) \\ h(L_3) \cup L_1 \\ h(L_3) \cup L_2 \\ h(L_3) \cup L_1 \cup L_2 \end{cases}$$

podle toho, zda jazyky L_1, L_2 obsahují ε, či nikoliv,

je v každém případě $L_1 . L_2 \in \mathcal{L}$.

b) Z uzavřenosti $\mathcal{L}$ vůči libovolnému homomorfismu je zřejmé, že $\{\varepsilon\} \in \mathcal{L}$. Tvrzení pak plyne z toho, že $L^* = L^+ \cup \{\varepsilon\}$ a uzavřenosti $\mathcal{L}$ vůči sjednocení a pozitivní iteraci.

c) Nechť $L \in \mathcal{L}$, $L \subseteq \Sigma^*$, $\Sigma = \{a_1, \dots, a_n\}$, δ nechť je libovolná regulární substituce:

$\delta(x_i) = R_i$, $R_i \subseteq \Sigma_i^*$, pro $1 \leq i \leq n$.

Sestrojme abecedu dvojníků $\bar{\Sigma} = \{\bar{x}_1, \dots, \bar{x}_n\}$ a definujme regulární jazyk $R = (\bigcup_1 R_i \bar{x}_i)^*$.

Dále definujme homomorfismus $h_1: (\bar{\Sigma} \cup \Sigma_1 \cup \dots \cup \Sigma_n)^* \longrightarrow \Sigma^*$ předpisem $h_1(\bar{x}_i) = x_i$ pro všechna $\bar{x}_i \in \bar{\Sigma}$ a $h_1(a) = \varepsilon$ pro všechna $a \in \Sigma_1 \cup \dots \cup \Sigma_n$.

Další homomorfismus $h_2: (\bar{\Sigma} \cup \Sigma_1 \cup \dots \cup \Sigma_n)^* \longrightarrow (\Sigma_1 \cup \dots \cup \Sigma_n)^*$ definujme předpisem

$$h_2(a) = \begin{cases} a, & \text{jestliže } a \in \Sigma_1 \cup \dots \cup \Sigma_n \\ \varepsilon, & \text{jestliže } a \in \bar{\Sigma} \end{cases}$$

Zřejmě platí, že

$$\delta(L) = h_2(h_1^{-1}(L) \cap R).$$

d) Nechť $S = (Q, \Sigma, \Delta, \delta, q_0, F)$ je libovolný zobecněný sekvenční stroj a nechť $L \in \mathcal{L}$. Můžeme předpokládat, že $L \subseteq \Sigma^*$, neboť $S(L) = S(L \cap \Sigma^*)$.

Zavedeme pomocnou abecedu

$Z = \{[q_i,a,w,q_j];\ (q_j,w) \in \delta(q_i,a)\}$.

Na Z definujme relaci r:

$r([q_i,a,w,q_j], [q_i',a',w',q_j']) \Leftrightarrow q_j = q_i'$.

Není těžké ověřit, že jazyk

$L_1 = \{z_1 \dots z_n \in Z^+;\ r(z_1, z_{i+1})$ pro všechna $1 \leq i \leq n-1\}$ je regulární. Jazyk $L_2 \subseteq L_1$ definovaný tak, že L_2 obsahuje právě všechna slova z L_1, jejichž první symbol začíná stavem q_0 a poslední symbol končí nějakým stavem q z F, je opět regulární. Konečně i jazyk

$$L_3 = \begin{cases} L_2, & \text{jestliže } q_0 \notin F \\ L_2 \cup \{\varepsilon\}, & \text{jestliže } q_0 \in F, \end{cases}$$

je regulární.

Definujme homomorfismy $h_1: Z^* \longrightarrow \Sigma^*$ a $h_2: Z \longrightarrow \Delta^*$ předpisem:

$h_1([q_i,a,w,q_j]) = a$, $h_2([q_i,a,w,q_j]) = w$.

Zřejmě platí, že

$$S(L) = h_2(h_1^{-1}(L) \cap L_3),$$ a tedy $S(L) \in \mathcal{L}$.

e) Definujme h_1, h_2 i L_3 stejně jako v d). Potom zřejmě platí, že

$$S^{-1}(L) = h_1(h_2^{-1}(L) \cap L_3).$$

f) Nechť $L \in \mathcal{L}$, R nechť je regulární jazyk a $L, R \subseteq \Sigma^*$. Dokážeme, že $R \setminus L \in \mathcal{L}$.

Zvolme libovolný symbol $c \notin \Sigma$ a sestrojme zobecněný sekvenční stroj S, který každé slovo typu $ucv \in \Sigma^* c \Sigma^*$ přepíše na v. Co udělá s ostatními slovy, není podstatné.

Definujme homomorfismus $h: (\Sigma \cup \{c\})^* \longrightarrow \Sigma^*$ předpisem:

$h(c) = \varepsilon$, $h(x) = x$ pro každé $x \in \Sigma$.

Potom $R \setminus L = S(h^{-1}(L) \cap R.\{c\}.\Sigma^*)$.

Zcela obdobně se ukáže, že $L / R \in \mathcal{L}$.

(3.25)

Důkaz provedeme sporem. Nechť pro libovolné dva bezkontextové jazyky L_1, L_2 je $L_1 \sqcup L_2$ opět bezkontextový jazyk. Zvolme $L_1 = \{a^n b^n;\ n \geq 1\}$, $L_2 = \{c^n d^n;\ n \geq 1\}$. Pak by i jazyk

$(L_1 \sqcup L_2) \cap [a^* c^* b^* d^*] = \{a^m c^n b^m d^n;\ m,n \geq 1\}$ měl být bezkontextový, což není.

4.1

a) výpočet nad řetězcem a*(a+a)

obsah zásobníku	symbol na vstupu	výstup
A	a	-
BA′	a	1
CB′A′	a	4
aB′A′	a	8
B′A′	*	-
*CB′A′	*	5
CB′A′	(	-
(A)B′A′	(	7
A)B′A′	a	-
BA′)B′A′	a	1
CB′A′)B′A′	a	4
aB′A′)B′A′	a	8
B′A′)B′A′	)	-
A′)B′A′	+	6
+BA′)B′A′	+	2
BA′)B′A′	a	-
CB′A′)B′A′	a	4
aB′A′)B′A′	a	8
B′A′)B′A′	)	-
A′)B′A′	)	6
)B′A′	)	3
B′A′	-	-
A′	-	6
-	-	3
		přijmout

b) výpočet nad řetězcem a(a+a)

obsah zásobníku	symbol na vstupu	výstup
A	a	-
BA′	a	1
CB′A′	a	4
aB′A′	a	8
B′A′	(	-
		chyba

c) výpočet nad řetězcem a+a

obsah zásobníku	symbol na vstupu	výstup
A	a	-
BA′	a	1
CB′A′	a	4
aB′A′	a	8
B′A′	+	-
A′	+	6
+BA′	+	2
BA′	a	-
CB′A′	a	4
aB′A′	a	8
B′A′	-	-
A′	-	6
-	-	3
		přijmout

4.2

	if	while	do	repeat	until	s	c	non	e
S	ifEthenS 1	whileEdoS 2	-	repeatSuntilE 3	-	s 4	-	-	-
E	-	-	-	-	-	-	c 5	none 6	-
if	krátit								
while		krátit							
do			kr						
repeat				krátit					
until					krátit				
s						k			
c							k		
non								kr	
e									při-jmout

4.3

Řídicí množiny pro prvních pět pravidel jsou patrné na první pohled. Pro 6. pravidlo je to zřejmě množina

$$D(S' \rightarrow \varepsilon) = \text{FOLLOW}(S') = \{\underline{\text{until}}, \underline{\text{end}}\}.$$

Je vidět, že se jedná o LL(1)-gramatiku. Její analyzátor vypadá takto:

	if	then	repeat	until	begin	end	s	c	;	e
S	if c then S 1		repeat SS′until c 2		begin SS′end 3		s 4			
S′				ε 6		ε 6			;SS′ 5	
if	krácení									
then		kr								
repeat			krácení							
until				krác						
begin					krácení					
end						k				
s							k			
c								k		
;									kr	
ε										při jm.

4.4

Přímo ze zadání gramatiky je vidět, že FIRST(B) = {b} a FOLLOW(A) = {b}. Odtud dostáváme

$D(S \rightarrow aAb) = \{a\}$ $\quad D(S \rightarrow BAb) = \{b\}$,
$D(A \rightarrow aS) = \{a\}$ $\quad D(A \rightarrow \varepsilon) = \{b\}$
$D(B \rightarrow bAb) = \{b\}$.

Gramatika je proto LL(1) a její LL(1)-analyzátor vypadá takto:

	a	b	e
S	aAb, 1	BAb, 2	
A	aS, 3	^, 4	
B		bAb, 5	
a	krátit		
b		krátit	
ε			přijmout

4.5

1) S pravidlem $X \rightarrow X\alpha$ musí gramatika obsahovat aspoň jedno pravidlo tvaru $X \rightarrow \beta$, kde $\beta \neq X\alpha$, jinak by neterminál X nemohl být přepsán na žádný terminální řetězec. Potom ovšem $\text{FIRST}(\beta) \subseteq \text{FIRST}(X)$.

Jelikož

$D(X \to X\alpha) = FIRST(X\alpha\, FOLLOW(X))$,

$D(X \to \beta) = FIRST(\beta\, FOLLOW(X))$ a

$FIRST(X) \subseteq FIRST(X\alpha)$,

dostáváme

$D(X \to \beta) = FIRST(\beta\, FOLLOW(X)) \subseteq FIRST(X\alpha\, FOLLOW(X)) = D(X \to X\alpha)$,

a proto

$\emptyset \neq D(X \to \beta) = D(X \to \beta) \cap D(X \to X\alpha)$,

tzn. gramatika není LL(1).

2) Jestliže v gramatice je možné odvození $X \Rightarrow^+ X\alpha$, potom pro každé pravidlo $X \to \beta$ je množina $FIRST(\beta) \subseteq FIRST(X)$.

Kdyby gramatika byla LL(1), mohla by pak obsahovat jediné X-pravidlo a tímto pravidlem začíná odvození $X \Rightarrow^+ X\alpha$. Toto pravidlo je proto nutně tvaru

$$X \to \gamma_1 Y \gamma_2 ,$$

kde $\gamma_1 \Rightarrow^* \varepsilon$ a $Y \Rightarrow^* X\omega$ pro nějaké ω. Odtud plyne, že

$$Y \Rightarrow^+ Y\gamma_2\omega .$$

Opakováním předchozí úvahy pro toto odvození zjistíme, že gramatika musí obsahovat jediné Y-pravidlo, kterým proto uvedené odvození začíná. Po konečném počtu kroků tak dojdeme ke sporu s předpokladem, že gramatika je redukovaná.

(4.6)

Zřejmě

$D(S \to BA) = \{a,b,c\}$, $D(B \to aA) = \{a\}$,

$D(A \to BS) = \{a,b,c\}$, $D(B \to bS) = \{b\}$,

$D(A \to d) = \{d\}$, $D(B \to c) = \{c\}$.

Gramatika je LL(1) a její analyzátor vypadá takto:

	a	b	c	d	e
S	BA, 1	BA, 1	BA, 1		
A	BS, 2	BS, 2	BS, 2	d,3	
B	aA, 4	bS, 5	c, 6		
a	krátit				
b		krátit			
c			krátit		
d				krátit	
ć					přijmout

(4.7)

Označme symbolem F funkci zkonstruovanou popsaným postupem. Ověříme, že F = FOLLOW. Přímo z definice funkce FOLLOW a relace $\rhd$ plyne, že pro každé neterminály X,Y platí

(*) $X \rhd Y \Rightarrow FOLLOW(X) \subseteq FOLLOW(Y)$.

Proto také

$X \sim Y \Rightarrow FOLLOW(X) = FOLLOW(Y)$.

Dále je zřejmé

(**) $FOL(X) \subseteq FOLLOW(X)$ prokaždý neterminál X. Z (*) a (**) plyne, že $F(X) \subseteq FOLLOW(X)$ pro každý neterminál X.

Zbývá dokázat opačnou implikaci. Nechť $a \in FOLLOW(X)$. Potom existuje levé odvození $S \Rightarrow^* uX\alpha$ takové, že $a \in FIRST(\alpha)$. Indukcí podle délky tohoto odvození ukážeme, že $a \in F(X)$. Podrobněji: pro každé $k \geq 0$ platí, že

(***) pokud $S \Rightarrow^* uX\alpha$ je odvozením délky k a $a \in FIRST(\alpha)$, potom $a \in F(X)$.

Pro $k = 0$ je $a = \varepsilon \in FOL(S) \subseteq F(S)$

pro $k = 1$ je $S \to uX\alpha \in P$, $a \in FOL(X) \subseteq F(X)$.

Předpokládejme nyní, že (***) je splněno, jestliže $k \leq n$, pro jisté n a mějme $k = n+1$. Jestliže $a \in FOL(X)$, jsme hotovi. Jestliže $a \notin FOL(X)$, je uvažované odvození možno rozepsat takto:

$S \Rightarrow^* vY\beta \Rightarrow v\gamma_1 X \gamma_2 B \Rightarrow^* uX\alpha$, kde $v\gamma_1 \Rightarrow^* u$, $\gamma_2\beta = \alpha$ a $Y \to \gamma_1 X \gamma_2 \in P$. Jelikož $a \notin First(\gamma_2)$ je $\gamma_2 \Rightarrow^* \varepsilon$ a tedy $Y \rhd X$. Proto je $a \in FOLLOW(Y)$.

Jelikož $S \Rightarrow^* uY\beta$ je odvození délky nejvýše n, je podle indukčního předpokladu $a \in F(Y)$, a proto také $a \in F(X)$.

(4.8)

a) konstrukce funkce FIRST

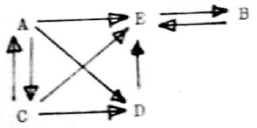
	A	B	C	D	E
1	-	ε	-	ε,p	q
2	a,p	ε,q	a	ε,p,q	p,q
3	a,p,q	ε,q,a,p	a,q	ε,p,q	p,q
4	a,p,q	ε,a,p,q	a,p,q	ε,p,q	a,p,q
5	a,p,q	ε,a,p,q	a,p,q	ε,a,p,q	a,p,q
6	a,p,q	ε,a,p,q	a,p,q	ε,a,p,q	a,p,q

$FIRST(A) = FIRST(C) = FIRST(E) = \{a,p,q\}$

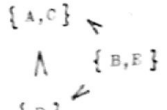
$FIRST(B) = FIRST(D) = \{\varepsilon,a,p,q\}$.

b) konstrukce funkce FOLLOW

graf relace $\rhd$:



graf uspořádání $<$:



$FOL(A) = \{\varepsilon,a,p,q,r\}$ $FOL(C) = \{a,b,p,q\}$

$FOLLOW(A) = FOLLOW(C) = \{\varepsilon,a,b,p,q,r\}$,

$FOL(D) = \{a\}$

$FOLLOW(D) = FOLLOW(A)$

$FOL(B) = \{a,p\}$ $FOL(E) = \{a,p,q,c\}$

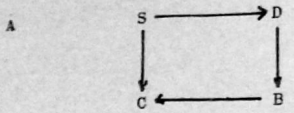
$FOLLOW(B) = FOLLOW(E) = \{\varepsilon,a,b,c,p,q,r\}$.

4.9

a) konstrukce funkce FIRST:

	S	A	B	C	D
1	a, ε	ε	f , ε	ε	a , ε
2	a, ε	a, ε	a,c,f, ε	a,g,h,ε	a, ε
3	a, ε	a,d,ε	a,c,d,f,ε	a,g,h,ε	a,ε

graf relace :



FOLLOW(A) = { a,c,d,f,b }
FOLLOW(S) = { a,c,d,g,ε }
FOLLOW(D) = { a,c,d,g,ε }
FOLLOW(B) = { a,b,c,d,g,ε }
FOLLOW(C) = { a,b,c,d,g,h,ε }

b)

D(S ⟶ aAAcCD) = {a}
D(S ⟶ ε) = {a,c,d,g,ε}
D(A ⟶ ASd) = {a,d}
D(A ⟶ ε) = {a,c,d,f,b}
D(B ⟶ SAc) = {a,c,d}
D(B ⟶ fC) = {f}
D(B ⟶ ε) = {a,b,c,d,g,ε}
D(C ⟶ Sg) = {a,g}
D(C ⟶ Ch) = {a,g,h}
D(D ⟶ aBD) = {a}
D(D ⟶ ε) = {a,c,d,g,ε}
D(C ⟶ ε) = {a,c,d,g,h,ε}

c) Gramatika není LL(1): neprázdný průnik mají řídicí množiny pravidel (1), (2), pravidel (3), (4), pravidel (5),(7), pravidel (8),(9),(10) a pravidel (11),(12).

4.10

a) Hodnoty funkce FIRST jsou patrné přímo ze zadání gramatiky:

FIRST(S) = { if, while, begin, s }
FIRST(T) = { else, ε }
FIRST(E) = { c }
FIRST(I) = { if }
FIRST(S′) = { ; , ε }
FIRST(E′) = { or, ε }

b) Konstrukce funkce FOLLOW:

graf relace ⊳ :

FOLLOW(S′) = { end }
FOLLOW(E) = { then, do }
FOLLOW(E′) = FOLLOW(E)
FOL(S) = { ε, end, ; }
FOL(I) = { else }
FOL(T) = ∅
FOLLOW(S) = FOLLOW(I) = FOLLOW(T) = { ε, ; , end, else }

Řídicí množiny pravidel:

D(S ⟶ IT) = { if }
D(S ⟶ begin SS′end) = { begin }
D(I ⟶ if E then S) = { if }
D(T ⟶ ε) = { ε, end, else, ; }
D(S′ ⟶ ε) = { end }
D(E′ ⟶ or cE′) = { or }
D(S ⟶ while E do S) = { while }
D(S ⟶ s) = { s }
D(T ⟶ else S) = { else }
D(S′ ⟶ ;SS′) = { ; }
D(E ⟶ cE′) = { c }
D(E′ ⟶ ε) = { then, do }

Gramatika není LL(1), neboť D(T ⟶ else S) ∩ D(T ⟶ ε) = { else }.
Závada vyplývá z dvojznačnosti řetězců typu if c then if c then s else s.

4.11

a) konstrukce funkce FIRST

	S	A	B	C
1	-	-	-	(, a
2	-	-	(, a	(, a
3	-	(, a	(, a	(, a
4	(, a	(, a	(, a	(, a

FIRST(S) = FIRST(A) = FIRST(B) = FIRST(C) = { (, a }.

b) konstrukce funkce FOLLOW

sestrojíme graf relace ⊳ :

funkce FOL:

FOL(S) = {ε} FOL(A) = { +,) } FOL(B) = { ♯ } FOL(C) = ∅ .

Z grafu relace ⊳ a funkce FOL dostáváme:

FOLLOW(S) = { ♯ }
FOLLOW(A) = { ε, +,) }
FOLLOW(B) = { ε, +,), ♯ }
FOLLOW(C) = { ε, +,), ♯ } .

4.12

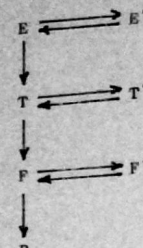
a) konstrukce funkce FIRST

	E	E′	T	T′	F	F′	P
1	-	+, ε	-	ε	-	*, ε	(,a,b,Λ
2	-	+, ε	-	ε	(,a,b,Λ	*, ε	(,a,b,Λ
3	-	+, ε	(,a,b,Λ	ε	(,a,b,Λ	*, ε	(,a,b,Λ
4	-	+, ε	(,a,b,Λ	ε,(,a,b,Λ	(,a,b,Λ	ε, ε	(,a,b,Λ
5	(,a,b,Λ	+, ε	(,a,b,Λ	ε,(,a,b,Λ	(,a,b,Λ	*, ε	(,a,b,Λ

FIRST(E) = FIRST(T) = FIRST(F) = FIRST(P) = { (,a,b,Λ } ,
FIRST(E′) = { +, ε } , FIRST(T′) = { ε, (,a,b,Λ }, FIRST(F′) = { *, ε } .

konstrukce funkce FOLLOW:

graf relace



funkce FOL

FOL(E) = {),ε}
FOL(E´) = ∅
FOL(T) = {+}
FOL(T´) = ∅
FOL(F) = {(,a,b,Λ}
FOL(F´) = ∅
FOL(P) = {*} .

FOLLOW(E) = FOLLOW(E´) = {),ε}
FOLLOW(T) = FOLLOW(T´) = {+,),ε}
FOLLOW(F) = FOLLOW(F´) = {(,a,b,Λ,+,),ε}
FOLLOW(P) = {*,(,a,b, ,+,),ε} .

b) řídicí množiny pravidel:

D(E ⟶ TE´) = {(,a,b,Λ}
D(E´⟶ +E) = {+}
D(E´⟶ ε) = {),ε}
D(T ⟶ FT´) = {(,a,b,Λ}
D(T´⟶ T) = {(,a,b,Λ}
D(T´⟶ ε) = {+,),ε}
D(F ⟶ PF´) = {(,a,b,Λ}
D(F´⟶ *F) = {*}
D(F´⟶ ε) = {(,a,b,Λ,+,),ε}
D(P ⟶ (E)) = {(}
D(P ⟶ a) = {a}
D(P ⟶ b) = {b}
D(P ⟶ Λ) = {Λ} .

Řídicí množiny pravidel splňují podmínky LL(1) gramatik.

LL(1)-analyzátor:

	+	*	(	)	a	b	Λ	ε
E	-	-	TE´;1	-	TE´;1	TE´;1	TE´;1	-
E´	+E;2	-	-	ε;3	-	-	-	ε;3
T	-	-	FT´;4	-	FT´;4	FT´;4	FT´;4	-
T´	ε;6	-	T;5	ε;6	T;5	T;5	T;5	ε;6
F	-	-	PF´;7	-	PF´;7	PF´;7	PF´;7	-
F´	ε;9	*F;8	ε;9	ε;9	ε;9	ε;9	ε;9	ε;9
P	-	-	(E);10	-	a;11	b;12	;13	-
+	kr	-	-	-	-	-	-	-
*	-	kr	-	-	-	-	-	-
(	-	-	kr	-	-	-	-	-
)	-	-	-	kr	-	-	-	-
a	-	-	-	-	kr	-	-	-
b	-	-	-	-	-	kr	-	-
Λ	-	-	-	-	-	-	kr	-
ε	-	-	-	-	-	-	-	př

(4.13)

Regulární jazyk lze reprezentovat deterministickým konečným automatem. Od automatu $A = (Q, \Sigma, \delta, q_0, F)$ lze standardním způsobem přejít k regulární gramatice G, která obsahuje pravidla tvaru $q \longrightarrow ap$, pokud $\delta(q,a) = p$ a pravidla $q \longrightarrow \varepsilon$, pokud $q \in F$. Tato gramatika už je LL(1), jak plyne z determinismu automatu A.

(4.14)

a) výpočet nad řetězcem aabbbbc

obsah zásobníku	symbol na vstupu	výstup
q_0	a	-
q_0 a q_2	a	-
q_0 a q_2 a q_7	b	-
q_0 a q_2 a q_7 A q_{10}	b	4
q_0 a q_2 a q_7 A q_{10} b q_{11}	b	-
q_0 a q_2 A q_3	b	3
q_0 a q_2 A q_3 b q_4	b	-
q_0 A q_1	b	3
q_0 A q_1 b q_8	b	-
q_0 A q_1 b q_8 b q_8	c	-
q_0 A q_1 b q_8 b q_8 c q_6	-	-
q_0 A q_1 b q_8 b q_8 B q_9	-	6
q_0 A q_1 b q_8 B q_9	-	5
q_0 A q_1 B q_5	-	1 přijmout

b) výpočet nad řetězcem aaabbb

obsah zásobníku	symbol na vstupu	výstup
q_0	a	-
q_0 a q_2	a	-
q_0 a q_2 a q_7	a	-
q_0 a q_2 a q_7 a q_7	b	-
q_0 a q_2 a q_7 a q_7 A q_{10}	b	4
q_0 a q_2 a q_7 a q_7 A q_{10} b q_{11}	b	-
q_0 a q_2 a q_7 A q_{10}	b	3
q_0 a q_2 a q_7 A q_{10} b q_{11}	b	-
q_0 a q_2 A q_3	b	3
q_0 a q_2 A q_3 b q_4	-	-
q_0 A q_1	-	3
-	-	2 ,přijmout

c) výpočet nad řetězcem aacbb

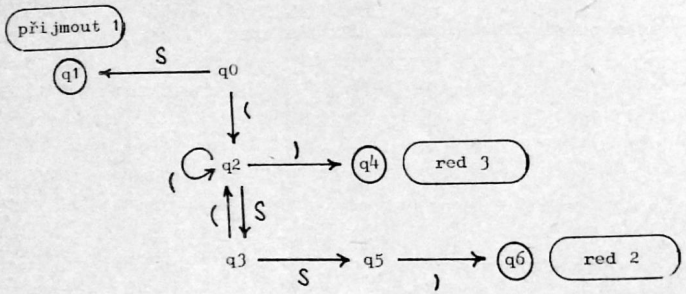
obsah zásobníku	symbol na vstupu	výstup
q_0	a	-
q_0 a q_2	a	-
q_0 a q_2 a q_7	c	-
		chyba

4.15

a) konstrukce položkového automatu

stav	položky	přechod symbolem	do stavu
q0	S'→.S	S	q1
	S→.(SS)	(	q2
	S→.()		
q1	S'→S.		
q2	S→(.SS)	S	q3
	S→(.)	)	q4
	S→.(SS)	(	q2
	S→.()		
q3	S→(S.S)	S	q5
	S→.(SS)	(	q2
	S→.()		
q4	S→().|		
q5	S→(SS.)	)	q6
q6	S→(SS).		

b) LR(0)-analyzátor

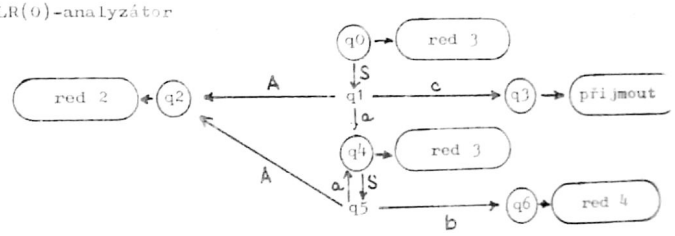


4.16

a) konstrukce položkového automatu

stav	položky	přechod symbolem	do stavu
q0	S'→.Sc	S	q1
	S→.SA		
	S→.		
q1	S→S.A	A	q2
	S'→S.c	c	q3
	A→.aSb	a	q4
q2	S→SA.		
q3	S→Sc.		
q4	A→a.Sb	S	q5
	S→.SA		
	S→.		
q5	A→aS.b	b	q6
	S→S.A	A	q2
	A→.aSb	a	q4
q6	A→aSb.		

b) LR(0)-analyzátor



4.17

a) konstrukce položkového automatu

stav	položky	přechod	do stavu
q0	S → .A	A	q1
	S → .A+B	B	q2
	A → .B		
	B → .B*C	C	q3
	B → .C	(	q4
	C → .(A)	a	q5
	C → .a		
q1	S → A.	+	q6
	S → A.+B		
q2	A → B.	*	q7
	B → B.*C		
q3	B → C.		
q4	C → (.A)	A	q8
	A → .A+B	B	q2
	A → .B		
	B → .B*C	C	q3
	B → .C	(	q4
	C → .(A)	a	q5
	C → .a		
q5	C → a.		
q6	A → A+.B	B	q9
	B → .B*C	C	q3
	B → .C	(	q4
	C → .(A)	a	q5
	C → .a		
q7	B → B*.C	C	q10
	C → .(A)	(	q4
	C → .a	a	q5
q8	C → (A.)	)	q11
	A → A.+B	+	q6
q9	A → A+B.	*	q7
	B → B.*C		
q10	B → B*C.		
q11	C → (A).		

Gramatika není LR(0): ve stavech q1, q2, q9 se vyskytují konflikty.

b) tabulka SLR(1)-analyzátoru

	(	)	+	*	a	ε	S	A	B	C
q0	q4	-	-	-	q5	-	-	q1	q2	q3
q1	-	-	q6	-	-	př. 0	-	-	-	-
q2	-	red 2	red 2	q7	-	red2	-	-	-	-
q3	-	red 4	red 4	red 4	-	red 4	-	-	-	-
q4	q4	-	-	-	q5	-	-	q8	q2	q3
q5	-	red 6	red 6	red 6	-	red 6	-	-	-	-
q6	q4	-	-	-	q5	-	-	-	q9	q3
q7	q4	-	-	-	q5	-	-	-	-	q10
q8	-	q11	q6	-	-	-	-	-	-	-
q9	-	red 1	red 1	q7	-	red 1	-	-	-	-
q10	-	red 3	red 3	red 3	-	red 3	-	-	-	-
q11	-	red 5	red 5	red 5	-	red 5	-	-	-	-

4.18

a) položkový automat

stav	položky	přechod symbolem	do stavu
q_0	Z → .A	A	q_1
	A → .aBcB	a	q_2
	A → .B	B	q_3
	A → .D	D	q_4
	B → .b	b	q_5
	B → .Ff	F	q_6
	D → .dE	d	q_7
	F → .b		
q_1	Z → A.		
q_2	A → a.BcB	B	q_8
	B → .b	b	q_5
	B → .Ff	F	q_6
	F → .b		
q_3	A → B.		
q_4	A → D.		
q_5	B → b.		
	F → b.		
q_6	B → F.f	f	q_9
q_7	D → d.E	E	q_{10}
	E → .FcA	F	q_{11}
	E → .FcE		
	F → .b	b	q_{12}
q_8	A → aB.cB	c	q_{13}

(pokračování)

q_9	B → Ff.		
q_{10}	D → dE.		
q_{11}	E → F.cA E → F.cE	c	q_{14}
q_{12}	F → b.		
q_{13}	A → aBc.B B → .b B → .Ff F → .b	B b F	q_{15} q_5 q_6
q_{14}	E → Fc.A E → Fc.E A → .aBcB A → .B A → .D B → .b B → .Ff D → .dE E → .FcA E → .FcE F → .b	A E a B D b F d	q_{16} q_{17} q_2 q_3 q_4 q_5 q_{18} q_7
q_{15}	A → aBcB.		
q_{16}	E → FcA.		
q_{17}	E → FcE.		
q_{18}	B → F.f E → F.cA E → F.cE	f c	q_9 q_{14}

Gramatika není LR(0): konflikt se objevuje ve stavu q_5. Okamžitě je zřejmé, že $c \in FOLLOW(B) \cap FOLLOW(F)$. Proto konflikt ve stavu q_5 neřeší ani informace poskytovaná funkcí FOLLOW. Gramatika tedy není ani SLR(1).

b) 1-položkový automat

stav	1-položky	přechod	do stavu
q_0	Z → .A,ε A → .aBcB,ε A → .B,ε A → .D,ε B → .b,ε B → .Ff,ε D → .dE,ε F → .b,f	A a B D b F d	q_1 q_2 q_3 q_4 q_5 q_6 q_7
q_1	Z → A.,ε		
q_2	A → a.BcB,ε	B	q_8

(pokračování)

	B → .b,c B → .Ff,c F → .b,f	b F	q_9 q_{10}
q_3	A → B.,ε		
q_4	A → D.,ε		
q_5	B → b.,ε F → b.,f		
q_6	B → F.f,ε	f	q_{11}
q_7	D → d.E,ε E → .FcA,ε E → .FcE,ε F → .b,c	E F b	q_{12} q_{13} q_{14}
q_8	A → aB.cB,ε	c	q_{15}
q_9	B → b.,c F → b.,f		
q_{10}	B → F.f,c	f	q_{16}
q_{11}	B → Ff.,ε		
q_{12}	D → dE.,ε		
q_{13}	E → F.cA,ε E → F.cE,ε	c	q_{17}
q_{14}	F → b.,c		
q_{15}	A → aBc.B,ε B → .b,ε B → .Ff,ε F → .b,f	B b F	q_{18} q_5 q_6
q_{16}	B → Ff.,c		
q_{17}	E → Fc.A,ε E → Fc.E A → .aBcB,ε A → .B,ε A → .D,ε B → .b,ε B → .Ff,ε D → .dE,ε E → .FcA,ε E → .FcE,ε F → .b,c,f	A E a B D b F d	q_{19} q_{20} q_2 q_3 q_4 q_{21} q_{22} q_7
q_{18}	A → aBcB,ε		
q_{19}	E → FcA.,ε		
q_{20}	E → FcE.,ε		

(pokračování)

q_{21}	B → b., F → b.,c,f		
q_{22}	B → F.f, E → F.cA, E → F.cA,	f c	q_{11} q_{17}

1-položkový automat konflikty neobsahuje. Gramatika je tedy LR(1). Její LR(1)-analyzátor je popsán následující tabulkou

	a	b	c	d	f	A	B	D	E	F	ε
q_0	q_2	q_5	-	q_7	-	q_1	q_3	q_4	-	q_6	-
q_1	-	-	-	-	-	-	-	-	-	-	red 0
q_2	-	q_9	-	-	red 9	-	q_8	-	-	q_{10}	-
q_3	-	-	-	-	q_{11}	-	-	-	-	-	red 2
q_4	-	-	-	-	-	-	-	-	-	-	red 3
q_5	-	q_{14}	q_{15}	-	-	-	-	-	q_{12}	q_{13}	red 4
q_6	-	-	red 4	-	red 9	-	-	-	-	-	-
q_7	-	-	-	-	q_{16}	-	-	-	-	-	-
q_8	-	-	q_{17}	-	-	-	-	-	-	-	red 5
q_9	-	-	red 9	-	-	-	q_{18}	-	-	q_6	red 6
q_{10}	-	q_5	-	q_7	-	q_{19}	q_3	q_4	-	q_{22}	-
q_{11}	-	q_5	red 5	-	-	-	-	-	q_{20}	-	-
q_{12}	-	q_{21}	-	q_7	-	-	-	-	-	-	red 1
q_{13}	q_2	-	-	-	-	-	-	-	-	-	red 7
q_{14}	-	-	-	-	-	-	-	-	-	-	red 8
q_{15}	-	-	-	-	red 9	-	-	-	-	-	red 4
q_{16}	-	-	red 9	-	-	-	-	-	-	-	-
q_{17}	-	-	q_{17}	-	q_{11}	-	-	-	-	-	-
q_{18}											
q_{19}											
q_{20}											
q_{21}											
q_{22}											

LITERATURA

[1] Hopcroft, J. E. - Ullman, J. D.: Formal Languages and Their Relation to Automata, Reading (Mass.), Addison-Wesley, 1969. (Slovenský překlad: Formálne jazyky a automaty, Bratislava, Alfa, 1978.)

[2] Chytil, M.: Automaty a gramatiky, Praha, SNTL, 1984.

[3] Chytil, M.: Teorie automatů a formálních jazyků, Praha, SPN 1978.

SBÍRKA ŘEŠENÝCH PŘÍKLADŮ
Z TEORIE AUTOMATŮ A FORMÁLNÍCH JAZYKŮ

RNDr. Michal Chytil, CSc.
Lektorovali: RNDr. Ivan Havel, CSc.
Vydala Univerzita Karlova, Praha 1987
jako skripta pro posluchače matematicko-fyzikální fakulty
Univerzity Karlovy
Dáno do tisku: srpen 1987
Vytiskly Tiskařské závody, n. p., Praha 1 - č. z. 21390
AA 9, 85 — VA 10,07 — 1. vydání — Náklad 100 výtisků
MŠ — MK 21 514/79
60 - 101 - 87 17/31
Brož. Kčs 8,50